Oracle® Communications Messaging Server

System Administrator's Guide

Release 8.1.0

F15146-02

July 2020

ORACLE®

Oracle Communications Messaging Server System Administrator's Guide, Release 8.1.0

F15146-02

Contents

3 Stopping and Starting Messaging Server

4 Configuring General Messaging Capabilities

5 Configuring and Administering Multiplexor Services

6 MTA Concepts

7 LMTP Delivery

8 Vacation Automatic Message Reply

9 Using and Configuring MeterMaid for Access Control

10 Implementing Greylisting by Using MeterMaid

11 MeterMaid Reference

17 Managing Logging

18 Monitoring Messaging Server

19 Monitoring the MTA

20 SNMP Support

21 Short Message Service (SMS)

Part II Improving Performance

35 Messaging Server Tuning and Best Practices

36 Tuning the mboxlist Database Cache in Unified Configuration

37 Best Practices for Messaging Server and ZFS

Part III Troubleshooting

38 Troubleshooting the MTA

42 Administering Very Large Mailboxes

43 Message Store Message Expiration

44 Configuring Message Expiration (Tasks)

62 Administering Message Store Database Snapshots (Backups)

63 Classic Messaging Server and Tiered Storage Overview

64 Message Store Command Reference

Part V Managing the Cassandra Message Store

65 Overview of Cassandra Message Store

66 About Elasticsearch

Preface

This guide explains how to administer Oracle Communications Messaging Server and its accompanying software components.

Audience

This document is intended for system administrators whose responsibility includes Messaging Server. This guide assumes you are familiar with the following topics:

- Messaging protocols
- Oracle Directory Server Enterprise Edition and LDAP
- System administration and networking
- General deployment architectures

Related Documents

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

- *Messaging Server Installation and Configuration Guide*: Provides instructions for installing and configuring Messaging Server.
- *Messaging Server Installation and Configuration Guide for Cassandra Message Store*: Provides instructions for installing and configuring the Cassandra message store.
- *Messaging Server Reference*: Provides additional information for using and configuring Messaging Server.
- *Messaging Server Release Notes*: Describes the new features, fixes, known issues, troubleshooting tips, and required third-party products and licensing.
- *Messaging Server Security Guide*: Provides guidelines and recommendations for setting up Messaging Server in a secure configuration.
- *Messaging Server MTA Developer's Reference*: Describes the Messaging Server MTA SDK.

Documentation Accessibility

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

Access to Oracle Support

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

Part I

Monitoring and Managing Messaging Server

Part I of *Messaging Server System Administrator's Guide* describes how to monitor and manage Oracle Communications Messaging Server.

Part I contains the following chapters:

- Handling Forged Email by Using the Sender Policy Framework
- Classic Message Store Directory Layout
- Monitoring LDAP Directory Server
- Monitoring System Performance
- Monitoring the Message Store
- Monitoring User Access to the Message Store
- Message Archiving
- Unified Messaging
- Messaging Server Command-Line Reference

1

Messaging Server System Administration Overview

This chapter provides an overview of Oracle Communications Messaging Server, and describes the basic administration tasks and tools used to perform those tasks.

About Messaging Server

Oracle Communications Messaging Server is an extensible framework of cooperative modules that creates an enterprise-wide, open standards–based, scalable electronic message handling system. This system is the combination of user message and transfer agents, message stores, and access units that together provide electronic messaging.

Messaging Server provides several messaging capabilities, including:

- **Message Store.** Provides a modular and scalable repository of user messages.
- **Message Transfer Agent.** Responsible for routing, transfer, and delivery of internet mail messages. Oracle Communications Messaging Server includes a fast, scalable, and flexible MTA that replaces the Sendmail utility bundled with most UNIX systems.
- **Message Access.** Provides client access to messages over standard protocols IMAP and POP3. HTTP-based access is offered for web clients and Lemonade Profile 1 is supported for mobile devices.

About Messaging Server Configuration

Messaging Server provides a set of tools called Unified Configuration to configure and administer the product. Unlike in legacy configurations, Unified Configuration uses validation to verify configuration accuracy, and employs a single tool to configure the entire Messaging Server configuration (with a few exceptions). See "Overview of Messaging Server Unified Configuration" for more information.

Note: This guide describes how to administer Messaging Server by using Unified Configuration.

Overview of Messaging Server Administration Tasks

A Messaging Server administrator is responsible for the day-to-day tasks of maintaining and managing Messaging Server and its users. The tasks also include managing Messaging Server components and potentially other Unified Communications Suite components.

You perform the following tasks as a Messaging Server administrator:

- Stopping and starting Messaging Server
- Managing user accounts
- Monitoring Messaging Server
- Tuning Messaging Server performance
- Migrating data to Messaging Server
- Backing up and restoring files
- Troubleshooting Messaging Server

About Messaging Server Administration Tools

Messaging Server provides the **msconfig** command-line utility for administering the server. See "Messaging Server Command-Line Reference" for more information. In addition, Messaging Server provides command-line utilities to manage the message store and the mail transfer agent (MTA). See *Messaging Server Reference* for more information.

Directory Placeholders Used in This Guide

Table 1–1 lists the placeholders that are used in this guide:

Table 1–1 Messaging Server Directory Placeholders

Placeholder	Directory
MessagingServer_home	Specifies the installation location for the Messaging Server software. The default is **/opt/sun/comms/messaging64**.
ConfigRoot	Specifies the location of the configuration files. The default is **/var/opt/sun/comms/messaging64/config**.
DataRoot	Specifies the location of the data files. The default is **/opt/sun/comms/messaging64/data**.

2

Overview of Messaging Server Unified Configuration

This chapter introduces Oracle Communications Messaging Server Unified Configuration, describes its capabilities, and provides initial guidelines on how to transition from a legacy configuration. Unified Configuration provides the ability to configure Messaging Server in a way that is much less error-prone and much easier to script and reuse across multiple hosts.

What Is Messaging Server Unified Configuration?

Unified Configuration is an improved process to configure and administer Messaging Server. Unlike in legacy configurations, Unified Configuration uses validation to verify configuration accuracy, and employs a single tool to configure the entire Messaging Server configuration (with a few exceptions).

Table 2–1 describes how Unified Configuration improves upon issues with legacy configuration.

Table 2–1 Legacy Versus Unified Configuration

Legacy Configuration	Unified Configuration Improvement
Dealing with many configuration files (with inconsistent formats) and hand-editing them can lead to errors and invalid configurations.	Unified Configuration "unifies" configuration management. The **msconfig** command tool administers Messaging Server configuration. Validation checking prevents introducing some configuration errors.
Configuration settings themselves are often complicated and not straight-forward.	Unified Configuration reduces redundancy and host specific-configurations, so that, for example, you can use the same settings for many options among the MMP, MTA, and message store configurations.
When problems arise, there are support challenges due to the many Messaging Server configuration files. Additionally, because passwords are contained in the configuration files, it makes it difficult for customers to just send these files to Oracle Support without first removing the passwords.	Unified Configuration uses only three text files to store configuration data, with most data stored in the **config.xml** file. Passwords are stored in a separate file, removing the need for customers to edit configuration files before sending to Oracle Support. In addition, Unified Configuration provides an audit trail of configuration changes. The changes (currently the last 100 changes) are actually stored in a repository, referred to as a *graveyard*. Storing of changes further enables you to restore an entire configuration, and to roll-back and roll-forward between configurations. See "Unified Configuration Files" for more information.

Table 2–1 (Cont.) Legacy Versus Unified Configuration

Legacy Configuration	Unified Configuration Improvement
It is difficult to separate instance-specific details from details shared by functionally similar machines.	Unified Configuration separates tasks for single instances (referred to as *instance*) of Messaging Server from global tasks for a group (referred to as *role*) of Messaging Server machines. The intention is that the role contain configuration information suitable for sharing with other hosts that have the same function in the deployment. At present, there is no mechanism to automatically share role configuration. See "Separating Roles and Instances" for more information.
Customers must create their own procedures and scripts with their own tools to manage a deployment.	New customers can write automation scripts by using the Unified Configuration *recipe language*. The recipe language introduces the ability to robustly change a configuration in a reproducible fashion in a way that can be sensitive to what was previously in the configuration. When you use recipes, you are able to use the Unified Configuration history and administrative undo features. In addition, Oracle can use the recipe language to automate configuration changes that are otherwise complex, interconnected, and require lots of documentation. The **SpamAssassin.rcp**, **HAConfig.rcp**, and **LMTPSingleSystem.rcp** recipes, available in the *MessagingServer_home*/**lib/recipes** directory, are good examples. See "Using Recipes" for more information.

Unified Configuration Files

Table 2–2 describes the Unified Configuration file names, file management tool, file format, character set, XML schema, ownership, and file permissions. The Unified Configuration files are located in the *ConfigRoot* directory by default.

Table 2–2 Unified Configuration File Properties

Configuration File	Description	File Management Tool	File Format	Char Set	File Ownership	Recommended File Permissions
restricted.cnf	Contains protected Messaging Server UID and GID information.	Text editor	option=value	ASCII	**root**	0644

Table 2–2 (Cont.) Unified Configuration File Properties

Configuration File	Description	File Management Tool	File Format	Char Set	File Owner ship	Recommended File Permissions
xpass.xml	Contains obfuscated passwords (*BASE64* encoded). This is the only file within Unified Configuration where password information is stored.	**msconfig** utility	XML 1.0	UTF-8	**mailsrv**	0600
config.xml	Contains most of the non-password configuration information. In addition, when necessary, you could send this entire file to Oracle Support to help with resolving problems.	**msconfig** utility	XML 1.0	UTF-8	**mailsrv**	0640
configlib.xml	Contains static, default mapping tables that are mostly concerned with character sets and language issues, including: ▪ **DISPOSITION_ LANGUAGE** ▪ **DOMAIN_DC** ▪ **LANGUAGE_ LOCALES** ▪ **LDAP_USERS_ LANGUAGE** ▪ **LDAP_USERS2_ LANGUAGE** ▪ **NOTIFICATION_ LANGUAGE**	Managed by Oracle (that is, the file is not to be edited)	XML 1.0	UTF-8	**mailsrv**	0644

Notes:

- When you perform the initial Messaging Server configuration, the **restricted.cnf** file sets the UID under which to run Messaging Server. After initial configuration, there should rarely be a need to edit this file. A legacy configuration can also use the **restricted.cnf** file for enhanced security.
- Never edit the **configlib.xml** file. Doing so causes an unsupported configuration.

Note: About MTA Tailor Options

In legacy configuration, you use the MTA tailor file of option settings (**imta_tailor**) to set various MTA installation and operational parameters. In Unified Configuration, the MTA tailor file is obsolete and no longer used. Unified Configuration replaces the MTA tailor options that specified locations of MTA directories or files with rationalized, consistent locations, which are based off the installation main location and located by using the **SERVERROOT** environment variable. Legacy configuration MTA tailor options that set other sorts of MTA operational parameters have typically been replaced with Unified Configuration options of the form **mta.***option-name*.

Enabling Unified Configuration in Messaging Server

There are two ways to enable Unified Configuration:

1. **When migrating to Unified Configuration**: Use the *MessagingServer_home*/**bin/configtoxml** program to migrate a legacy configuration to Unified Configuration. When you run **configtoxml**, your old configuration is converted to Unified Configuration.
   - The legacy configuration is saved in the *ConfigRoot*/**legacy-config** directory.
   - If necessary, you can use the **configtoxml -undo** command to restore a saved legacy configuration.
2. **For a new Messaging Server instance**: Run the **init-config** command to enable Unified Configuration by default.
   - The presence of a **config.xml** file in the **config** directory indicates that Unified Configuration is enabled.
   - When you perform a fresh installation of Messaging Server and choose to configure a Unified Configuration, you cannot revert that Unified Configuration to a legacy configuration. If, however, you upgrade Messaging Server and convert to a Unified Configuration (by running the **configtoxml** command), you can revert back to the legacy configuration.
   - A Unified Configuration is more simple than a legacy configuration. In addition, where appropriate, modern default values are established and seldom used features are removed (for example, the **tcp_tas** channel is not present in Unified Configuration).
   - Legacy configuration files such as **dispatcher.cnf**, **option.dat**, and so on, are ignored when Unified Configuration is enabled.

To Determine if Unified Configuration Is Deployed

The following example shows how to determine if Unified Configuration is deployed on your system:

```
cd /opt/sun/comms/messaging64/bin
imsimta version
...
Using /opt/sun/comms/messaging64/config/config.xml
SunOS host2.example.com 5.10 Generic_142901-03 i86pc i386 i86pc
```

In this example, the presence of the **config.xml** file indicates that Unified Configuration has been enabled on this host.

If you are using a compiled configuration and see in that the status is not compiled, you should recompile the configuration. For example:

```
/opt/sun/comms/messaging64/bin/imsimta version
...
Using /opt/sun/comms/messaging64/config/config.xml (not compiled)
SunOS host1.example.com 5.10 Generic_147441-09 i86pc i386 i86pc

/opt/sun/comms/messaging64/bin/imsimta cnbuild
```

See *Messaging Server Reference* for more information.

Understanding Unified Configuration Limitations

In general, Unified Configuration has consolidated all the various Messaging Server files. Nevertheless, the current Unified Configuration implementation has a few limitations:

- The channel sieves and channel header trimming option files have not yet been converted to XML.
- Some files, such as localized templates for DSNs and NDNs, might remain in their current format and not be converted to XML.
- The content of the **conversions** file is a mono-block in XML.
- The **msconfig** tool does not issue a warning when an option that requires a restart is modified. The **configutil** tool does issue such a warning.

Using the Repository of Previous Configurations

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

To List Configurations

- Use the **msconfig history** command to show a list of configurations currently in the graveyard.

To Compare Configurations

- To compare configurations, use the **msconfig differences** *m _n* command, where *m* and *n* are the numbers of the previous configurations from the **history** command that you want to compare.

Using Legacy Configuration Tools with Unified Configuration

Once Unified Configuration in enabled, legacy configuration tools might work differently than in previous releases. Specifically:

- Scripts that directly alter legacy configuration files do not work in a Unified Configuration.
- The **configutil** command can still set, get, and delete a legacy configuration. Additionally, in Unified Configuration, **configutil** options can also automatically perform:
  - Option name translations
  - Option value translations
  - Option value validations

> **Note:** Use of the **configutil** command is deprecated in Unified Configuration mode. Some configuration changes that were previously possible with the **configutil** command are only possible by using the **msconfig** command, for example, some changes to notification configuration.

- The "mkbackupdir" command, which creates and synchronizes the backup directory with the information in the message store, works with Unified Configuration.
- The imsmita program command, which manipulates the program delivery options, does not work with Unified Configuration. You must use the **msconfig** command instead.
  - This command issues an error and exits with **1** when Unified Configuration is being used.
- All other utilities only consume options and continue to work with both Unified Configuration and legacy configurations.

Separating Roles and Instances

Unified Configuration separates tasks for single instances (referred to as *instance*) of Messaging Server from global tasks for a group (referred to as *role*) of Messaging Server machines. The intention is that the role contain configuration information suitable for sharing with other hosts that have the same role in the deployment.

> **Note:** At present, there is no mechanism to automatically share role configuration.

Any configuration option can be an instance setting, a role setting, or both. When the same option is in both the role and instance, the instance value takes precedence. Both the **init-config** and **msconfig** commands put a given setting in either the instance or the role based on the likely scope for the option. Normally, you use the default location (instance or role) determined by the **msconfig** command and not explicitly specify one or the other.

Initial configuration generates an instance and role, as does migrating from a legacy configuration to Unified Configuration.

More About Unified Configuration Options

This section provides information about password, restricted, and obsolete options.

Options That Have Passwords

The password options have the following characteristics:

- Options can be marked as being a password.
- By default, password values are not displayed.
- The passwords are stored in obfuscated form in the **xpass.xml** file. Because Messaging Server stores passwords in a separate file, you do not need to edit configuration files to remove them before sharing with Oracle Support.

Restricted Options

Some configuration options are marked "restricted" by Oracle. Additionally, the XML schema exposes the entire configuration, so there are no longer any hidden configuration options.

Options may be restricted for several reasons, including:

- The option has complex and subtle consequences and would cause harm in all but a very few rare circumstances.
- The option might be a legacy option that should not be used in new systems.
- The option might be a placeholder for a feature that has not yet been implemented.

Restricted options have the following characteristics:

- The **msconfig** command displays a warning when you attempt to set a restricted option. In addition, the **msconfig** command requires an extra step to actually set the restricted option.
- The restriction is noted within the configuration file itself, which helps you to be aware of any special circumstances. For example:

```
<delimiter_char v="127" xannotation="RESTRICTED USAGE OPTION: user remark"
xauthor="dcn@example.com" xmtime="2010-05-12T17:42:19-08:00"/>
```

The **user remark** text is any optional remark added by the administrator when the configuration was updated. The **"RESTRICTED USAGE OPTION"** is text inserted by Oracle into the remark field when the option is restricted.

Caution: If you set a restricted option without being advised to do so by Oracle Support, your configuration is considered unsupported by Oracle.

Obsolete Options

When an option has been marked by Oracle as being obsolete, the configuration no longer uses it. However, you cannot remove it from the XML Schema as that would make existing configurations invalid.

When marked as obsolete, the option:

- Remains in the XML Schema
- Can no longer be set or changed
- Can only be deleted from a configuration

Option Relationships

Unified Configuration enables some relationships between options to be expressed so that when a particular option is set, other unnecessary options can be automatically removed. For example, if you set the **mx** option on a channel (for MX mail forwarding records), any of the **nomx**, **randommx**, and other related "mx" options are removed. In addition, Unified Configuration uses the concept of default relationships to help with configuration. For example, option X and option Y might have a default relationship such that when X is not set, the value is taken from Y; or when Y is not set, then Y's default value is *value*. Furthermore, Unified Configuration has the capability to know in which release an option became available and warn when a certain configuration is not release-suitable. In general, option relationships help to reduce configuration mistakes.

Unified Configuration Option Names

Unified Configuration uses a "unified" option naming convention that is reminiscent of legacy **configutil** option names.

In general, this option naming convention uses the following structure:

[*role.*]*group*[*.sub-group* | *.sub-group*].*option*

[*instance.*]*group*[*.sub-group* | *.sub-group*].*option*

The following example shows a *group.sub-group.option* convention:

```
imap.logfile.flushinterval
```

In this example, **imap** is the group, **logfile** is the sub-group, and **flushinterval** is the option.

This example shows a *group.option* convention:

```
mta.mm_debug
```

In this example, **mta** is the group and **mm_debug** is the option.

Characteristics about option names to keep in mind:

- Many groups only appear once (for example, **imap** and **pop**).
- Some groups may appear many times. For example:

  ```
  channel
  mapping
  sectoken
  alias
  task
  ```

- The group or sub-group can include a *:name* portion used for "named" groups. For example:

  ```
  channel:tcp_local.slave_debug
  partition:primary.path
  ```

Characteristics about instances and roles to keep in mind:

- An option in the "instance" overrides the same option in the "role." For example, IMAP is effectively disabled by this configuration:

  ```
  instance.imap.enable = 0
  role.imap.enable = 1
  ```

- There actually is no option called **imap.enable**. It is either **role.imap.enable** or **instance.imap.enable**.
- When setting options, you typically do not specify either "role" or "instance." The **msconfig** command applies heuristics to determine whether "role" or "instance" applies. Here is a sample, basic **config.xml** file that shows how the configuration uses instance and role:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xconfig...>
<role name="store">
<base>
<defaultdomain v="example.com"/> role.base.defaultdomain
</base>
<imap>
<enable v="1"/> role.imap.enable
<numprocesses v="2"/> role.imap.numprocesses
</imap>
<mapping name="ABC">
<rule pattern="x*y" template="$N"/> role.mapping:ABC.rule
<rule pattern="*" template="$Y"/> role.mapping:ABC.rule
</mapping>
</role>
<instance name="ims" roleref="store">
<base>
<hostname v="wassonite.example.com"/> instance.base.hostname
</base>
<mta>
<mm_debug v="5"/> instance.mta.mm_debug
</mta>
</instance>
</xconfig>
```

In the preceding example, some option names that you would see upon listing them with the **msconfig show** command are displayed in bold. Also, you can see that the default domain (**defaultdomain**), number of IMAP processes (**numprocesses**), and mappings (**mapping name**) have been defined for the store role; and that the host name (**hostname**) and MTA logging debug level (**mm_debug**) have been set for the store instance.

> **Tip:** Use the **configutil -H** command to translate the legacy **configutil** option names to Unified Configuration names. For example:
>
> ```
> configutil -H -o logfile.imap.expirytime
> Configuration option: logfile.imap.expirytime
> Unified Config Name: imap.logfile.expirytime
> ```

Example of Legacy Configuration and Unified Configuration

Unified Configuration greatly simplifies the configuration process, as shown in this example of configuring the SMTP server for debugging.

In a legacy configuration, you must perform the following steps:

1. Edit the **imta.cnf** file and modify the **tcp_local** channel entry:

```
tcp_local identnonenumeric inner loopcheck maysaslserver maytlsserver mx \
pool SMTP_POOL remotehost saslswitchchannel tcp_auth smtp sourcespamfilter1 \
switchchannel master_debug
```

```
tcp_local-daemon
```

2. Edit the **option.dat** file and add the following option:

```
mm_debug=3
```

3. Edit (or create if it does not exist) the **tcp_local_option** file and add the following option:

```
trace_level=2
```

4. Check permissions on all files, especially the **tcp_local_option** file.

In Unified Configuration, the equivalent steps to the preceding task are the following:

```
% msconfig
msconfig> set channel:tcp_local.slave_debug
msconfig# set mm_debug 3
msconfig# set channel:tcp_local.options.trace_level 2
msconfig# write
```

Using Recipes

You use recipe files, which are expressed by using a programming language, to automate configuration tasks, typically by scripting them. Recipes are located in the *MessagingServer_home*/**lib/recipes** directory. The primary inspiration for the recipe language is the Icon programming language designed by Ralph Griswald. As such, it supports C-like expressions, operators, and assignments, Sieve-like conditionals, and loops. The available data types are integers, strings, and lists.

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

By using the recipe language, you can more easily script complex configuration changes. For more information on writing recipes, see the help text for the recipe language by typing **msconfig -help** and choosing help for the **Recipe_language** topic, or refer to the discussion on recipe language in *Messaging Server Reference*.

To Run a Recipe

To run a recipe, type the following command:

```
msconfig run recipe_name
```

Helpful Commands

This section provides some helpful commands to get started with Unified Configuration.

To Show Settings

Use the **msconfig show** command to display current settings. For example, to show all currently enabled options:

```
msconfig show *enable
role.watcher.enable = 1
role.schedule.enable = 1
```

```
role.store.enable = 1
role.store.purge.enable = 1
role.imap.enable = 1
role.pop.enable = 1
role.mta.enable = 1
role.dispatcher.service:SMTP.enable = 1
role.dispatcher.service:SMTP_SUBMIT.enable = 1
role.mmp.enable = 0
role.ens.enable = 1
role.http.enable = 1
```

To Get Help

Use the **msconfig help** command to display help text.

3

Stopping and Starting Messaging Server

This chapter describes how to stop and start Oracle Communications Messaging Server services.

Starting and Stopping Services

Topics in this section:

- To Start and Stop Messaging Server Services
- To Start Up, Shut Down, or View the Status of Messaging Services
- Starting and Stopping a Messaging Server Running in MTA-only Mode

To stop and start Messaging Server services installed in a highly available environment, see "Stopping and Starting Messaging Services in an HA Environment".

To Start and Stop Messaging Server Services

Start and stop Messaging Server services from the command line by using the following commands:

```
MessagingServer_home/bin/start-msg
MessagingServer_home/bin/stop-msg
```

Though you can use the command template to start and stop services individually (*MessagingServer_home*/**bin/stop-msg** *service* (where *service* can be **mta**, **imap**, **pop**, **store**, **http**, **ens**, **sched**, **purge**, **mfagent**, **snmp**, **mmp**, **sms**, **metermaid**, **cert**, **dispatcher**, **job_controller** or **watcher**)), do not do so except in specific tasks as described. Certain services have dependencies on other services and must be started in a prescribed order. Complications can arise when trying to start services on their own. For this reason, you should start and stop all the services together by using the **start-msg** and **stop-msg** commands.

Note: You must first enable services before starting or stopping them. See "Enabling and Disabling Services" for more information.

Important: If a server process crashes, other processes might hang as they wait for locks held by the server process that crashed. If you are not using automatic restart (see "Automatic Restart of Failed or Unresponsive Services"), and if any server process crashes, it is generally safer to stop **all** processes, then restart **all** processes. This includes the POP, IMAP, and MTA processes, as well as the **stored** (message store) process, and any utilities that modify the message store, such as **mboxutil**, **deliver**, **reconstruct**, **readership**, or **upgrade**.

To Start Up, Shut Down, or View the Status of Messaging Services

Do not shut down individual services except in the specific tasks as described. Certain services have dependencies on other services and must be started in a prescribed order. Complications can arise when trying to start services on their own. For this reason, you should start and stop all the services together by using the **start-msg** and **stop-msg** commands.

However, when you make a configuration change that requires restart of a service, and you want to minimize service disruption, then use **stop-msg***service* followed by **start-msg***service*. For example, when changing an MTA option that requires a restart of the dispatcher but does not require a restart of the job_controller, it is better to run **stop-msg dispatcher; start-msg dispatcher** to avoid unnecessarily flushing the job_controller's cache. See Messaging Server Reference Guide for more information on the **start-msg** and **stop-msg** commands.

The services must be enabled to stop or start them. See "To Specify What Services Can Be Started" for more information.

To Specify What Services Can Be Started

By default the following services are started with **start-msg**:

```
start-msg
Connecting to watcher ...
Launching watcher ... 9347
Starting store server .... 9356
Checking store server status ..... ready
Starting purge server .... 9413
Starting imap server .... 9420
Starting pop server .... 9425
Starting http server .... 9437
Starting sched server ... 9451
Starting dispatcher server .... 9461
Starting job_controller server .... 9466
```

The complete list of scopes with an "enable" option is as follows. Some of these, such as **notifytarget**, **service**, and **task** are named scopes. See the discussion in *Messaging Server Reference Guide* on scope syntax in **msconfig** command for more information.

```
autorestart
notifytarget
ens
folderquota
imap
indexer
messagetype
msghash
dbreplicate
```

```
mta
metermaid
mmp
pab
pop
purge
relinker
schedule
service
smime
sms_gateway
snmp
store
task
typequota
watcher
http
```

Set both **imap.enable** and **imap.enablesslport** to *0* to disable IMAP. The same goes for POP and HTTP. See *Messaging Server Reference* for details.

Starting and Stopping a Messaging Server Running in MTA-only Mode

To start an MTA-only system, you should also start **imsched**. Before you do this, remove any scheduled jobs that are not appropriate to your installations.

imsched is an individual component of Messaging Server that must be started separately if you are not starting all of Messaging Server. If you start your MTA-only system by using **start-msg imta** or **start-msg mta**, then you do not run the **imsched** process.

To run messaging server in MTA mode only (no store, imap, or pop processes), you can either select the MTA to be only installed and configured during the Messaging Server configuration after initial install (*MessagingServer_home***/bin/configure**), or manually disable the message store and **mshttp** process by using the following commands:

```
msconfig set store.enable 0
msconfig set http.enable 0
```

Once you have disabled HTTP and other store processes, you can then start Messaging Server by running the following command:

```
start-msg
Connecting to watcher ...
Launching watcher ... 4034
Starting ens server ... 4035
Starting sched server ... 4036
Starting dispatcher server .... 4038
Starting job_controller server .... 4042
```

All the appropriate processes are started, including **imsched** and **imta**. This way you do not have to remember to start the **sched** process.

Stopping and Starting Messaging Services in an HA Environment

While Messaging Server is running under HA control, you cannot use the normal Messaging Server start, restart, and stop commands to control individual Messaging Server services. For example, if you attempt a **stop-msg** in an HA deployment, the

system warns that it has detected an HA setup and informs you how to properly stop the system.

The appropriate HA start, stop, and restart commands are shown in Table 3–1, Table 3–2, Table 3–3, and Table 3–4. There are no specific HA commands to individually start, restart, or stop other Messaging Server services (for example, SMTP). However, you can run a **stop-msg***service* command to stop/restart individual servers such as **imap**, **pop** or **sched**.

The finest granularity in Oracle Solaris Cluster (formerly known as Sun Cluster) is that of an individual resource. Because Messaging Server is known to Oracle Solaris Cluster as a resource, the Oracle Solaris Cluster **scswitch** commands affect all Messaging Server services as a whole.

Table 3–1 Start, Stop, Restart in an Oracle Solaris Cluster 3.0/3.1 Environment

Action	Individual Resource	Entire Resource Group
Start	**scswitch -e -j***resource*	**scswitch -Z -g***resource_group*
Restart	**scswitch -n -j***resource***scswitch -e -j***resource*	**scswitch -R -g***resource_group*
Stop	**scswitch -n -j***resource*	**scswitch -F -g***resource_group*

Table 3–2 Start, Stop, Restart in an Oracle Solaris Cluster 3.2 Environment

Action	Individual Resource	Entire Resource Group
Start	**clrs online***resource*	**clrg online***resource_group*
Restart	**clrs disable***resourceclrs enableresource*	**clrg restart***resource_group*
Stop	**clrs offline***resource*	**clrg disable***resource_group*

Table 3–3 Start, Stop, Restart in Veritas 3.5, 4.0, 4.1 and 5.0 Environments

Action	Individual Resource	Entire Resource Group
Start	**hares -online***resource_name* **-sys** *system_name*	**hagrp -online***group_name* **-sys** *system_name*
Restart	**hares -offline** *resource_name* **-sys** *system_name* **hares -online** *resource_name* **-sys** *system_name*	**hagrp -offline** *service_group* **-sys** *system_name* **hagrp -online** *service_group* **-sys** *system_name*
Stop	**hares -offline** *resource_name* **-sys** *system_name*	**hagrp -offline** *service_group* **-sys** *system_name*

Table 3–4 Start, Stop, Restart in an Oracle Clusterware 12.1 Environment

Action	Individual Resource
Start	**crsctl start resource** *MS resource* **-n** *system*
Restart	**crsctl stop resource** *MS resource* **-n** *system* **crsctl start resource** *MS resource* **-n** *system*
Stop	**crsctl stop resource** *MS resource* **-n** *system*

Automatic Restart of Failed or Unresponsive Services

This section describes how Messaging Server monitors and automatically restarts unresponsive services.

Overview of Messaging Server Monitoring Processes

Messaging Server provides two processes called **watcher** and **msprobe** that transparently monitor services and automatically restart them if they crash or become unresponsive (the services hangs). **watcher** monitors server crashes. **msprobe** monitors non-responsive server processes by checking their response times. When a server fails or stops responding to requests, it is automatically restarted. Table 3–5 shows the services monitored by each utility.

Table 3–5 Services Monitored by watcher and msprobe

watcher (crash)	msprobe (unresponsive hang)
IMAP, POP, HTTP, job controller, dispatcher, message store (**stored**), **imsched**, MMP. (LMTP/SMTP servers are monitored by the dispatcher and LMTP/SMTP clients are monitored by the job_controller.) The watcher also monitors all processes that access the message store in such a way that they could hold outstanding message store locks when they crash. This includes **ims_master**, **lmtp_server**, and store utilities.	IMAP, POP, HTTP, cert, job controller, message store (**stored**), **imsched**, ENS, LMTP, SMTP

Enabling the watcher (**watcher.enable 1**, default) monitors process failures and unresponsive services and logs error messages to the **default** log file indicating specific failures. To enable automatic server restart, use **msconfig** to set **base.autorestart.enable 1**. By default, this option is set to no (*0*).

If any of the message store services fail or freeze, all message store services that were enabled at start-up are restarted. For example, if **imapd** fails, at the least, **stored** and **imapd** are restarted. If other message store services were running, such as the POP or HTTP servers, then those are restarted as well, whether or not they failed.

Automatic restart also works if a message store utility fails or freezes. For example, if **mboxutil** fails or freezes, the system automatically restarts all the message store servers. However, it does not restart the utility. **msprobe** runs every 10 minutes. Service and process restarts are performed once within a 10-minute period (and are configurable by using **base.autorestart**). If a server fails more than once during this designated period of time, then the system stops trying to restart this server. If this happens in an HA system, Messaging Server is shut down and a failover to the other system occurs.

Whether or not **base.autorestart.enable** is enabled, the system still monitors the services and sends failure or non-response error messages to the console and *DataRoot*/**log/watcher** listens to port 49994 by default, but this is configurable with the **watcher.port** option.

A watcher log file is generated in *DataRoot*/**log/watcher**. This log file is not managed by the logging system (no rollover or purging) and records all server starts and stops. The following is an example log:

```
watcher process 13425 started at Mon June 4 11:23:54 2012

Watched 'imapd' process 13428 exited abnormally
```

```
Received request to restart:  store imap pop http
Connecting to watcher ...
Stopping http server 13440 .... done
Stopping pop server 13431 ... done
Stopping pop server 13434 ... done
Stopping pop server 13435 ... done
Stopping pop server 13433 ... done
imap server is not running
Stopping store server 13426 .... done
Starting store server .... 13457
checking store server status ...... ready
Starting imap server ..... 13459
Starting pop server ....... 13462
Starting http server ...... 13471
```

See "Monitoring Using msprobe and watcher Functions" for more details on how to configure this feature.

msprobe is controlled by **imsched**. If **imsched** crashes, this event is detected by **watcher** and triggers a restart (if autorestart is enabled). However, in the rare occurrence of **imsched** hanging, you must kill **imsched** with a **kill***imsched_pid* command, which causes the watcher to restart it.

Automatic Restart in High Availability Deployments

Table 3–6 shows the configuration options to be set for automatic restart in high availability deployments:

Table 3–6 HA Automatic Restart Options

Option	Description/ HA Value
watcher.enable	Enable watcher on **start-msg** startup. Default is enabled (**1**).
base.autorestart.enable	Enable automatic restart of failed or frozen (unresponsive) servers including IMAP, POP, HTTP, job controller, dispatcher, and MMP servers. Default is enabled).
base.autorestart.timeout	Failure retry time-out. If a server fails more than once during this designated period of time, then the system stops trying to restart this server. If this happens in an HA system, Messaging Server is shutdown and a failover to the other system occurs. The value (set in seconds) should be set to a period value longer than the msprobe interval. (See **schedule.task:** in the following section). Default is 600.
schedule.task:msprobe.crontab	**msprobe** runs schedule. A crontab style schedule string. Default is **5,15,25,35,45,55 * * * * lib/msprobe**. To disable, run the following command:**msconfig set schedule.task:msprobe.enable 0**

4

Configuring General Messaging Capabilities

This chapter describes the general Oracle Communications Messaging Server tasks, such as configuring directory access by using command-line utilities. Tasks specific to administering individual Messaging Server services, such as POP, IMAP, HTTP, and SMTP, are described later in this guide.

Modifying Your Passwords

If you set up multiple administrators with the same password during the initial Messaging Server configuration, you might want to change the passwords of those administrators.

Table 4–1 shows the password options that are set up during initial runtime configuration. Use the **msconfig** command to make changes to the Messaging Server configuration, or **ldapmodify** to update information stored in Directory Server.

Table 4–1 Passwords Set in Messaging Server Initial Runtime Configuration

Option	Description
base.ugldapbindcred	Password for the Messaging Server LDAP user/group access account (**base.ugldapbinddn**). Use **msconfig** to change.
base.proxyadminpass	Password for the Proxy Administrator account (**base.proxyadmin**), which is used to provide proxy authentication access to end-user mailboxes. Use **msconfig** to change.
http.smtpauthpassword	Password used when **mshttpd** submits mail to the MTA. Set by initial configuration to the same password as **base.proxyadminpass**. Use **msconfig** to change.
SSL passwords for key files	Passwords that are stored in the **xpass.xml** file. Use the **msconfig set -prompt "sectoken:Internal (Software) Token"** command to change. This command causes **msconfig** to prompt for the password without an echo.
Admin Account credentials	The "admin" account is both in the service administrator group by default and is a store admin by default. You are prompted for this password during initial configuration. By default, the "admin" account is used for proxy and SMTP authentication, so this password needs to match the settings for **base.proxyadminpass** and **http.smtpauthpassword**.

Table 4–1 (Cont.) Passwords Set in Messaging Server Initial Runtime Configuration

Option	Description
Messaging End User Administrator	This is the LDAP user for this specific host. The **base.ugldapbindcred** entry and the "Messaging End User Administrator" actually refer to the same password, which is set both in the option and in the **userPassword** attribute for that user in the LDAP directory. The password is generated randomly by initial configuration and is only used by one single Messaging Server host to bind to the LDAP directory server to perform searches.

The following example uses the **proxyadminpass** option to change the password of the Proxy Administrator account. You should not set passwords from the command line, so this example shows using **msconfig** in interactive mode.

```
msconfig
msconfig> set -prompt proxyadminpass
Password:
Verify:
msconfig# write
msconfig> exit
```

Managing Mail Users, Mailing Lists and Domains

User, mailing list, and domain information is stored as entries in an LDAP directory. An LDAP directory can contain a wide range of information about an organization's employees, members, clients, or other types of individuals that in one way or another "belong" to the organization. These individuals constitute the *users* of the organization.

Overview of Messaging Server and LDAP

In the LDAP directory, the information about users is structured for efficient searching, with each user entry identified by a set of attributes. Directory attributes associated with a user can include the user's name and other identification, division membership, job classification, physical location, name of manager, names of direct reports, access permission to various parts of the organization, and preferences of various kinds.

In an organization with electronic messaging services, many if not all users hold mail accounts. Messaging Server stores copies of some account information (uid and quota in particular) on local servers. In general, the LDAP directory is considered authoritative for account information by Messaging Server. Once account information for a mail user is present in the LDAP directory, then the mail server named in the **mailHost** attribute automatically creates that user without any additional mail server specific configuration.

Creating and managing mail users and mailing lists consists of creating and modifying user and mailing list entries in the LDAP directory. This is done by using the Delegated Administrator GUI or command-line utilities, or by directly modifying the LDAP directory information.

Note: In general, the Messaging Server documentation does not describe how to directly modify the LDAP directory. Consult the Directory Server documentation for more information.

To Remove a User from Messaging Server by Using Delegated Administrator

1. Mark the user as deleted by running the **commadmin user delete** command. (For more information, see the discussion about removing users, groups, and services from a domain in *Delegated Administrator System Administrator's Guide*.)
2. Remove services from the user. A service can be a mailbox or a calendar. For the current version of Messaging Server, the program is called "msuserpurge".
3. Permanently remove the user, by invoking the **commadmin domain purge** command.

To Remove a Domain from Messaging Server using Delegated Administrator

1. Mark the domain as deleted by running the **commadmin domain delete** command. (See the discussion on removing users, groups, and services from a domain in *Delegated Administrator System Administrator's Guide* for more information.)
2. Remove services from the users of that domain. A service can be a mailbox or a calendar. For Messaging Server, the program is called "msuserpurge".
3. Permanently remove the domain, by invoking the **commadmin domain purge** command.

Scheduling Automatic Tasks

Messaging Server enables you to schedule automatic tasks, such as running the **imexpire** command at predetermined times.

Overview of Scheduling Automatic Tasks

Messaging Server provides a general task scheduling mechanism by using a process called **imsched**. It is intended for scheduling Messaging Server processes. It is enabled by setting the **schedule.task** option. If you modify the schedule, either restart the scheduler with the command **stop-msg sched** and **start-msg sched**, or refresh the scheduler process (**refresh sched**).

This option requires a command and a schedule on which to execute the command. The format is as follows:

```
schedule.task:taskname.crontab = schedule
```

where:

- *taskname* is the name of the command to run, for example, **expire**, **msprobe**, and so on.
- *schedule* is a non-empty string with the following format:

  ```
  minute hour day-of-month month-of-year day-of-week command args
  ```

- *command args* can be any Messaging Server command and its arguments. Paths can be relative to *MessagingServer_home* or absolute paths. See "Pre-defined Automatic Tasks" for relative path examples.

minute hour day-of-month month-of-year day-of-week is the schedule for running the command. It follows the UNIX **crontab** time format.

The values are separated by a space or tab and can be 0-59, 0-23, 1-31, 1-12 or 0-6 (with 0=Sunday) respectively. Each time field can be either an asterisk (meaning all legal

values), a list of comma-separated values, or a range of two values separated by a hyphen. Days can be specified by both day of the month and day of the week and both are required if specified. For example, setting the 17th day of the month and Tuesday only runs the command on the 17th day of a month when it is Tuesday.

If you modify scheduler, either restart the scheduler with the command **stop-msg sched** and **start-msg sched**, or refresh the scheduler by running **refresh sched**.

- To disable a scheduled task:

```
msconfig set schedule.task:taskname.enable = 0
refresh sched
```

Scheduler Examples

Run **imexpire** at 12:30am, 8:30am, and 4:30pm:

```
msconfig set schedule.task:expire.crontab "30 0,8,16 * * * bin/imexpire"
```

Run **imsbackup** Monday through Friday at midnight (12AM):

```
msconfig set schedule.task:msbackup.crontab "0 0 * * 1-5 bin/imsbackup -f
backupfile /primary"
```

Pre-defined Automatic Tasks

At installation, Messaging Server creates, schedules and enables the following set of pre-defined automatic tasks:

The following automatic tasks are set and enabled for the message store:

```
schedule.task:expire.crontab = 0 23 * * * bin/imexpire
schedule.task:snapshot.crontab = 0 2 * * * bin/imdbverify -s -m
schedule.task:snapshotverify.crontab = 5,15,25,35,45,55 * * * * bin/imdbverify
```

The following automatic tasks are set and enabled for the MTA:

```
schedule.task:purge.crontab = 0 0,4,8,12,16,20 * * * bin/imsimta purge
schedule.task:return_job.crontab = 30 0 * * * lib/return_job
```

The following automatic task is set and enabled for the message store:

```
schedule.task:msprobe.crontab = 5,15,25,35,45,55 * * * * lib/msprobe
```

Configuring a Greeting Message

Messaging Server enables you to create an email greeting message to be sent to each new user.

To Create a New User Greeting

To create a new-user greeting:

```
msconfig set base.welcomemsg Message
```

Where *Message* must contain a header (with at least a subject line), followed by $$, then the message body. The $ represents a new line.

For example, to enable this option, you can set the following configuration variables:

```
msconfig set base.welcomemsg 'Subject: Welcome!! $$ example.com welcomes you to
the premier Internet experience in Dafandzadgad!'
```

Depending on the shell that you are using, it might be necessary to append a special character before $ to escape the special meaning of $. ($ is often the escape character for the shell.) Alternatively, you can do this within the **msconfig** prompt so that you do not need to the $. Simply run **msconfig**, then issue the **set***optionvalue* command.

To Set a Per-Domain Greeting Message

Whenever you create a new hosted domain, create per-domain greeting messages for your supported languages. If this is not done, the generic greeting message set by **base.welcomemsg** is sent.

You can set a greeting message for new users in each domain. The message can vary depending on the user's, the domain's, or the site's preferred language. This is done by setting the **mailDomainWelcomeMessage** attribute in the desired LDAP domain entry. The attribute syntax is as follows:

mailDomainWelcomeMessage;lang-*userprefLang*

mailDomainWelcomeMessage;lang-*domain_prefLang*

mailDomainWelcomeMessage;lang-*gen.sitelanguage*

The following example sets the domain welcome message for English:

mailDomainWelcomeMessage;lang-en: Subject: Welcome!! $$Welcome to the mail system.

The following example sets the domain welcome message for French:

mailDomainWelcomeMessage;lang-fr: Subject: Bienvenue!! $$Bienvenue a example.org!

Using these examples, assume the following:

- The domain is *example.org*.
- A new user belongs to this domain.
- The user's preferred language is French as specified by the LDAP attribute **preferredlanguage**.
- The **example.org** domain has the above English and French welcome messages available.
- The site language is **en** as specified by **gen.sitelanguage**.

For a list of supported locales and their language value tag, see *Directory Server Reference*.

When users log in for the first time, they receive the French greeting. If the French welcome message isn't available, they get the English greeting.

Greeting Message Theory of Operations

Greeting messages can be set by both the LDAP attribute **mailDomainWelcomeMessage**, the **base.welcomemsg** option, and the **message_language:***langcode***.welcomemsg** option.The **base.welcomemsg** option is the default, the **message_language:***langcode***.welcomemsg** option is language-code specific. The order in which a message is chosen, with the top one having the highest preference, is shown below:

mailDomainWelcomeMessage;lang-*user_prefLang*

mailDomainWelcomeMessage;lang-*domain_prefLang*

mailDomainWelcomeMessage;lang-*gen.sitelanguage*

mailDomainWelcomeMessage

base.welcomemsg;lang-"$user-*prefLang*"

base.welcomemsg;lang-"$domain-*prefLang*"

base.welcomemsg;lang-"$gen.*sitelanguage*"

base.welcomemsg

The algorithm works as follows: if there are no domains (or there are, but there is no per domain welcome message provisioned for them), a welcome message is configured with the **base.welcomemsg** option, if specified. If a user has a preferred language (set with the **preferredlanguage** LDAP attribute) and **base.welcomemsg;lang-***user_prefLang* is set, the user will receive that welcome message at the time of their first log in to the server. If **base.welcomemsg;lang-***gen.sitelanguage* is set, and **preferredlanguage** is not set, but the site language is set (using **base.sitelanguage** option), user will receive that message. If no language tag option is set and a untagged **base.welcomemsg** is set, then that message will be sent to the user. If none of the values are set, user will not receive any welcome message.

If the user is in a domain, then similar to the discussion above, the user might receive one of **mailDomainWelcomeMessage;lang-xx**, depending on which one is available in the list and in the order given.

Example: Domain is **example.org**. The domain preferred language is German (de). But the new user in this domain has preferred language of Turkish (tr). Site language is English. The following values are available (**mailDomainWelcomeMessage** are attributes of the domain example.org):

```
mailDomainWelcomeMessage;lang-fr
mailDomainWelcomeMessage;lang-ja
base.welcomemsg;lang-de
base.welcomemsg;lang-en
base.welcomemsg
```

According to the algorithm, the message sent to the user is **base.welcomemsg;lang-de**.

Setting a User-Preferred Language

Messaging Server enables you to set a user preferred language.

Overview of Setting a User-Preferred Language

You can set a preferred language for the GUI and server-generated messages by setting the attribute **preferredLanguage** in the user's LDAP entry.

When the server sends messages to users outside of the server's administrative domain it does not know what their preferred language is unless it is responding to an incoming message with a preferred language specified in the incoming message's header. The header fields (**Accept-Language**, **Preferred-Language** or **X-Accept-Language**) are set according to attributes specified in the user's mail client.

If there are multiple settings for the preferred language, the server chooses the preferred language. For example, if a user has a preferred language attribute stored in the Directory Server and also has a preferred language specified in their mail client, the server chooses the preferred language in the following order:

1. The **Accept-Language** header field of the original message.

2. The **Preferred-Language** header field of the original message.
3. The **X-Accept-Language** header field of the original message.
4. The preferred language attribute of the sender (if found in the LDAP directory).

To Set a Domain Preferred Language

A domain preferred language is a default language specified for a particular domain. For example, you can specify Spanish for a domain called **mexico.example.org**. Administrators can set a domain preferred language by setting the attribute **preferredLanguage** in the domain's LDAP entry.

To Specify a Site Language

You can specify a default site language for your server as follows. The site language is used to send language-specific versions of messages if no user preferred language is set.

Specify a site language as follows:

```
msconfig set base.sitelanguage value
```

where *value* is one of the local supported languages. See the Directory Server documentation for a list of supported locales and the language value tag.

Encryption Settings

This is described in enabling SSL and selecting ciphers in *Messaging Server Security Guide*, which also contains background information on all security and access-control topics for Messaging Server.

Setting a Failover LDAP Server

It is possible to specify more than one LDAP server for the user/group directory so that if one fails another takes over.

To set a failover LDAP server:

1. Set **base.ugldaphost** to the multiple replicated LDAP servers. For example:

   ```
   msconfig set base.ugldaphost "ldap1.example.com ldap2.example.com:389"
   ```

2. If you are using a compiled MTA configuration then recompile the MTA configuration file.

   ```
   imsimta cnbuild
   ```

3. Restart Messaging Server.

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

5

Configuring and Administering Multiplexor Services

This chapter describes the Messaging Multiplexor (MMP) for standard mail protocols (POP, IMAP, and SMTP).

Multiplexor Services in Unified Configuration Overview

The MMP configuration is stored in the Unified Configuration. Table 5–1 lists the MMP configuration files that are no longer used in Unified Configuration:

Table 5–1 Legacy MMP Configuration Files

File Type	Legacy File Names
POP SSL MMP Encryption File	**PopProxyAService.cfg**
POP Services Configuration Template	**PopProxyAService-def.cfg**
IMAP SSL MMP Encryption File	**ImapProxyAService.cfg**
IMAP Services Configuration Template	**ImapProxyAService-def.cfg**
Service Starting Configuration File	**AService.cfg**
Service Starting Configuration Template	**AService-def.cfg**
SMTP SSL MMP Encryption File	**SmtpProxyAService.cfg**
SMTP Services MMP Configuration Template	**SmtpProxyAService-def.cfg**

In Unified Configuration, you enable and modify the MMP configuration by running the **msconfig** command to set the appropriate MMP options. The **ServiceList** and **SSLports** options are gone in Unified Configuration. You now use the **imapproxy**, **popproxy**, and **smtpproxy** configuration groups, and the **tcp_listen** option. Use the following commands to view the initial MMP configuration settings.

```
msconfig
msconfig> show mmp*
role.mmp.enable = 0
msconfig> show imapproxy*
role.imapproxy.connlimits = :20
```

```
role.imapproxy.tcp_listen:imapproxy1.tcp_ports = 143
msconfig> show popproxy*
role.popproxy.connlimits = :20
role.popproxy.tcp_listen:popproxy1.tcp_ports = 110
msconfig> show submitproxy*
role.submitproxy.connlimits = :20
```

In addition, the **ssl_ports** option works the **like tcp_ports** option but enables SSL services (thus fixing the problem in legacy configuration, where an SSL proxy service had to be listed in both **ServiceList** and **SSLPorts** options).

The following examples commands show how to update the MMP configuration:

- To enable MMP: **msconfig set mmp.enable 1**
- To change an IMAP proxy option: **msconfig set imapproxy.***optionvalue*
- To change a POP proxy option: **msconfig set popproxy.***optionvalue*
- To change an SMTP proxy option: **msconfig set smtpproxy.***optionvalue*
- To set a certmap default option: **msconfig set base.certmap:default.***optionvalue*

See *Messaging Server Reference*, or option descriptions in the **msconfig** online help, for more information.

Multiplexor Services

A multiplexor is necessary to achieve horizontal scalability (the ability to support more users by adding more machines), because it provides a single domain name that can be used to connect indirectly to multiple mail stores. A multiplexor can also provide security benefits.

In Unified Configuration, MMP is no longer managed separately from Oracle Communications Messaging Server.

Multiplexor Benefits

Message stores on heavily used messaging servers can grow quite large. Spreading user mailboxes and user connections across multiple servers can therefore improve capacity and performance. In addition, it can be more cost-effective to use several small server machines than one large, high-capacity, multiprocessor machine.

If the size of your mail-server installation requires the use of multiple message stores, your organization can benefit in several ways from using the multiplexor. The indirect connection between users and their message stores, coupled with the ease of reconfiguration of user accounts among messaging servers allows for the following benefits:

- **Simplified User Management** Because all users connect to one server (or more, if you have separate multiplexor machines for POP, IMAP, SMTP, or web access), you can preconfigure email clients and distribute uniform login information to all users. This simplifies your administrative tasks and reduces the possibility of distributing erroneous login information.

  For especially high-load situations, you can run multiple multiplexor servers with identical configurations and manage connections to them by DNS round robin or by using a load-balancing system. Because the multiplexors use information stored in the LDAP directory to locate each user's Messaging Server, moving a user to a new server is simple for the system administrator and transparent to the user. The administrator can move a user's mailbox from one Messaging Server

host to another, and then update the user's entry in the LDAP directory. The user's mail address, mailbox access, and other client preferences need not change.

- **Improved Performance** If a message store grows prohibitively large for a single machine, you can balance the load by moving some of the message store to another machine.

  You can assign different classes of users to different machines. For example, you can choose to locate premium users on a larger and more powerful machine.

  The multiplexors perform some buffering so that slow client connections (through a modem, for example) do not slow down the Messaging Server.

- **Decreased Cost** Because you can efficiently manage multiple Messaging Server hosts with a multiplexor, you might be able to decrease overall costs by purchasing several small server machines that together cost less than one very large machine.

- **Better Scalability** With the multiplexors, your configuration can expand easily. You can incrementally add machines as your performance or storage-capacity needs grow, without replacing your existing investment.

- **Minimum User Downtime.** Using the multiplexors to spread a large user base over many small store machines isolates user downtime. When an individual server fails, only its users are affected.

- **Increased Security** You can use the server machine on which the multiplexor is installed as a firewall machine. By routing all client connections through this machine, you can restrict access to the internal message store machines by outside computers. The multiplexors support both unencrypted and encrypted communications with clients.

About Messaging Multiplexor

The Messaging Multiplexor (MMP) is a specialized messaging server that acts as a single point of connection to multiple back-end messaging servers. With Messaging Multiplexor, large-scale messaging service providers can distribute POP and IMAP user mailboxes across many machines to increase message store capacity. All users connect to the single multiplexor server, which redirects each connection to the appropriate messaging server.

If you provide electronic mail service to many users, you can install and configure the Messaging Multiplexor so that an entire array of Messaging Server hosts appear to your mail users to be a single host.

The Messaging Multiplexor is provided as part of Messaging Server. You can install the MMP at the same time you install Messaging Server or other Communications Suite servers, or you can install the MMP separately at a later time. The MMP supports the following items:

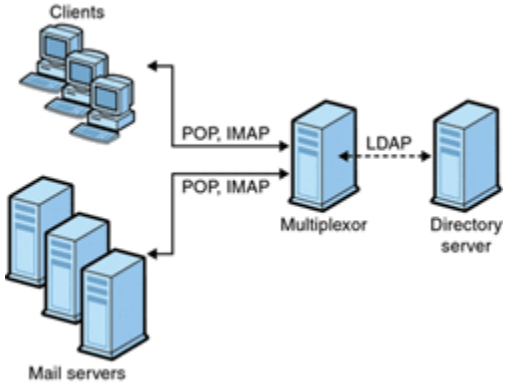
- Both unencrypted and encrypted (SSL) communications with mail clients.
- Client certificate-based authentication, described in "Certificate-Based Client Authentication".
- User pre-authentication, described in "User Pre-Authentication".
- Virtual domains that listen on different IP addresses and automatically append domain names to user IDs, described in "MMP Virtual Domains".
- Multiple installations of the MMP on different servers.

- Enhanced LDAP searching.
- POP before SMTP service for legacy POP clients.

How the Messaging Multiplexor Works

The MMP is a multithreaded server that facilitates distributing mail users across multiple server machines. The MMP handles incoming client connections destined for other server machines (the machines on which user mailboxes reside). Clients connect to the MMP itself, which determines the correct server for the users, connects to that server, and then passes data between the client and server. This capability allows Internet service providers and other large installations to spread message stores across multiple machines (to increase capacity) while providing the appearance of a single mail host for users (to increase efficiency) and for external clients (to increase security). Figure 5–1 shows clients and servers in an MMP installation.

Figure 5–1 Clients and Servers in an MMP Installation



All POP, IMAP, and SMTP clients work with the Messaging Multiplexor. The MMP accepts connections, performs LDAP directory lookups, and routes the connections appropriately. As is typical with other mail server installations, each user is assigned a specific address and mailbox on a specific Messaging Server. However, all connections are routed through the MMP.

In more detail, these are the steps involved in establishing a user connection:

1. A user's client connects to the MMP, which accepts preliminary authentication information (user name).
2. The MMP queries the Directory Server to determine which Messaging Server contains that user's mailbox.
3. The MMP connects to the proper Messaging Server, replays authentication, then acts as a pass-through pipe for the duration of the connection.

Encryption (SSL) Option

Messaging Multiplexor supports both unencrypted and encrypted (SSL) communications between the Messaging Server(s) and their mail clients. The current version of Messaging Server supports the new certificate database format (**cert9.db**).

When SSL is enabled, the MMP IMAP supports both STARTTLS on the standard IMAP port and IMAP+SSL on port 993. The MMP can also be configured to listen on port 995 for POP+SSL.

In legacy configuration, to enable SSL encryption for your IMAP, POP, and SMTP services, you would uncomment the appropriate SSL settings from the **.cfg** files. In Unified Configuration, you use the **msconfig** command to set the appropriate options. You must also set the list of all IMAP, POP, and SMTP server ports regardless of whether or not they are secure. See "Configuring MMP with SSL or Client Certificate-Based Login" for details.

By default, SSL is not enabled. To enable SSL, you must install an SSL server certificate. Then, you should use the **msconfig** command to set the SSL options. For a list of the SSL options, see the **ssl*** options in the **msconfig** online help.

Certificate-Based Client Authentication

In Unified Configuration, the certificate mapping file (**certmap.conf**), which matches a client's certificate to the correct user in the Users/Groups Directory Server, is no longer used. Instead, you use the following **msconfig** command to set the appropriate options:

```
msconfig set base.certmap:default.option value
```

The **default** can be replaced with the certificate **issuerDN** to have configuration specific to that certificate. That replaces the other groups in the **certmap.conf** file.

To use certificate-based client authentication, you must also enable SSL encryption. See "Encryption (SSL) Option" for more information.

You also have to configure a store administrator. You can use the mail administrator, but it is recommended that you create a unique user ID, such as **mmpstore** for this purpose so that you can set permissions as needed.

In Unified Configuration, the MMP supports the **dncomps** and **filtercomps** options. The values of these two options has the form *fromattr=toattr*. A *fromattr* value in a certificate's subjectDN can be used to form an LDAP query with the *toattr=value* element. For example, a certificate with a subjectDN of "**cn=Pilar Lorca, ou=pilar, o=example.com**" could be mapped to an LDAP query of "**(uid=pilar)**" with the line:

```
msconfig> set base.certmap:default.filtercomps ou=uid
```

To Enable Certificate-based Authentication for Your IMAP or POP Service

1. Decide on the user ID you intend to use as store administrator. While you can use the mail administrator for this purpose, it is recommended that you create a unique user ID for store administrator (for example, **mmpstore**).
2. Make sure that SSL encryption is (or will be) enabled. See "Encryption (SSL) Option" for more information.
3. Configure the MMP to use certificate-based client authentication by specifying default **certmap** option in your configuration. For example:

   ```
   msconfig set base.certmap:default.dncomps ""
   ```

4. Install at least one trusted CA certificate, as described in the discussion on installing certificates of trusted CAs in *Messaging Server Security Guide*.

User Pre-Authentication

The MMP provides you with the option of pre-authenticating users by binding to the directory as the incoming user and logging the result.

Note: Enabling user pre-authentication reduces server performance.

The log entries are in the format:*datetime*(**sid 0x***hex*) user *name***pre-authenticated - clientIP***address*, **server IP***address*

Where *date* is in the format **yyyymmdd**, *time* is in the time configured on the server in the format **hhmmss**, *hex* is the session identifier (**sid**) represented as a hexidecimal number, the **user***name* includes the virtual domain (if any), and the IP address is in dot-quad format.

MMP Virtual Domains

An MMP *virtual domain* is a set of configuration settings associated with one or more server IP addresses. The primary use of this feature is to provide different default domains for each server IP address. The **hosteddomains** option defaults to 1 (enabled).

A user can authenticate to the MMP with either a short-form userID or a fully qualified userID in the form **user@domain**. When a short-form userID is supplied, the MMP will append the **defaultdomain** setting, if specified. Consequently, a site which supports multiple hosted domains can permit the use of short-form user IDs simply by associating a server IP address and MMP virtual domain with each hosted domain.

To configure a virtual domain option, use the following command:

```
msconfig set vdomain:IP_address.option value
```

For example, to set the default domain, use the following command:

```
msconfig set vdomain:192.0.2.0.defaultdomain example.com
```

When set, virtual domain configuration option values override global configuration option values.

You can specify the following configuration options for a virtual domain:

```
authcachettl
authenticationldapattributes
authservice
authservicettl
binddn
bindpass
clientlookup
crams
debugkeys
defaultdomain
domainsearchformat
ehlokeywords
failovertimeout
hosteddomains
ldapcachesize
ldapcachettl
mailhostattrs
popbeforesmtpkludgechannel
preauth
replayformat
restrictplainpasswords
searchformat
smtpproxypassword
smtprelays
ssladjustciphersuites
```

```
sslnicknames
storeadmin
storeadminpass
tcpaccess
tcpaccessattr
virtualdomaindelim
```

For more information on these configuration options, see the **msconfig** online help or *Messaging Server Reference*.

> **Tip:** View the reference information for these configuration options at:
>
> `http://msg.wikidoc.info/index.php/MMP_Reference#optionname`
>
> For example, to view the description for the **tcpaccess** option, use the following URL:
>
> `http://msg.wikidoc.info/index.php/MMP_Reference#tcpaccess`

About SMTP Proxy

The MMP includes an SMTP submission proxy, which is disabled by default. Most sites do not need the SMTP proxy because Internet Mail standards already provide an adequate mechanism for horizontal scalability of SMTP (DNS MX records).

The SMTP proxy is useful for the security features it provides. First, the SMTP proxy is integrated with the POP proxy to implement the POP before SMTP authorization facility required by some legacy POP clients. For more information, see the discussion on using the MMP SMTP proxy in *Messaging Server Installation and Configuration Guide*. In addition, an investment in SSL acceleration hardware can be maximized by using the SMTP proxy.

Setting Up the Messaging Multiplexor

During the initial runtime configuration of Messaging Server, you determined if you wanted to configure the MMP on a machine. You could either set it up on the same machine as your Messaging Server or set it up on a separate machine.

> **Note:** MMP does not cache DNS results. A high quality caching DNS server on the local network is a requirement for a production deployment of Messaging Server.

Before You Configure MMP

Before configuring the MMP:

1. Choose the machine on which you will configure the MMP. It is best to use a dedicated machine for the MMP.

> **Note:** It is recommended that the MMP not be enabled on a machine that is also running either the POP or IMAP servers. If you install MMP on the same machine as Messaging Server, you must make sure that the POP and IMAP servers are set to nonstandard ports. That way, the MMP and Messaging Server ports do not conflict with one another.

2. On the machine where the MMP is to be configured, create a UNIX system user to be used by the MMP. This new user must belong to a UNIX system group. See the discussion on creating UNIX system users and groups in *Messaging Server Installation and Configuration Guide*.
3. Set up the Directory Server and its host machine for use with Messaging Server, if they are not already set up. See the discussion on preparing directory server for Messaging Server configuration in *Messaging Server Installation and Configuration Guide*.
4. When upgrading an MMP or back-end IMAP server, the MMP's **capability** option should be set to only include capabilities present on all back-end IMAP servers. See the discussion on the **capability** option in *Messaging Server Reference* for more information.

Multiplexor Configuration

To configure the MMP, you must use the Messaging Server **init-config** program, which gives you the option of enabling the Messaging Multiplexor. For detailed information about the **init-config** program, see *Messaging Server Reference Guide*.

To Configure the MMP

1. Install Messaging Server software on the machine where you are installing and configuring the MMP.
2. Configure the MMP by creating the Messaging Server Initial Runtime Configuration. See the discussion about creating the initial Messaging Server runtime configuration in *Messaging Server Installation and Configuration Guide*.

Multiplexor Configuration Options

You control how the MMP operates by specifying various configuration options in the Unified Configuration. See *Messaging Server Reference* for more information.

Starting the Multiplexor

To start, stop, or refresh an instance of the Messaging Multiplexor, use one of the commands in Table 5–2. These commands are located in the *MessagingServer_home*/**bin** directory.

Table 5–2 MMP Commands

Option	Description
start-msg mmp	Starts the MMP (only if the MMP is enabled and one is not already running).
stop-msg mmp	Stops the most recently started MMP.
refresh mmp	Causes an MMP that is already running to refresh its configuration without disrupting any active connections.

Modifying an Existing MMP

1. To modify an existing instance of the MMP, use the **msconfig** command to edit the configuration as necessary.
2. The run either **refresh mmp** or **stop-msg mmp**; **start-msg mmp**.

Use the former only if you changed "refreshable" options and the latter if you changed any "non-refreshable" options.

Configuring MMP with SSL or Client Certificate-Based Login

This section describes how to configure MMP with SSL or client certificate-based login.

Note: It is assumed that the MMP is installed on a machine that does not have a Message Store or MTA.

To Configure MMP with SSL

1. Generate and install the certificate by using the **certutil** command. See the discussion on certificate based authentication for Messaging Server in *Messaging Server Security Guide* for details.
2. Set the password used for the certificate file. For example:

```
msconfig
msconfig> set "sectoken:Internal (Software) Token.tokenpass" newpassword
msconfig> write
```

   The default setting for this password was provided during initial configuration, but it might be different. It must match the password that was used when the certificate db was created by running the **certutil -N** command.
3. Set either **sslenable** on the relevant proxy (for STARTTLS) and/or set the **ssl_ports** on a **tcp_listen** for the appropriate proxy. In general, the default settings cover the remainder of the configuration and you do not need to be changed.
4. Start the MMP:

```
MessagingServer_home/bin/start-msg
```

5. If you do not want to use SSL between the MMP and the back-end server, then set the **sslbacksideport** option to 0 for **imapproxy** and **popproxy** as appropriate.

To Configure MMP with Client Certificate-based Login

To configure client certificate based login:

1. Get a copy of a client certificate and the CA certificate which signed it.
2. Import the CA certificate as a Trusted Certificate Authority (see the discussion on obtaining and managing certificates in *Messaging Server Security Guide*).
3. Use the Store Administrator you created during your Messaging Server installation. See "Specifying Administrator Access to the Message Store" for more information.
4. Create a **certmap.conf** file for the MMP. For example:

```
msconfig> set base.certmap:default.dncomps ""
msconfig# set base.certmap:default.filtercomps "e=mail"
```

   This means to search for a match with the **e** field in the certificate DN by looking at the mail attribute in the LDAP server.

5. Use the **msconfig** command to update the configuration with the following options:
   a. Set **storeadmin** and **storeadminpass** to values from Step 3.
   b. Set **usergroupdn** to the root of your Users and Groups tree.
6. If you want client certificates with POP3, repeat Step 5 for the **popproxy** group.
7. If the MMP is not already running, start it with the following command in the *MessagingServer_home*/**bin** directory:

   start-msg mmp
8. Import the client certificate into your client. In Netscape Communicator, click the padlock (Security) icon, then select Yours under Certificates, then select Import a Certificate and follow the instructions.

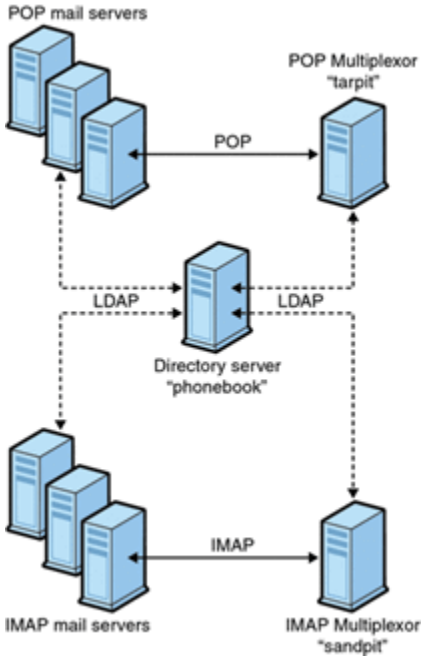
Note: All your users have to perform this step if you want to use client certificates everywhere.

A Sample Topology

The fictional Example Corporation has two Messaging Multiplexors on separate machines, each supporting several Messaging Servers. POP and IMAP user mailboxes are split across the Messaging Server machines, with each server dedicated exclusively to POP or exclusively to IMAP. (You can restrict client access to POP services alone by removing the **imapproxy** entry from the MMP configuration. Likewise, you can restrict client access to IMAP services alone by removing the **popproxy** entry from the MMP configuration). Each Messaging Multiplexor also supports only POP or only IMAP. The LDAP directory service is on a separate, dedicated machine.

Figure 5–2 illustrates this topology.

Figure 5–2 Multiple MMPs Supporting Multiple Messaging Servers



MMP Tasks

MMP configuration tasks include:

- To Configure Mail Access with MMP
- To Set a Failover MMP LDAP Server

To Configure Mail Access with MMP

The MMP does not use the **PORT_ACCESS** mapping table. If you want to reject SMTP connections from certain IP addresses and you are using the MMP, you must use the **tcpaccessattr** option.

To Set a Failover MMP LDAP Server

It is possible to specify more than one LDAP server for the MMP so that if one fails another takes over. Modify the **ugldaphost** option. For example:

```
msconfig set ugldaphost "ldap1.example.com ldap2.example.com"
```

Note: Make sure there is a space between the host names in the preceding configuration, and because of that space, to enclose the hosts in quotation marks.

6

MTA Concepts

This chapter provides a conceptual description of the Oracle Communications Messaging Server MTA.

The MTA Functionality

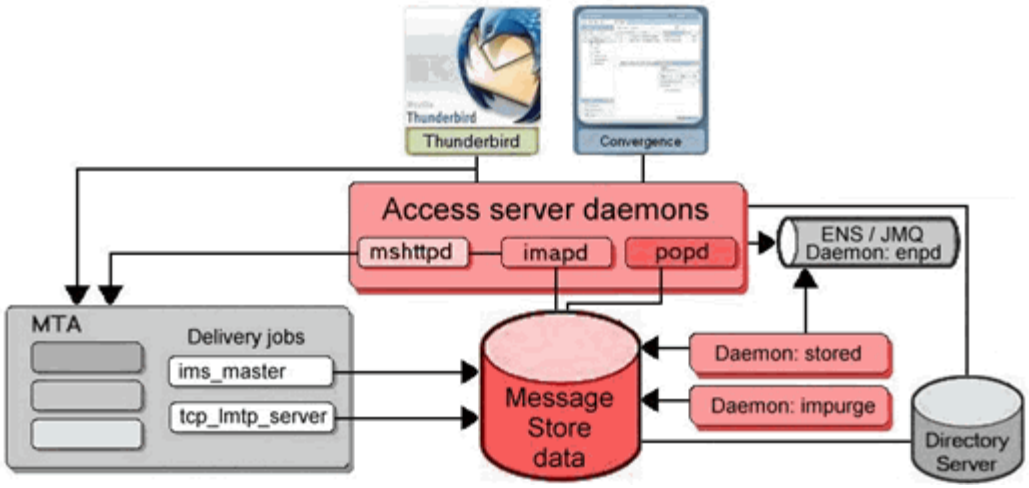
The Message Transfer Agent (MTA) is a component of the Messaging Server. At its most basic level, the MTA is a message router. It accepts messages from other servers, reads the address, and routes it to the next server on way to its final destination, typically a user's mailbox.

Over the years, a lot of functionality has been added to the MTA, and with it, size, power, and complexity. These MTA functions overlap, but, in general, can be classified as follows:

- **Routing.** Accepts a message, expands or transforms it if necessary (for example if it is an alias), and routes it to the next server, channel, program, file, or whatever. The routing function has been expanded to allow administrator specification of the internal and external mechanics of how messages are routed. For example, it is possible to specify things such as SMTP authentication, use of various SMTP commands and protocol, TCP/IP or DNS lookup support, job submission, process control and message queueing and so on.
- **Address Rewriting.** Envelope addresses are often rewritten as part of the routing process, but envelope or header addresses can also be rewritten to a more desired or appropriate form.
- **Filtering.** The MTA can filter messages based on address, domain, possible virus or spam content, size, IP address, header content, and so on. Filtered messages can be discarded, rejected, modified, sent to a file, sent to a program, or be sent to the next server on its way to a user mailbox.
- **Content Modification.** Message headers or content can be modified. Example: making a message readable to a specific client or in a specific character set or checking for spam or viruses.
- **Auditing.** Tracking who submitted what, where and when.

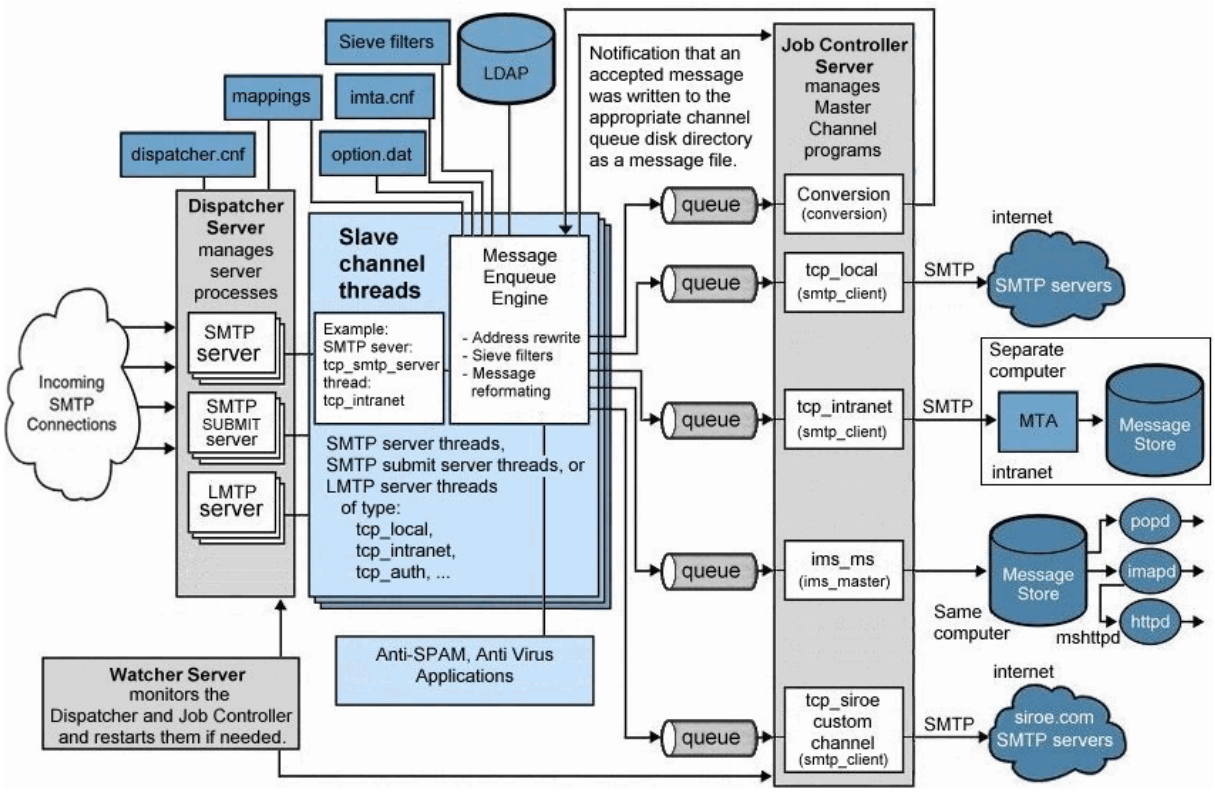
Figure 6–1, "High Level Message Store and MTA Architecture" shows the high-level message store and MTA architecture.

Figure 6–1 High Level Message Store and MTA Architecture



Several subcomponents and processes support these functions and are shown in Figure 6–2.

Figure 6–2 MTA Architecture



The information in this chapter describes these subcomponents and processes. In addition, a number of tools enable you to manage and configure these functions. These include Unified Configuration options, mapping tables, channel options, channels, and rewrite rules. These tools are described in the following MTA topics:

- Channel Configuration in *Messaging Server Reference*
- Integrating Spam and Virus Filtering Programs in *Messaging Server Reference*

- LMTP Delivery
- Managing Logging
- Monitoring Messaging Server
- Security and Access Control in *Messaging Server Security Guide*
- Troubleshooting the MTA
- Vacation Automatic Message Reply

MTA Architecture and Message Flow Overview

This section provides a short overview of MTA architecture and message flow (see Figure 6–2, "MTA Architecture"). The MTA is a highly complex component and this figure is a *simplified* depiction of messages flowing through the system. In fact, this picture is not a perfectly accurate depiction of all messages flowing through the system. For purposes of conceptual discussion, however, it must suffice.

Dispatcher and SMTP Server (Slave Program)

Messages enter the MTA from the Internet or intranet via SMTP sessions. When the MTA receives a request for an SMTP connection, the MTA *dispatcher* (a multithreaded connection dispatching agent), executes a *slave* program (**tcp_smtp_server**) to handle the SMTP session. The dispatcher maintains pools of multithreaded processes for each service. As additional sessions are requested, the dispatcher activates an SMTP server program to handle each session. A process in the Dispatcher's process pool may concurrently handle many connections. Together the dispatcher and slave program perform a number of different functions on each incoming message. Three primary functions are:

- **Message blocking.** Messages from specified IP addresses, mail addresses, ports, channels, header strings and so on, may be blocked.
- **Address changing.** Incoming **From:** or **To:** addresses may be rewritten to a different form.
- **Channel enqueueing.** Addresses are run through the rewrite rules to determine which channel the message should be sent.

See "The Dispatcher" for more information.

Routing and Address Rewriting

SMTP servers enqueue messages, but so can a number of other channels including, the conversion channel and reprocess channel. Many tasks are achieved during this phase of delivery, but the primary tasks are:

- Alias expansion.
- Running the addresses through the rewrite rules which do two things:
  - Rewrite the domain part of addresses into a desired format.
  - Direct messages to the appropriate channel queue.

Channel

The channel is the fundamental MTA component used for message processing. A channel represents a message connection with another system (for example, another MTA, another channel, or the local message store). As mail comes in, different messages require different routing and processing depending on the message's source

and destination. For example, mail to be delivered to a local message store is processed differently from mail to be delivered to the Internet, which is processed differently from mail to be sent to another MTA within the mail system. Channels provide the mechanism for customizing the processing and routing required for each connection. In a default installation, the majority of messages go to a channels handling Internet, intranet, and local messages.

Specialized channels for specific situations can also be created. For example, suppose that a certain Internet domain processes mail very slowly causing mail addressed to this domain to clog up the MTA. A special channel could be created to provide special handling for messages addressed to the slow domain, thus relieving the system of this domain bottleneck.

The domain part of the address determines to what channel the message is enqueued. The mechanism for reading the domain and determining the appropriate channel is called the rewrite rules (see "Rewrite Rules").

Channels typically consist of a channel queue and a channel processing program called a *master program*. After the slave program delivers the message to the appropriate channel queue, the master program performs the desired processing and routing. Here is an example of a channel entry:

```
tcp_intranet smtp mx single_sys subdirs 20 noreverse maxjobs 7 SMTP_POOL
maytlsserver allowswitchchannel saslswitchchannel tcp_auth
tcp_intranet-daemon
```

The first word, in this case **tcp_intranet** is the channel name. The last word is called the channel tag. The words in between are called channel options (formerly called channel keywords) and specify how messages are to be processed. Hundreds of different options enable messages to be processed in many ways. See the discussion about channel options in *Messaging Server Reference* for a complete description of channel options.

Message Delivery

After the message is processed, the master program sends the message to the next stop along the message's delivery path. This may be the intended recipient's mailbox, another MTA, or even a different channel. Forwarding to another channel is not shown in the picture, but is a common occurrence.

The Dispatcher

The Dispatcher is a multithreaded dispatching agent that permits multiple multithreaded server processes to share responsibility for SMTP connection services. When using the Dispatcher, it is possible to have several multithreaded SMTP server processes running concurrently, all handling connections to the same port. In addition, each server may have one or more active connections.

The Dispatcher acts as a central receiver for the TCP ports listed in its configuration. For each defined service, the Dispatcher may create one or more SMTP server processes to handle the connections after they are established.

In general, when the Dispatcher receives a connection for a defined TCP port, it checks its pool of available worker processes for the service on that port and chooses the best candidate for the new connection. If no suitable candidate is available and the configuration permits it, the Dispatcher may create a new worker process to handle this and subsequent connections. The Dispatcher may also create a new worker process in expectation of future incoming connections. There are several configuration options which may be used to tune the Dispatcher's control of its various services, and

in particular, to control the number of worker processes and the number of connections each worker process handles.

For more information, see the discussion about the dispatcher configuration file in *Messaging Server Reference*.

Creation and Expiration of Server Processes

Automatic housekeeping facilities within the Dispatcher control the creation of new and expiration of old or idle server processes. The basic options that control the Dispatcher's behavior are **service:***name***.min_procs** and **service:***name***.max_procs**. The **service:***name***.min_procs** option provides a guaranteed level of service by having a number of server processes ready and waiting for incoming connections. The **service:***name***.max_procs** option, on the other hand, sets an upper limit on how many server processes may be concurrently active for the given service.

It is possible that a currently running server process might not be able to accept any connections because it is already handling the maximum number of connections of which it is capable, or because the process has been scheduled for termination. The Dispatcher may create additional processes to assist with future connections.

The **min_conns** and **max_conns** options provide a mechanism to help you distribute the connections among your server processes. The **min_conns** option specifies the number of connections that flags a server process as "busy enough," while the **max_conns** option specifies the "busiest" that a server process can be.

In general, the Dispatcher creates a new server process when the current number of server processes is less than the value of the **service:***name***.min_procs** option or when all existing server processes are "busy enough" (the number of currently active connections each has is at least **min_conns**).

If a server process is killed unexpectedly, for example, by the UNIX system **kill** command, the Dispatcher still creates new server processes as new connections come in.

For information about configuring the Dispatcher, see the discussion about the dispatcher configuration file in *Messaging Server Reference*.

To Start and Stop the Dispatcher

To start the Dispatcher, run the following command:

```
start-msg dispatcher
```

This command subsumes and makes obsolete any other *start-msg* command that was used previously to start up a component of the MTA that the Dispatcher has been configured to manage. Specifically, you should no longer use **imsimta start smtp**. An attempt to execute any of the obsoleted commands causes the MTA to issue a warning.

To shut down the Dispatcher, run the following command:

```
stop-msg dispatcher
```

What happens with the server processes when the Dispatcher is shut down depends upon the underlying TCP/IP package. If you modify your MTA configuration or options that apply to the Dispatcher, you must restart the Dispatcher so that the new configuration or options take effect.

To restart the Dispatcher, run the following command:

```
imsimta restart dispatcher
```

Restarting the Dispatcher has the effect of shutting down the currently running Dispatcher, then immediately starting a new one.

MTA Configuration Overview

In a legacy configuration, you manage the MTA configuration by editing various text files. In Unified Configuration, you manage the MTA configuration by using the **msconfig** command. The Unified Configuration stores the MTA configuration in a single file, **config.xml** (for the most part). The **msconfig** command performs syntax validation on option names and values as well as the Unified Configuration structure to a limited degree. See *Messaging Server Reference Guide* for more information about the **msconfig** command syntax and options.

Caution: Only edit your Unified Configuration by running the **msconfig** command. This saves old configurations and allows for rollback. Do not hand-edit any of the Unified Configuration files. Oracle Support may occasionally edit these files to work around any issues pertaining to **msconfig**.

Table 6–1 provides a comparison of how the MTA configuration is managed in legacy and Unified Configuration.

Table 6–1 MTA Configuration: Legacy Versus Unified Configuration

Legacy Configuration Files	Unified Configuration Method
mappings file	Running **msconfig edit mappings** loads mappings in legacy format in the administrator's chosen editor.
imta.cnf file	Running **msconfig edit channels** loads channel blocks in legacy format in the administrator's chosen editor. Running **msconfig edit rewrite** loads rewrite rules in legacy format in the administrator's chosen editor.
option.dat file	Get or set options directly by running **msconfig**.
job_controller.cnf file	Get or set options directly by running **msconfig**; most options require **job_controller** prefix.
dispatcher.cnf file	Get or set dispatcher options; most require a **dispatcher.** prefix, service-specific settings are in a **service:***name* group, such as **service:SMTP.tcp_ports**.
conversions file	Run **msconfig edit conversions**.
aliases file	Run **msconfig edit aliases**.

For more information on MTA services see *Messaging Server Reference Guide*.

Rewrite Rules

Rewrite rules determine the following:

- How to rewrite the domain part of an address into its proper or desired format.
- To which channel the message should be enqueued after the address is rewritten.

Each rewrite rule consists of a *pattern* and a *template*. The pattern is a string to match against the domain part of an address. The template specifies the actions to take if the domain part matches the pattern. It consists of two things:

1. A set of instructions (that is, a string of control characters) specifying how the address should be rewritten.
2. The name of the channel to which the message shall be sent. After the address is rewritten, the message is enqueued to the destination channel for delivery to the intended recipient.

Here is an example of a rewrite rule:

```
example.org $U%$D@tcp_example-daemon
```

example.org is the domain pattern. Any message with the address containing **example.org** will be rewritten as per the template instructions (**$U%$D**). **$U** specifies that the rewritten address use the same user name. **%** specifies that the rewritten address use the same domain separator. **$D** specifies that the rewritten address use the same domain name that was matched in the pattern. **@tcp_example-daemon** specifies that the message with its rewritten address be sent to the channel called **tcp_example-daemon**.

For more information about configuring rewrite rules, see the discussion about the MTA configuration file in *Messaging Server Reference*.

Channels

The channel is the fundamental MTA component that processes a message. A channel represents a connection with another computer system or group of systems. The actual hardware connection or software transport or both may vary widely from one channel to the next.

Channels perform a variety of functions, including:

- Transmitting messages to remote systems, deleting them from their queue after they are sent
- Accepting messages from remote systems, placing them in the appropriate channel queues
- Delivering messages to the local message store
- Delivering messages to programs for special processing

Messages are enqueued by channels on the way into the MTA and dequeued on the way out. Typically, a message enters by one channel and leaves by another. A channel might dequeue a message, process the message, or enqueue the message to another MTA channel.

Master and Slave Programs

Generally (but not always), a channel is associated with two programs: master and slave. The slave program accepts messages from another system and adds them to a channel's message queue. The master program transfers messages from the channel to another system.

For example, an SMTP channel has a master program that transmits messages and a slave program that receives messages. These are, respectively, the SMTP client and server.

The master channel program is typically responsible for outgoing connections where the MTA has initiated the operation. The master channel program:

- Runs in response to a local request for processing.
- Dequeues the message from the channel message queue.
- If the destination format is not the same format as the queued message, performs conversion of addresses, headers, and content, as necessary.
- Initiates network transport of the message.

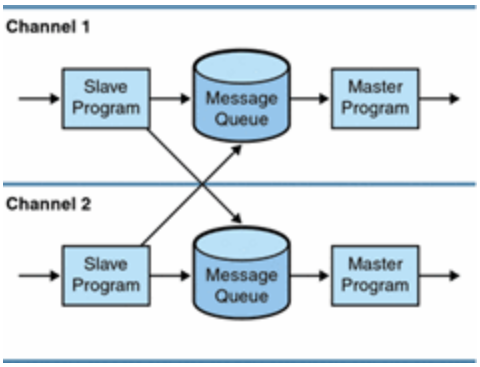
The slave channel program typically accepts incoming connections where the MTA is responding to an external request. The slave channel program:

- Runs in response to an external event or upon local demand.
- Enqueues a message to a channel. The target channel is determined by passing envelope addresses through a rewrite rule.

For example, Figure 6–3 shows two channel programs, Channel 1 and Channel 2. The slave program in Channel 1 receives a message from a remote system. It looks at the address, applies rewrite rules as necessary, then based on the rewritten address enqueues the message to the appropriate channel message queue.

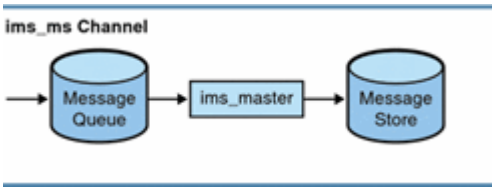
The master program dequeues the message from the queue and initiates network transport of the message. Note that the master program can only dequeue messages from its own channel queue.

Figure 6–3 Master and Slave Program Interaction



Although a typical channel has both a master and a slave program, it is possible for a channel to contain only a slave program *or* a master program. For example, the **ims-ms** channel supplied with Messaging Server contains only a master program because this channel is responsible only for dequeuing messages to the local message store, as shown in Figure 6–4.

Figure 6–4 ims-ms Channel



Channel Message Queues

All channels have an associated message queue. When a message enters the messaging system, a slave program determines to which message queue the message is enqueued. The enqueued messages are stored in message files in the channel queue directories. By default, these directories are stored at the following location: *DataRoot*/**queue/***channel*/*****. For more information, see the discussion about message queue sizing in *Messaging Server Installation and Configuration Guide*.

Caution: Do not add any files or directories in the MTA queue directory. When using a separate file system for the MTA queue directories, create a subdirectory under that mount point and specify that subdirectory in the **SERVERROOT** environment variable. Sites may change it by using either a symbolic link or using it as a file system mount point. The default value is: **IMTA_ROOT:data/queue/**.

Channel Definitions

In Unified Configuration, use the **msconfig edit channels** and **msconfig edit rewrites** commands to view and edit the channel block. (See *Messaging Server Reference* for information about setting up channels.)

A channel definition contains the name of the channel followed by an optional list of keywords that define the configuration of the channel, and a unique channel tag, which is used in rewrite rules to route messages to the channel. Channel definitions are separated by single blank lines. Comments, but no blank lines, may appear inside a channel definition. The following represents the channel format.

```
[blank line]
! sample channel definition
<Channel_Name> <keyword1> <keyword2>
<Channel_Tag>
[blank line]
```

Collectively, the channel definitions are referred to as the channel host table. An individual channel definition is called a channel block. In the following example, the channel host table contains three channel definitions or blocks.

```
! test.cnf - An example configuration file.
!
! Rewrite Rules
.
.
.

! BEGIN CHANNEL DEFINITIONS
! FIRST CHANNEL BLOCK
l
local-host

! SECOND CHANNEL BLOCK
a_channel defragment charset7 usascii
a-daemon

! THIRD CHANNEL BLOCK
b_channel noreverse notices 1 2 3
b-daemon
```

A typical channel entry looks something like the following, shown in legacy format as would display using the **msconfig edit channels** command:

```
tcp_intranet smtp mx single_sys subdirs 20 noreverse maxjobs 7 SMTP_POOL \
maytlsserver allowswitchchannel saslswitchchannel tcp_auth
tcp_intranet-daemon
```

The first word, in this case **tcp_intranet**, is the channel name. The last word, in this case **tcp_intranet-daemon**, is called the *channel tag*. The channel tag is the name used by rewrite rules to direct messages. The words in between the channel name and channel tag are called channel options (formerly channel keywords) and specify how the message is to be processed. Hundreds of different keywords allow messages to processed in many ways. A complete listing of channel keywords is provided in *Messaging Server Reference*.

The channel host table defines the channels Messaging Server can use and the names of the systems associated with each channel.

On UNIX systems, the first channel block in the file always describes the local channel, **l**. (An exception is a **defaults** channel, which can appear before the local channel.) The local channel is used to make routing decisions and for sending mail sent by UNIX mail tools.

You can also set global options for channels or set options for a specific channel. For more information on the option files, the TCP/IP (SMTP) channel option files, and configuring channels, see *Messaging Server Reference*.

The MTA Directory Information

For each message that it processes, the MTA needs to access directory information about the users, groups, and domains that it supports. This information is stored in an LDAP directory service. The MTA directly accesses the LDAP directory.

The Job Controller

Each time a message is enqueued to a channel, the Job Controller ensures that there is a job running to deliver the message. This might involve starting a new job process, adding a thread, or simply noting that a job is already running. If a job cannot be started because the job limit for the channel or pool has been reached, the Job Controller waits until another job has exited. When the job limit is no longer exceeded, the Job Controller starts another job.

Channel jobs run inside processing pools within the Job Controller. A pool can be thought of a "place" where the channel jobs are run. The pool provides a computing area where a set of jobs can operate without vying for resources with jobs outside of the pool. For more information on pools, see the discussion about processing pools for channel execution jobs in *Messaging Server Reference*.

Job limits for the channel are determined by the **channel:***name***.maxjobs** option. Job limits for the pool are determined by the **job_controller.job_pool:***name***.job_limit** option for the pool.

Messaging Server normally attempts to deliver all messages immediately. If a message cannot be delivered on the first attempt, however, the message is delayed for a period of time determined by the appropriate **backoff** option. As soon as the time specified in the **backoff** option has elapsed, the delayed message is available for delivery, and if necessary, a channel job is started to process the message.

The Job Controller's in-memory data structure of messages currently being processed and awaiting processing typically reflects the full set of message files stored on disk in the MTA queue area. However, if a backlog of message files on disk builds up enough to exceed the Job Controller's in-memory data structure size limit, then the Job Controller tracks in memory only a subset of the total number of messages files on disk. The Job Controller processes only those messages it is tracking in memory. After a sufficient number of messages have been delivered to free enough in-memory storage, the Job Controller automatically refreshes its in-memory store by scanning the MTA queue area to update its list of messages. The Job Controller then begins processing the additional message files it just retrieved from disk. The Job Controller performs these scans of the MTA queue area automatically.

In previous versions of Messaging Server, the Job Controller read all the files in the queue directory in the order in which they are found. It now reads several channel queue directories at once. This makes for much more reasonable behavior on startup, restart, and after **max_cache_messages** has been exceeded. The number of directories to be read at once is controlled by the Job Controller option **rebuild_parallel_channels**. This can take any value between 1 and 100. The default is 12.

If your site routinely experiences heavy message backlogs, you might want to tune the Job Controller by using the **max_cache_messages** option. By increasing the **max_cache_messages** option value to allow Job Controller to use more memory, you can reduce the number of occasions when message backlogs overflow the Job Controller's in-memory cache. This reduces the overhead involved when the Job Controller must scan the MTA queue directory. Keep in mind, however, that when the Job Controller does need to rebuild the in-memory cache, the process will take longer because the cache is larger. Note also that because the Job Controller must scan the MTA queue directory every time it is started or restarted, large message backlogs mean that starts or restarts of the Job Controller will incur more overhead than starts or restarts when no such backlog exists.

You do not want to overwhelm the job controller by keeping information about huge numbers of messages in memory. For this reason, there has to be a and upper and lower limit. The number specified by **max_cache_messages** is the number of messages that the job controller will hold in memory. It will get this high if there are new messages delivered, for instance ones received by **tcp_smtp_server**. Beyond this number, messages are queued (put on disk), but not put into the job controller memory structure. The job controller notices this condition and when the number of messages in memory drops below half this maximum, it starts scanning the disk queues for more messages. It always looks for untried messages "ZZ..." files first, then previously tried messages.

In addition, the job controller limits the number of messages reclaimed from disk. It only reads from disk up to three-quarters of the **max_cache_messages** to allow for headroom for new messages (if messages are being reclaimed from disk, they have been delayed, which is an undesirable state).

Furthermore, you want to avoid cluttering up the memory structure with delayed messages (those that cannot be processed yet). When a message is delayed because it cannot be delivered immediately (a delivery attempt has failed if the number of messages the job controller knows about is greater than 5/8 of **max_cache_messages** and the number of delayed messages is greater than 3/8 of **max_cache_messages**) the message is forgotten until the next sweep of the on disk structures, which will be when the number of messages drops below 1/2 **max_cache_messages**.

The only obvious problems with having **max_cache_messages** too small is that the scheduling of jobs will become suboptimal. The scanning of the disk queues is also a bit simplistic. If you have huge numbers of messages backlogged in both the **tcp_local**

and **ims_ms** queues, then the rebuild thread finds all the messages for one channel first, then the ones for the next channel. This can result in alarmed administrators reporting that they've fixed one issue, but are only seeing only one specific channel dequeuing.

This is not a problem. There is a memory cost of approximately 140 bytes for each message. Having a message limit of 100000, you are limiting the job controller data structures to about 20 Megabytes (there are other data structures representing jobs, channels, destination hosts and so on). This is insignificant on a big server.

All the named objects in the job controller are tracked in a hash table. This is sized at the next power of 2 bigger than **max_cache_messages**, and is never re-sized. Each entry in this hash table is a pointer, so we are looking at a memory usage of four times **max_cache_messages** rounded up to a power of two. Being a hash table, this tends all to be in memory as the hash function is supposed to be random. This is another 0.5 Megabytes in the default case.

For information about pools and configuring the Job Controller, see the discussion about configuring message processing and delivery in *Messaging Server Reference*.

To Start and Stop the Job Controller

To start the Job Controller, run the following command:

```
start-msg job_controller
```

To shut down the Job Controller, run the following command:

```
stop-msg job_controller
```

To restart the Job Controller, run the following command:

```
imsimta restart job_controller
```

Restarting the Job Controller has the effect of shutting down the currently running Job Controller, then immediately starting a new one.

On Demand Mail Relay

Support for On Demand Mail Relay as specified in RFC 2645 has been completed. This support piggybacks off existing support for SMTP **TURN** and doesn't involve any new options. Specifically, the **turn_in** channel option now enables both **TURN** and **ATRN**.

Note that the optional parameter to the **ATRN** command is only allowed if the **ODMR** channel is marked **single_sys** or **single**. This is so messages sent to a particular email domain can be reliably retrieved without getting messages sent to other domains.

It is strongly recommended that each administrative domain that requires **ATRN** service be configured as a separate channel rather than relying on **ATRN**'s domain selection capabilities. Specifically, the recommended configuration process to provide relay on demand for a new administrative domain foo is as follows:

1. Create a new channel for the domain foo:

   ```
   tcp_foo mustsaslserver single_sys slave smtp turn_in
   tcp_auth-daemon
   ```

   Or in **msconfig**:

   ```
   set channel:tcp_foo.official_host_name tcp_foo-daemon
   set channel:tcp_foo.mustsaslserver
   ```

```
set channel:tcp_foo.single_sys
set channel:tcp_foo.slave
set channel:tcp_foo.smtp
set channel:tcp_foo.turn_in
```

Note the presence of the **slave** channel option. This prevents the channel from trying to perform regular SMTP deliveries. This can be removed if such delivery attempts are desired.

Also note the presence of **single_sys**. **multiple** can be used instead and will be more efficient if multiple email domains are involved and the **ATRN** command is always used without an argument.

2. Create rewrite rules for all email domains associated with the foo administrative domain, for example:

```
foo.example.com      $U%$D@tcp_foo-daemon
foo.example.org      $U%$D@tcp_foo-daemon
```

Or in **msconfig**:

```
set rewrite.rule foo.example.com $U%$D@tcp_foo-daemon
set rewrite.rule foo.example.org $U%$D@tcp_foo-daemon
```

3. Create a regular email account for foo with whatever name and credentials are desired. The account should include the LDAP attribute **mailSubmitChannel** with the value **tcp_foo**.
4. (optional) Create a dispatcher configuration identical to the regular service on port 25 except that it listens on port 366, the On Demand Mail Relay port.
5. (optional) Add a rule to the **FROM_ACCESS** mapping to block all mail coming from the ODMR port unconditionally.
6. (optional) Add a rule to the **FROM_ACCESS** mapping to block all mail from the **tcp_foo** channel unconditionally. This will prevent the administrative account from being used to send authenticated mail.

At this point **ATRN** should be usable by connecting, authenticating as the newly created user, and issuing an **ATRN**.

Priority Message Handling

This chapter describes Messaging Server's support of the **MT-PRIORITY** SMTP extension defined in RFC 6710. **MT-PRIORITY** values are always in the range -9 to 9. Priority 0 is the default.

The support for this extension consists of the following parts:

- The **mtprioritiesallowed** and **mtprioritiesrequired** source channel options. Both accept either one- or two-integer arguments. Two-integer arguments specify the range. You can specify the arguments in any order. Table 6–2 shows the channel options and their descriptions.

Table 6–2 MT-PRIORITY SMTP Extension Support

Channel Option	Description
mtprioritiesallowed *int1 [int1]*	Specifies the range of **MT-PRIORITY** values that will accepted. **MT-PRIORITY** values outside this range will be adjusted up or down so they fall within the allowed range. If a single argument is given it specifies the highest priority value that will be accepted. The default if this option is not specified is for the **MT-PRIORITY** extension not to be offered and for **MT-PRIORITY** options not to be accepted.
mtprioritiesrequired *int1 [int2]*	Specifies the range of **MT-PRIORITY** that will be accepted for enqueue. If a single argument is given it specifies the lowest priority value that will be accepted. The message will be rejected if the specified **MT-PRIORITY** value or the default value of 0 falls outside the required range.

- An **INCLUDE_MTPRIORITY** MTA option. This is a bit-encoded option. The default value is 0. Table 6–3 shows the bits, corresponding values, and descriptions of the **INCLUDE_MTPRIORITY** MTA option.

Table 6–3 INCLUDE_MTPRIORITY MTA Option

Bit	Value	Description
0	1	Appends the **MT-PRIORITY** and expected message size as separate fields to the **FROM_ACCESS** mapping probe immediately after any **INCLUDE_ SPARES** values.
1	2	Appends the **MT-PRIORITY** and expected message size as separate fields to the **FORWARD** mapping probe immediately after the conversion tag field.
2	4	Appends the **MT-PRIORITY** and expected message size as separate fields to the **ORIG_SEND_ACCESS** mapping probe immediately after any INCLUDE_SPARES values.
3	8	Appends the MT-PRIORITY and expected message size as separate fields to the SEND_ACCESS mapping probe immediately after any **INCLUDE_ SPARES** values.
4	16	Appends the **MT-PRIORITY** and expected message size as separate fields to the **ORIG_MAIL_ACCESS** mapping probe immediately after any **INCLUDE_SPARES** values.
5	32	Appends the **MT-PRIORITY** and expected message size as separate fields to the **MAIL_ACCESS** mapping probe immediately after any **INCLUDE_ SPARES** values.
6	64	Appends the **MT-PRIORITY** and actual message size values in the form: **;MT-PRIORITY=<value>;BLOCKS=<value>** to the conversion mapping probe immediately after any **tag=** clause.
7	128	Append the **MT-PRIORITY** and expected message size as separate fields to any domain catchall mapping probe immediately after the conversion tag.

The expected message size is the size of the queued message entry for internal channels. It is the value given by the SMTP **SIZE** extension for incoming SMTP channels. The size is given in MTA blocks.

- A **LOG_MTPRIORITY** MTA option. This is a bit-encoded option. The default is 0. Table 6–4 shows the bits, corresponding values, and descriptions of the **LOG_ MTPRIORITY** MTA option. An **mp** element is used in the new XML log format to log the priority information. See "Managing Logging" for more information about the XML log format.

Table 6–4 LOG_MTPRIORITY MTA Option

Bit	Value	Description
0	1	Enables logging of the **MT-PRIORITY** associated with each transaction. The **MT-PRIORITY** appears immediately after the message sensitivity and before the header-based priority in each log entry.
1	2	Message transfer priority appears in the **LOG_ACTION** mapping table probe, immediately after the sensitivity field and before the header-based priority field.

- Rewrite rule metacharacters that test both the current MT-PRIORITY value and the expected message size. For each metacharacter, *n* is a signed integer value. Note that the sign is required even when *n* is positive. The expected message size is the size of the queued message entry for internal channels. It is the value given by the SMTP **SIZE** extension for incoming SMTP channels. The size is given in MTA blocks.

  Table 6–5 shows the rewrite rule metacharacters and their descriptions.

Table 6–5 Rewrite Rule Metacharacters

Metacharacter	Description
$n<P	Rule succeeds only if *n* is less than the current **MT-PRIORITY**.
$n<=P	Rule succeeds only if *n* is less than or equal to the current **MT-PRIORITY**.
$n>P	Rule succeeds only if *n* is greater than the current **MT-PRIORITY**.
$n>=P	Rule succeeds only if *n* is greater than or equal to the current **MT-PRIORITY**.
$n=P	Rule succeeds only if *n* is equal to the current **MT-PRIORITY**.
$n<>P	Rule succeeds only if *n* is not equal to the current **MT-PRIORITY**.
$n<B	Rule succeeds only if *n* is less than the expected message size.
$n<=B	Rule succeeds only if *n* is less than or equal to the expected message size.
$n>B	Rule succeeds only if *n* is greater than the expected message size.
$n>=B	Rule succeeds only if *n* is greater than or equal to the expected message size.
$n=B	Rule succeeds only if *n* is equal to the expected message size.
$n<>B	Rule succeeds only if *n* is not equal to the expected message size.

- A **-mtpriority** switch added to **imsimta test -rewrite, imsimta calc**, and **imsimta test -expression** utilities. A single integer argument is required specifying the initial **MT-PRIORITY** value.
- A Sieve environment item, **vnd.oracle.mt-priority**. This item returns the current **MT-PRIORITY** value as a string.
- A nonstandard Sieve action, **setmtpriority**. This action accepts a single integer or string argument and sets the current **MT-PRIORITY** to the argument value. This action is only allowed in system-level Sieves and the argument must be in the -9 to 9 range of valid **MT-PRIORITY** values.
- A bit defined in the **MESSAGE_SAVE_COPY_FLAGS** MTA option. Bit 3 (value 8), if set, causes the **MT-PRIORITY** value for the current message to be included, delimited by vertical bars, immediately after the conversion tag.
- An **MTPRIORITY_POLICY** MTA option. This option is used to specify the priority handling policy the MTA has been configured to support. This name is

announced in the SMTP EHLO response on any channel where the **MT-PRIORITY** extension is enabled. The default is the empty string, meaning that no policy is announced.

MTA Command-line Utilities

Oracle Communications Messaging Server Message Transfer Agent (MTA) command-line utilities are used to perform various MTA maintenance, testing, and management tasks.

The MTA commands are also referred to as the **imsimta** commands. The **imsimta** script is located in the *MessagingServer_home*/**bin** directory.

For more information on MTA commands, see *Messaging Server Reference*.

7

LMTP Delivery

This chapter provides an overview of the Local Mail Transfer Protocol (LMTP), and describes how to configure both the LMTP client and server.

Overview of LMTP

The Oracle Communications Messaging Server MTA can use Local Mail Transfer Protocol (LMTP), as defined in RFC 2033, to deliver messages to the message store in a multi-tiered Messaging Server deployment. In this scenario, the front-end relays become responsible for address expansion and delivery methods such as autoreply and forwarding, and also for mailing list expansion. Delivery to the back-end stores historically has been over SMTP, which requires the back-end system to look up the recipient addresses in the LDAP directory again, thereby engaging the full machinery of the MTA. For speed and efficiency, the MTA can use LMTP rather than SMTP to deliver messages to the back-end store. The Messaging Server's LMTP server is not intended as a general purpose LMTP server, but rather as a private protocol between the relays and the back-end message stores. For simplicity of discussion, examples involving two-tiered deployments are used.

Note: LMTP is recommended for use in multi-tiered deployments. Also, the Messaging Server's LMTP service as implemented is not designed to work with third-party LMTP servers or third-party LMTP clients.

LMTP Delivery Features

The MTA's LMTP server is more efficient for delivering to the back-end message store because it:

- Reduces the load on the back-end stores. Because relays are horizontally scalable and back-end stores are not, it is good practice to push as much processing to the relays as possible.
- Reduces the load on the LDAP servers. The LDAP infrastructure is often a limiting factor in large messaging deployments.
- Reduces the number of message queues. Having queues on both the relay and the back-end store makes finding a lost message that much harder for administering a messaging deployment.

LMTP Client and Server to Detect and Respond to Certain Conditions

The LMTP client and server can detect and respond to the condition where a given host's LMTP server is responding but isn't associated with the master store replica. When this happens the LMTP server produces a banner or **MAIL FROM** response of the form:

```
423 4.3.2 Host not master for store; correct master host is HOSTNAME
```

or, if the correct master host is not known:

```
421 4.3.2 Host not master for store; cannot determine correct master host
```

When the LMTP client sees the 423 banner it will immediately disconnect and reconnect to the correct host. The affinity subsystem is also informed of the new master host.

The 421 banner is treated like any other 4YZ banner response; the client disconnects and tries the next host in the affinity group.

A 423 **MAIL FROM** response engages the LMTP client's fast retry logic; the client will abort the delivery attempt and disconnect but schedule the message for fast retry.

The handling of a 421 **MAIL FROM** response is unchanged.

The following LMTP channel-specific parameters support this capability:

- **WRONG_MASTER_FAST_RETRY** - Controls the actual value used to override the normal backoff value. The default is 1, meaning to ask the Job Controller for a fairly quick retry. Setting the option to 0 disables backoff override in this case. Positive values greater than 1 result in a retry, but not quite as quick of one. The formula used is n + rand() % (15 * n) seconds, where n is the **WRONG_MASTER_FAST_RETRY** value.
- **WRONG_MASTER_FALLBACK_ATTEMPTS** - Controls the number of times the LMTP client will honor a 423 banner redirect. The default is 1, meaning a single redirect will be honored for a given message.

Support for LMTP Client and Server to Use UID Extension

This extension, if present, provides a single UID parameter on the **MAIL FROM** command that accepts the values "NO" or "RET". If the latter is specified, the final LMTP responses to DATA/BDAT include the UID, UIDVALIDITY, digest value (if available), and optionally the folder if different from INBOX, separated by colons and enclosed in angle brackets. For example:

```
S: 220 multke.mrochek.com -- Server LMTP (Oracle Communications Messaging Server
8.0.2.0.0 64bit (built Sep 15 2017))
      C: LHLO multke.mrochek.com
      S: 250-multke.mrochek.com
      S: 250-8BITMIME
      S: 250-UID
      S: 250-PIPELINING
      S: 250-CHUNKING
      S: 250-XDFLG
      S: 250-XQUOTA
      S: 250-XAFLG
      S: 250-XSPARE
      S: 250-ENHANCEDSTATUSCODES
      S: 250-HELP
      S: 250 SIZE 0
      C: MAIL FROM:<> UID=RET
```

```
S: 250 2.5.0 Address Ok.
C: RCPT TO:<test1+folder@ims-ms-daemon> XDFLGS=5
S: 250 2.1.5 test1@ims-ms-daemon OK.
C: RCPT TO:<test2@ims-ms-daemon>
S: 250 2.1.5 test2@ims-ms-daemon OK.
C: DATA
S: 354 Enter mail, end with a single ".".
C: Subject: Test message
C:
C: This is a test.
C: .
S: 250 2.5.0 <1445028362:2::folder> Delivery to user OK
S: 250 2.5.0 <1440097745:2:> Delivery to user OK
```

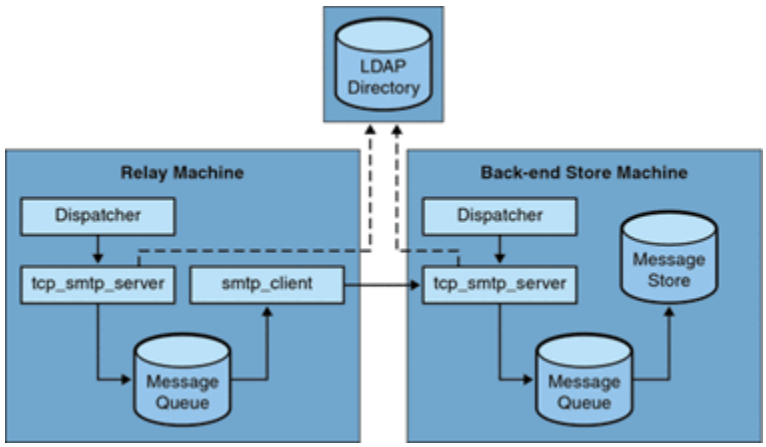
This extension is enabled by default and cannot be disabled. The LMTP client uses it if it is present to obtain UID information to use in conjunction with the **LOG_MAILBOX_UID** MTA option as well as the recall facility.

Additionally, the **LOG_MAILBOX_UID** MTA option now enables logging of UID information in the S records logged by the LMTP server. Note that this logging does not depend on use of the LMTP extension.

Messaging Processing in a Two-Tiered Deployment Without LMTP

Figure 7–1 shows message processing in a two-tiered deployment without LMTP.

Figure 7–1 Two-Tiered Deployment Without LMTP



Without LMTP, in a two-tiered deployment with relays "in front" of the store systems, inbound message processing begins with a connection on the SMTP port picked up by the dispatcher on the relay machine and handed off to a **tcp_smtp_server** process. This process does several things with the inbound message including:

- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain
- Rewriting the envelope address as **@mailhost:user@domain**
- Enqueuing the message for delivery to the mailhost

The **smtp_client** process then picks up the mail message from the queue and sends it to the mailhost. On the mailhost, some very similar processing takes place. A

connection on the SMTP port is picked up by the dispatcher and handed off to a **tcp_smtp_server** process. This process does several things to the message, including:

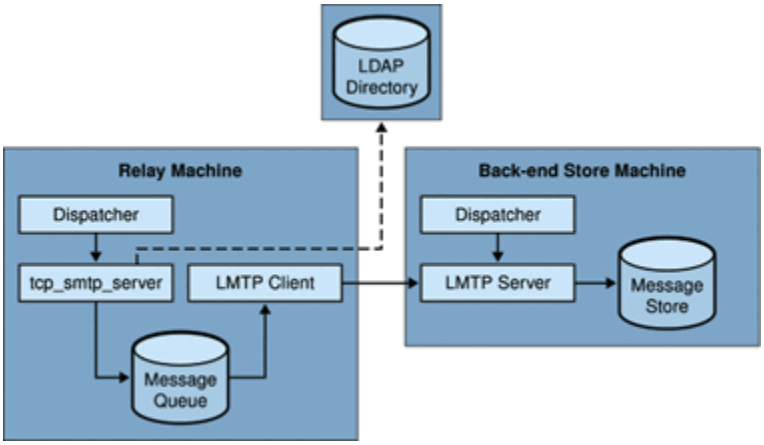
- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain
- Rewriting the envelope address to direct the message to the **ims_ms** channel
- Enqueuing the message for delivery to the store

Then the **ims_ms** process picks up the mail message and attempts to deliver it to the store. In this scenario, the enqueuing processing is performed twice, and the MTAs each perform an LDAP lookup.

Messaging Processing in a Two-Tiered Deployment With LMTP

Figure 7–2 shows message processing in a two-tiered deployment scenario with LMTP.

Figure 7–2 Two-Tiered Deployment With LMTP



With LMTP in place, a connection on the SMTP port of the relay machine is picked up by the dispatcher and handed off to a **tcp_smtp_server** process. This process does several things with the inbound message including:

- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain
- Determining which back end message store machine hosts the mailbox for the user
- Enqueuing the message for delivery to the mailhost

On the store machine, a connection to the LMTP port is received by the dispatcher and handed off to the **lmtp_server** process. The LMTP server then inserts the message into the user's mailbox or into the UNIX native mailbox. If message delivery is successful, the message is dequeued on the relay machine. If unsuccessful, the message remains on the relay machine. Note that the LMTP process on the message store does not engage any MTA machinery for processing addresses or messages.

LMTP Architecture

An LMTP configuration consists of setting up LMTP channels, one for the LMTP client and one for the LMTP server. LMTP channels are special cases of TCP/IP channels. The general SMTP-over-TCP/IP channel is often configured for bidirectional use. An LMTP channel, on the other hand, is configured to be dedicated for either client or server use. You configure the LMTP client channel on the MTA front-end relay host and the LMTP server channel on the back-end message store. Running **commpkg install** on both the front-end relay and back-end store system will configure LMTP properly.

For the most part, the MTA itself can be basically absent from the back-end server. The following items are the only necessary MTA components on the back end:

- The dispatcher
- The LMTP server
- A simple MTA configuration including **mappings**

The dispatcher must run on the back-end server so that it can start the LMTP servers that run under it. Because the dispatcher and the LMTP server use various functions of **libimta**, this needs to be present on the back-end server as well.

The LMTP server does not perform any of the usual MTA enqueuing or dequeuing functions, header processing, or address translations. The front-end relay system performs all the manipulation of the content of the messages and addresses, which then presents to the LMTP server the message in exactly the form to be delivered to the message store and with the delivery address already in the form required by the store. Additional recipient information that is usually available as a message that is delivered to the store, such as the user's quota, is presented along with the recipient address as LMTP options. Should a delivery attempt fail, the message is left enqueued in the LMTP queue on the relay system.

Configuring LMTP

Setting up LMTP requires that you configure both the front-end relay hosts and back-end message store hosts. On the relays, you must change the **mta.delivery_options** option so that messages being delivered to the stores are passed to the LMTP channel. The back-end store must be configured with the dispatcher, but does not need the job controller. On the LMTP client, you must configure the job controller to run the LMTP client.

This section describes how to configure the LMTP front ends and back ends by using Unified Configuration recipes. You could also perform the same configuration manually by using the **msconfig** command, however, using recipes is faster and makes the task repeatable. You run one recipe on the LMTP front ends and one on the LMTP back ends. Messaging Server ships with the following two LMTP recipes:

- **LMTPSingleSystem.rcp**: Use this recipe primarily for evaluation and testing purposes.
- **LMTPBackendFailover.rcp**: Use this recipe for both LMTP and failover without download. This recipe is easily modified (lines removed) for just LMTP support.

See "Using Recipes" and the discussion on recipe language in *Messaging Server Reference* for more information on Unified Configuration recipes.

Before You Begin

To see if any LMTP options are already enabled on your Messaging Server hosts, run the following command:

```
msconfig show mta.delivery_options -default
```

To Configure the Front-end MTA Relay with LMTP

Use the following "Example Recipe to Configure LMTP Front-end Relay with LMTP" to configure the inbound MTA relay for LMTP.

1. Make a copy of the example recipe file and save it as *recipe***.rcp** in the **config/recipes** directory.
2. In this example, replace the following items with your site-specific values:
   - **myIP**
   - **myNetmaskbits**
3. To run the recipe, type the following command:

   ```
   cd MessagingServer_home/bin
   msconfig run MessagingServer_home/config/recipes/recipe_name
   ```

4. Recompile if running a compiled configuration

   ```
   imsimta cnbuild
   ```

5. Restart Messaging Server.

   ```
   cd MessagingServer_home/bin
   stop-msg
   start-msg
   ```

Example Recipe to Configure LMTP Front-end Relay with LMTP

```
# -*- mode: sieve; -*-
description("frontendMTA to backend LMTP store");
keywords(["frontend","MTA", "LMTP"]);
#
# Sample recipe for a frontend MMP/MTA to a backend LMTP store
#   the corresponding recipe for the backend store/LMTP is backendLMTP.rcp
#
################################################################################
# !!!!!!!!!!!!!!!!!!!!
# CHANGE THESE
# !!!!!!!!!!!!!!!!!!!!
#
# constants - supplied input
#
# my IP address
myIP = "10.133.158.10";
# network portion of IP address in number of bits
myNetmaskbits = "8";

################################################################################
#
# configure LMTP frontend
#

#
# disable store
```

```
#
set_option("store.enable", "0");
set_option("imap.enable", "0");
set_option("pop.enable", "0");

#
# add to rewrite rules
# .lmtp $E$F$U%$H.lmtp@lmtpcs-daemon
# .lmtp $B$F$U%$H@$H@lmtpcs-daemon
#
# should really check to see if the rewrite rule exists instead
# of unilaterally appending it
#
append_rewrites([".lmtp", "$E$F$U%$H.lmtp@lmtpcs-daemon"]);
append_rewrites([".lmtp", "$B$F$U%$H@$H@lmtpcs-daemon"]);

#
# add channel tcp_lmtpcs
# ,----
# | tcp_lmtpcs defragment lmtp multigate connectcanonical fileinto @$4O:$U+$S@$D
f\
# | lagtransfer multigate connectcanonical port 225 nomx sin\
# | gle_sys pool SMTP_POOL dequeue_removeroute
# | lmtpcs-daemon
# `----
#
set_option("channel:tcp_lmtpcs.connectcanonical");
set_option("channel:tcp_lmtpcs.defragment");
set_option("channel:tcp_lmtpcs.fileinto", "@$4O:$U+$S@$D");
set_option("channel:tcp_lmtpcs.flagtransfer");
set_option("channel:tcp_lmtpcs.lmtp");
set_option("channel:tcp_lmtpcs.multigate");
set_option("channel:tcp_lmtpcs.nomx");
set_option("channel:tcp_lmtpcs.pool", "SMTP_POOL");
set_option("channel:tcp_lmtpcs.port", "225");
set_option("channel:tcp_lmtpcs.single_sys");
set_option("channel:tcp_lmtpcs.dequeueremoveroute");
set_option("channel:tcp_lmtpcs.official_host_name", "lmtpcs-daemon");

#
# ! The modified DELIVERY_OPTIONS which activate LMTP
# ! delivery from a frontend relay to the backend
#
# DELIVERY_OPTIONS=\
#       #*mailbox=@$X.LMTP:$M%$\$2I$_+$2S@lmtpcs-daemon,\
#       #&members=*,\
#       #*native=@$X.LMTPN:$M+$2S@native-daemon,\
#       #*unix=@$X.LMTPN:$M,\
#       #*file=@$X.LMTPN:+$F,\
#       #&@members_offline=*,\
#       #/hold=@hold-daemon:$A,\
#       #program=$M%$P@pipe-daemon,\
#       #forward=**,\
#       #*^!autoreply=$M+$D@bitbucket
#
# NOTE NOTE NOTE - have to escape the backslash in the "mailbox=..."
set_option("mta.delivery_options", "#*mailbox=@$X.LMTP:$M%$\\$2I$_
+$2S@lmtpcs-daemon,#&members=*,#*native=@$X.LMTPN:$M+$2S@native-daemon,#*unix=@$X.
LMTPN:$M,#*file=@$X.LMTPN:+$F,#&@members_
offline=*,#/hold=@hold-daemon:$A,#program=$M%$P@pipe-daemon,#forward=**,#*^!autore
```

```
ply=$M+$D@bitbucket");

########################################################################
write("use \"write -remark frontendLMTP.rcp\" to write out the changes\n");
```

This recipe performs the following configuration on the LMTP front end:

1. Disables the message store, as it is not needed.
2. Enables the necessary rewrite rules. The recipe language **append_rewrites** function adds entries to existing rewrite rules, or, if none exists, adds them as new rules.
3. Adds the **tcp_lmtpcs** channel with the appropriate options:
   - **connectcanonical**: Tells the MTA to compare the recipient envelope address domain with the channel host proper names, and if the domain name matches one of the channel's host proper names, then connect to the host name corresponding to that host proper name.
   - **defragment**: Any message/partial messages queued to the channel are placed in the **defragmentation** channel queue instead. Once all the parts have arrived, the message is rebuilt and sent on its way.
   - **fileinto**, value of **@$4O:$U+$S@$D**: Specifies how to alter an address when a Sieve filter "fileinto" action is applied. In $4O, the O is the capital or majuscule letter "o", not the numeral zero 0. The effect is that the explicit source route to the mailhost should be preserved if present, and the foldername should be inserted as a subaddress into the original address, replacing any originally present subaddress.
   - **flagtransfer**: Enables SMTP client support of the XDFLG private SMTP extension command.
   - **lmtp**: Specifies that channel supports LMTP protocol.
   - **multigate**: Instructs the MTA to route the message to the daemon mailbox specified by the daemon channel option on the system specified in the message's To: address.
   - **nomx**: Disables MX lookups.
   - **pool**, value of **SMTP_POOL**: Specifies the **SMTP_POOL** pool where the jobs are created for this channel.
   - **port**, value of **225**: Specifies dispatcher port number.
   - **single_sys**: Creates a single copy of the message for each destination system (more precisely, each destination domain name) associated with a recipient address.
   - **dequeueremoveroute**: Causes source routes to be stripped from envelope recipient addresses when the channel dequeues messages (but after the channel has determined how to route the message).
   - **official_host_name**, value of **lmtpcs-daemon**: Specifies the "name" of the system with which this channel communicates. (In legacy configuration, the official host name is specified as the first name on the second line of a channel definition.)
4. Finally, the delivery options are set, which activates LMTP delivery from the front-end relay to the back-end message store. In the ***mailbox=@$X.LMTP:$M%$\$2I$_+$2S@lmtpcs-daemon,** portion of the delivery

options, the script "escapes" the forward slash, so that the actual entry in the recipe is as follows:

```
..."#*mailbox=@$X.LMTP:$M%$\\$2I$_+$2S@lmtpcs-daemon,#&members=*,...
```

Note: You must add the full **delivery_options** as shown in the example. You cannot override just the default for one option. If you specify **delivery_options** at all, you must define them all.

To Configure Back-End Stores with LMTP and a Minimal MTA

Use the "Example Recipe to Configure LMTP Back-end Store with LMTP" to configure the back-end store for LMTP. This recipe:

- Prompts for **myIP** and **feIPs**, if you do not specify them in the recipe (but does not validate the IP addresses)
- Checks if the **tcp_lmtpss** channel exists before creating it
- Checks if the dispatcher group exists before creating it
- Checks if **PORT_ACCESS** entries exist before creating them
- Treats **feIPs** as a list to allow multiple front ends
- Can be run multiple times correctly

To create and run the recipe, follow these steps:

1. Make a copy of the example recipe file and save it as *recipe***.rcp** in the **config/recipes** directory.
2. In this example, replace the following items with your site-specific values:
   - **myIP**
   - **feIPs**
3. To run the recipe, type the following command:

   ```
   cd MessagingServer_home/bin
   msconfig run MessagingServer_home/config/recipes/recipe_name
   ```

4. Recompile if running a compiled configuration.

   ```
   imsimta cnbuild
   ```

5. Restart Messaging Server.

   ```
   cd MessagingServer_home/bin
   stop-msg
   start-msg
   ```

Example Recipe to Configure LMTP Back-end Store with LMTP

```
# -*- mode: sieve; -*-
description("backend store via LMTP");
keywords(["backend", "store", "LMTP"]);
#
# Sample recipe for a backend store via LMTP
#   the corresponding recipe for the frontend MMP/MTA is frontendLMTP.rcp
##########################################################################
# !!!!!!!!!!!!!!!!!!!!
# CHANGE THESE
```

```
# !!!!!!!!!!!!!!!!!!!!
#
# constants - supplied input
# if you leave it blank, the script will prompt for it
#  sample entry
#myIP = "10.133.152.193";
myIP = "";

#
# list of frontend machines that access this store via LMTP
# if you leave it blank, the script will prompt for it
#   sample entry:
#feIPs = ["10.133.152.192", "10.133.152.193"];
feIPs = [];

########################################################################
# prompt for myIP and feIP if needed


if (length(myIP) <= 0) {
  myIP = read("Enter the IP address of this host: ");
}

if (length(feIPs) <= 0) {
  loop {
    ip = read("Enter the IP address of a frontend machine (<RET> if no more): ");
    exitif (ip == "");
    push(feIPs, ip);
  }
}

########################################################################
#
# configure LMTP backend

#
# create tcp_lmtpss channel. In legacy config, this would show up as:
#   tcp_lmtpss lmtp flagtransfer identnonenumeric
#   tcp_lmtpss-daemon
if exists_channel("tcp_lmtpss") {
  warn("-- WARNING: tcp_lmtpss channel already exists.");
} else {
  # This creates the channel by using individual options
  #set_option("channel:tcp_lmtpss.flagtransfer");
  #set_option("channel:tcp_lmtpss.identnonenumeric");
  #set_option("channel:tcp_lmtpss.lmtp");
  #set_option("channel:tcp_lmtpss.official_host_name", "tcp_lmtpss-daemon");
  # an alternative way of doing it as a one-liner
  print("-- INFO: Adding tcp_lmtpss channel\n");
  add_channel("tcp_lmtpss",
              ["flagtransfer", "",
               "identnonenumeric", "",
               "lmtp", "",
               "official_host_name", "tcp_lmtpss-daemon"]);
}

#
# dispatcher.cnf
#  uncomment [SERVICE=LMTPSS] block
#
```

```
if exists_group("dispatcher.service:LMTPSS") {
  warn("-- WARNING: dispatcher.service:LMTPSS group already exists");
} else {
  print("-- INFO: Creating dispatcher.service:LMTPSS group\n");
  # This creates the group by using individual options
  #set_option("dispatcher.service:LMTPSS.image", "IMTA_BIN:tcp_lmtp_server");
  #set_option("dispatcher.service:LMTPSS.logfilename", "IMTA_LOG:tcp_lmtpss_
server.log");
  #set_option("dispatcher.service:LMTPSS.parameter", "CHANNEL=tcp_lmtpss");
  #set_option("dispatcher.service:LMTPSS.tcp_ports", "225");
  #set_option("dispatcher.service:LMTPSS.stacksize", "2048000");
  #set_option("dispatcher.service:LMTPSS.enable", "1");

  # alternate way of doing this in one line
  add_group("dispatcher.service:LMTPSS",
            ["image", "IMTA_BIN:tcp_lmtp_server",
             "logfilename", "IMTA_LOG:tcp_lmtpss_server.log",
             "parameter", "CHANNEL=tcp_lmtpss",
             "tcp_ports", "225",
             "stacksize", "2048000",
             "enable", "1"]);
}

#
# add PORT_ACCESS mapping entries
#
#   allow frontends (feIPs) to access LMTP port
# TCP|*|225|10.133.152.192|*  $Y
# TCP|*|226|10.133.152.192|*  $Y
#
# ! Allow 'msprobe' on this host (myIP) to connnect to the LMTP ports
# !
# TCP|*|225|10.133.152.193|*  $Y
# TCP|*|226|10.133.152.193|*  $Y

portAccess_optlist =  get_mapping("PORT_ACCESS");
#print ("\n -- DEBUG optlist for PORT_ACCESS" . portAccess_optlist . "\n");

# list of IP addresses
ipaddrs = [feIPs];
push(ipaddrs, myIP);
numips = length(ipaddrs);
ports = ["225", "226"];
numports = length(ports);

i = 1;
loop {                          #loop over all ipaddrs
  p = numports;
  loop {
    tmp = "TCP|*|" + ports[p] + "|" + ipaddrs[i] + "|*";
    if exists_optlist(get_mapping("PORT_ACCESS"), tmp) {
      warn ( "-- WARNING: optlist for " . tmp . " exists in PORT_ACCESS");
    } else {
      print("-- INFO: Adding IP " + ipaddrs[i] + " port " + ports[p] + " to PORT_
ACCESS mapping\n");
      prepend_mapping("PORT_ACCESS", [tmp, "$Y"]);
    }
    exitif(p==1);
    p--;
  }
```

```
  exitif(i==numips);
  i++;
}

##########################################################################
write("use \"write -remark backendLMTP.rcp\" to write out the changes\n");
```

LMTP Protocol as Implemented

This section provides a sample LMTP dialogue with an explanation of what is seen in that dialogue. The LMTP client on the relay uses standard LMTP protocol to talk to the LMTP server on the back end store. However the protocol is used in specific ways. For example:

```
---> LHLO
<--- 250 OK
```

No action is taken on the **LHLO** message. The reply is always **250 OK**.

```
---> MAIL FROM: address size=messageSizeInBytes
<--- 250 OK
```

No checks or conversions are made on the originator address. The **size=** option gives a size in bytes for the message that is to be delivered. This is the size of the message exactly as it appears in the protocol. It is not necessarily the exact size of the message, but the actual message size will not exceed this size. The LMTP server allocates a memory buffer of this size to receive the message.

```
---> RCPT TO: uid+folder@domain xquota=size,number xdflg=xxx
<--- 250 OK
```

No checks are made on the recipient addresses at the time they are received, but a list of recipients is built for later use. Note that the **@domain** part of the address is omitted for **uids** in the primary domain, and that the **+folder** part is optional. This is the same address format used by the message store channel in the MTA.

The **xquota=** option gives the user's message quotas which consist of the maximum total size and the maximum number of messages. The MTA provides this information which it retrieves while performing an LDAP lookup on the user to do the address translation. This information is used to keep the quota information in the message store synchronized with the directory. Getting the quota information does not result in an additional performance hit.

The **xdflg=** option specifies a number which is interpreted as a bit field. These bits control how the message is delivered. For example, the bit whose value is 2, if set, guarantees delivery of the message even if the user is over quota. (Note that **xdflg** is an internal option and the bits in it are subject to change or addition without notice. Oracle does not support other clients using this extension with Messaging Server, nor does Oracle support using the Messaging Server LMTP client with some other server and this option.)

This interaction may be repeated many times, once for each recipient.

```
--->DATA
---> <the message text>
--->.
```

The LMTP client then sends the entire message, dot-stuffed, just as SMTP does. The message finishes with a dot (**.**) alone on a line. If the message size is exceeded the LMTP server sends:

<--- 500 message too big

and ends the connection.

Assuming that the message is received correctly, the LMTP server then sends back to the LMTP client the status for each recipient given in the **RCPT TO:** lines. For instance, if the message is delivered successfully, the response is:

<--- 250 2.5.0 address OK

where **address** is exactly as it appeared on the **RCPT TO:** line.

The conversation can either repeat with another **MAIL FROM:** line or end with the following interaction:

```
---> quit
<--- 221 OK
```

Table 7–1 shows the possible status codes for each recipient. This three-column table shows the short code in the first column, its long-code equivalent in the second column and the status text in the third column. 2.x.x status codes are success codes, 4.x.x codes are retryable errors, and 5.x.x codes are non-retryable errors.

Table 7–1 LMTP Status Codes for Recipients

Short Code	Long Code	Status Text
250	2.5.0	OK
420	4.2.0	Mailbox Locked
422	4.2.2	Quota Exceeded
420	4.2.0	Mailbox Bad Formats
420	4.2.0	Mailbox not supported
430	4.3.0	IMAP IOERROR
522	5.2.2	Persistent Quota Exceeded
523	5.2.3	Message too large
511	5.1.1	mailbox nonexistent
560	5.6.0	message contains null
560	5.6.0	message contains nl
560	5.6.0	message has bad header
560	5.6.0	message has no blank line

Otherwise, there are changes to the delivery options for mailbox, native (and, therefore, UNIX), and file. The object of these rules is to generate addresses that will cause the messages to be sent through the appropriate LMTP channel to the back end servers. The addresses generated are source routed addresses of the form:

```
@sourceroute:_localpart_@_domain_
```

8

Vacation Automatic Message Reply

For automatically generated responses to email (autoreply), specifically vacation messages, the MTA uses Message Disposition Notifications (*MDNs*) and the Sieve scripting language. MDNs are email messages sent by the MTA to a sender and/or postmaster reporting on a message's delivery disposition. MDNs are also known as read receipts, acknowledgments, receipt notifications, or delivery receipts. The Sieve is a simple scripting language used to create mail filters.

This chapter describes the vacation autoreply mechanism. In most cases, you do not need to modify the default configuration. However, you might want to configure your system such that an MTA relay performs vacation processing rather than the back-end message stores.

Vacation Autoreply Overview

Vacation Sieve scripts are generated automatically from the various LDAP vacation attributes (see "Vacation Autoreply Attributes"). They can also be specified explicitly for additional flexibility. The underlying mechanism for tracking vacations is a set of files, one per intended recipient, that keep track of when replies were sent to the various senders.

By default, the MTA evaluates vacation on the back-end store systems. However, because MTA relays do not do as much work as back-end stores, for performance reasons, you can have the MTA evaluate vacation on the mail relay machines instead of on the back-end store. Use of this feature, however, can result in vacation responses being sent out more often than intended because different relays handle different messages. If you do not want vacation messages to be sent out more often than you intend, you may share the tracking of files between the relays. If this is also unacceptable to you, you can always have vacation evaluated on the back-end store systems.

Configuring Autoreply

Delivery addresses are generated through a set of patterns. The patterns used depend upon the values defined for the **mailDeliveryOption** attribute. One delivery address is generated for each valid **mailDeliveryOption** that is set in the recipient's LDAP entry. The patterns are defined by the setting of the **delivery_options** MTA option, which consists of a comma-separated list of *option=pattern* pairs. The default autoreply *option=pattern* pair is:

```
^*!autoreply=$M+$D@bitbucket
```

Table 8–1 shows the prefix characters used for the autoreply rule in the first column and their definitions in the second column.

Table 8–1 Prefix Characters for the Autoreply Rule in delivery_options

Prefix Character	Definition
!	Enables the generation of the autoreply Sieve script.
#	Allows the processing to take place on relays.
^	Option is only evaluated if the vacation dates indicate that it should be evaluated.
*	Option is only evaluated for users, not groups.
@	Extracts preferred language information from various message header fields as well as from LDAP entries associated with envelope **From:** addresses. For this information to be available at the correct time, the message must pass through the reprocess channel when autoreply is engaged. This is done by adding the **@** flag to the **autoreply** delivery option. The addition of a channel hop increases message processing overhead.

The autoreply rule itself specifies an address destined for the bitbucket channel. The mail is considered delivered by this method once the autoreply is generated, but the MTA machinery requires a delivery address. Anything delivered to the bitbucket channel is discarded.

To Configure Autoreply on the Back-end Store System

The default autoreply rule in **delivery_options** causes the autoreply to take place on the mail server that serves the user. If you want vacation messages to be evaluated on the back-end store system, you do not have to configure anything. This is the default behavior.

To Configure Autoreply on a Relay

If you want to evaluate vacation on the relay rather than on the back-end store system to enhance performance, add the **#** prefix to the autoreply pattern. This can be done by performing the following steps:

1. Use the **msconfig** command to determine the current setting of **delivery_options**:

```
msconfig
msconfig> show delivery_options
```

2. If the **delivery_options** MTA option is not set, determine the current default by using the **show -default** command and then cutting and pasting to set the option to that default:

```
msconfig> show -default delivery_options
delivery_options: *mailbox=$M%$\$2I$_
+$2S@ims-ms-daemon,&members=*,*native=$M@native-daemon,/hold=@hold-daemon:$A,*
unix=$M@native-daemon,&file=+$F@native-daemon,@members_
offline=*,program=$M%$P@pipedaemon,#forward=**,^*!autoreply=$M+$D@bitbucket,#*
&nomail=$M+$D@bitbucket
msconfig> set delivery_options "*mailbox=$M%$\$2I$_
+$2S@ims-ms-daemon,&members=*,*native=$M@native-daemon,/hold=@hold-daemon:$A,*
unix=$M@native-daemon,&file=+$F@native-daemon,&@members_
offline=*,program=$M%$P@pipe-daemon,#forward=**,^*!autoreply=$M+$D@bitbucket,#
*&nomail=$M+$D@bitbucket"
```

3. Once the option is set, you can use the **edit option** command to edit the option value and add the **#** if it is missing:

```
msconfig# edit option delivery_options
*mailbox=$M%$$2I$_
+$2S@ims-ms-daemon,&members=*,*native=$M@native-daemon,/hold=@
hold-daemon:$A,*unix=$M@native-daemon,&file=+$F@native-daemon,&@members_
offline=
*,program=$M%$P@pipe-daemon,#forward=**,^*!autoreply=$M+$D@bitbucket,#*&nomail
=$
M+$D@bitbucket
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"/var/tmp/manage.jgaiy2" [Incomplete last line] 1 line, 254 characters
```

4. After changing the **delivery_options** option, use the **msconfig diff** command to verify that you have made the correct change. Then write the change out after you have made sure that it is correct:

```
msconfig# diff
< role.mta.delivery_options = *mailbox=$M%$$2I$_
+$2S@ims-ms-daemon,&members=*,*native=$M@native-daemon,/hold=@hold-daemon:$A,*
unix=$M@native-daemon,&file=+$F@native-daemon,&@members_
offline=*,program=$M%$P@pipe-daemon,#forward=**,^#*!autoreply=$M+$D@bitbucket,
#*&nomail=$M+$D@bitbucket
msconfig# write -remark="Enable autoreply processing on relays"
```

   Autoreply processing now takes place on the relays.

To Share Autoreply Information Between Relays

For information about sharing autoreply information between relays, see the discussion about the **vacation_template** autoresponse periodicity MTA option in *Messaging Server Reference*.

Vacation Autoreply Theory of Operation

The vacation action, when invoked, works as follows:

1. Oracle Communications Messaging Server checks to make sure that the vacation action was performed by a user-level rather than a system-level Sieve script. An error results if vacation is used in a system-level script.
2. The "no vacation notice" internal MTA flag is checked. If it is set, processing terminates and no vacation notice is sent.
3. The return address for the message is checked next. If it is blank, processing terminates and no vacation notice is sent.

4. The MTA checks to see if the user's address or any of the additional addresses specified in the **:addresses** tagged argument appear in a **To:**, **Cc:**, **Resent-to:**, or **Resent-cc:** header field for the current message. Processing terminates and no vacation notice is sent if none of the addresses is found in any of the header fields.
5. Messaging Server constructs a hash of the **:subject** argument and the reason string. This string, along with the return address of the current message, is checked against a per-user record of previous vacation responses. Processing terminates and no response is sent if a response has already been sent within the time allowed by the **:days** argument.
6. Messaging Server constructs a vacation notice from the **:subject** argument, reason string, and **:mime** argument. Two basic forms for this response message are possible:
   - A message disposition notification of the form specified in RFC 2298, with the first part containing the reason text.
   - A single part text reply. (This form is only used to support the "reply" autoreply mode attribute setting.)

The **mailautoreplymode** is automatically set to **reply** when vacation messages are configured through mshttpd.

The "no vacation notice" MTA flag is clear by default. It can be set by a system-level Sieve script through the use of the nonstandard **novacation** action. The **novacation** Sieve action is only allowed in a system-level Sieve script. It generates an error if it is used in a user-level script. You can use this action to implement site-wide restrictions on vacation replies such as blocking replies to addresses containing the substring "**MAILER-DAEMON**."

Per-user per-response information is stored in a set of flat text files, one per local user. The location and naming scheme for these files is specified by the setting of the **mta.vacation_template** option. This option should be set to a **file: URL**.

Maintenance of these files is automatic and controlled by the **mta.vacation_cleanup** integer MTA option setting. Each time one of these files is opened, the value of the current time in seconds modulo this value is computed. If the result is zero the file is scanned and all expired entries are removed. The default value for the option is 200, which means that there is 1-in-200 chance that a cleanup pass is performed.

The machinery used to read and write these flat text files is designed in such a way that it should be able to operate correctly over NFS. This enables multiple MTAs to share a single set of files on a common file system.

Vacation Autoreply Attributes

The set of user LDAP directory attributes that the vacation action uses are:

- Attribute defined by the MTA option **mta.ldap_autoreply_addressess** This attribute provides the ability to generate **:addresses** arguments to sieve vacation. This option has no value by default. The attribute can be multivalued, with each value specifying a separate address to pass to the **:addresses** vacation option.
- Attribute defined by **mta.ldap_personal_name** Alias processing keeps track of personal name information specified in this attribute and uses this information to construct From: fields for any MDNs or vacation replies that are generated. Use with caution to avoid exposing personal information.
- **vacationStartDate** Vacation start date and time. The value is in the format **YYYYMMDDHHMMSSZ**. This value is normalized to GMT. An autoreply should

only be generated if the current time is after the time specified by this attribute. No start date is enforced if this attribute is missing. The MTA can be instructed to look at a different attribute for this information by setting the **mta.lda_start_date** MTA option to a different attribute name. This attribute is read and checked by the code that generated the Sieve script. Vacation processing is aborted if the current date is before the vacation start date. This attribute cannot be handled by the script itself because at present Sieve lacks date/time testing and comparison facilities.

- **vacationEndDate** Vacation end date and time. The value is in the format **YYYYMMDDHHMMSSZ**. This value is normalized to GMT. An autoreply should only be generated if the current time is before the time specified by this attribute. No end date is enforced if this attribute is missing. The MTA can be instructed to look at a different attribute for this information by setting the **mta.ldap_end_date** MTA option to a different attribute name. This attribute is and checked by the code that generated the Sieve script. Vacation processing is aborted if the current date is after the vacation end date. This attribute cannot be handled in the script itself because at present Sieve lacks date/time testing and comparison facilities.
- **mailAutoReplyMode** Specifies autoreply mode for the user mail account. Valid values of this attribute are:
  - **echo** - Create a multipart that echoes the original message text in addition to the added **mailAutoReplyText** or **mailAutoReplyTextInternal** text.
  - **reply** - Send a single part reply as specified by either **mailAutoReplyText** or **mailAutoReplyTextInternal** to the original sender. These modes appear in the Sieve script as nonstandard **:echo** and **:reply** arguments to the vacation action. **echo** produces a "processed" message disposition notification (MDN) that contains the original message as returned content. **reply** produces a pure reply containing only the reply text. An illegal value does not manifest as any argument to the vacation action and this produces an MDN containing only the headers of the original message. Selecting an autoreply mode of echo causes an automatic reply to be sent to every message regardless of how recently a previous reply was sent. The MTA can be instructed to use a different attribute for this information by setting the **mta.ldap_autoreply_mode** MTA option to a different attribute name.
- **mailAutoReplySubject** Specifies the contents of the subject field to use in the autoreply response. This must be a UTF-8 string. This value gets passed as the **:subject** argument to the vacation action. The MTA can be instructed to use a different attribute for this information by setting the **mta.ldap_autoreply_subject** MTA option to a different attribute name.
- **mailAutoReplyText** Autoreply text sent to all senders except users in the recipient's domain. If not specified, external users receive no vacation message. The MTA can be instructed to use a different attribute for this information by setting the **mta.ldap_autoreply_text** MTA option to a different attribute name.
- **mailAutoReplyTextInternal** Auto-reply text sent to senders from the recipients domain. If not specified, then internal users get the mail autoreply text message. The MTA can be instructed to use a different attribute for this information by setting the **mta.ldap_autoreply_text_int** MTA option to a different attribute name. The MTA passes either the **mailAutoReplyText** or **mailAutoReplyTextInternal** attribute value as the reason string to the vacation action.
- **mailAutoReplyTimeOut** Duration, in hours, for successive autoreply responses to any given mail sender. Used only when **mailAutoReplyMode=reply**. If value is *0* then a response is sent back every time a message is received. This value is converted to the nonstandard **:hours** argument to the vacation action. (Normally

the Sieve vacation action only supports the **:days** argument for this purpose and does not allow a value of **0**.) If this attribute does not appear on a user entry, a default time-out is obtained from the **mta.autoreply_timeout_default** MTA option. The MTA can be instructed to use a different attribute for this information by setting the **mta.ldap_autoreply_timeout** MTA option.

The MTA can choose between multiple LDAP attributes and attribute values with different language tags and determine the correct value to use. The language tags in effect are compared against the preferred language information associated with the envelope from address. Currently the only attributes receiving this treatment are **mta.ldap_autoreply_subject** (normally **mailAutoReplySubject**), **mta.ldap_autoreply_text** (normally **mailAutoReplyText**), **mta.ldap_autoreply_text_int** (normally **mailAutoReplyTextInternal**), **mta.ldap_spare_4**, **mta.ldap_spare_5**, **mta.ldap_prefix_text** and **mta.ldap_suffix_text**.

It is expected that each attribute value has a different language tag value. If different values have the same tag value the choice between them is essentially random.

Other Auto Reply Tasks and Issues

This section describes auto reply tasks and issues not described in the configuration section.

To Send Autoreply Messages for Email That Have Been Automatically Forwarded from Another Mail Server

An autoreply problem can occur when the MTA receives a message that has been automatically forwarded from another system in some other administrative domain. For example, if a customer has a home account with **example.com** and the customer sets that account to automatically forward messages to their work account at **example.org** and if **example.org** uses Messaging Server and that user has set his account to autoreply a vacation message, then Messaging Server has a problem sending out a vacation message.

The problem occurs because the **example.com** mail server changes the envelope **To:** address from **user@example.com** to **user@example**, but it does not change the **To:** header, which remains **user@example.com**. When the MTA receives the message, it looks at the header address only. It attempts to match this address with an address in the LDAP user directory. If it finds a match with someone who has set autoreply, then a vacation message is sent. Because there is no LDAP address match to **user@example.com**, no vacation message is sent. The problem is that the actual address is in the envelope and not in the header.

Because the recipient's address known to the remote system doing automatic forwarding is not known to correspond to the user by the local system, there needs to be a way for the recipient to make such addresses known to the local system so vacation replies are sent when necessary.

The **:addresses** argument to the Sieve **vacation** action provides this capability. It accepts a list of addresses that correspond to the recipient for purposes of making this check. The attribute defined by the MTA option **ldap_autoreply_addresses** allows specification of such addresses in the user's LDAP entry.

To provide autoreply capability for messages that have been automatically forwarded from mail servers in some other administrative domain, the user or administrator would set the email addresses from where those messages may be forwarded to the attribute defined by **mta.ldap_autoreply_addresses**.

9

Using and Configuring MeterMaid for Access Control

This chapter describes the MeterMaid server, which can provide centralized metering and management of connections and transactions through monitoring IP addresses and SMTP envelope addresses. Functionally, MeterMaid can be used to limit how often a particular IP address can connect to the MTA. Limiting connections by particular IP addresses is useful for preventing excessive connections used in denial-of-service attacks. MeterMaid supplants conn_throttle.so by providing similar functionality, but extending it across the Oracle Communications Messaging Server installation. No new enhancements are planned for **conn_throttle.so** and MeterMaid is its more effective replacement.

Overview of MeterMaid

conn_throttle.so is a shared library used as a callout from the MTA's mapping table that uses an in-memory table of incoming connections to determine when a particular IP address has recently connected too often and should be turned away for awhile. While having an in-memory table is good for performance, its largest cost is that each individual process on each server maintains its own table.

In most cases, the conn_throttle.so callout is done in the PORT_ACCESS mapping that is accessed by the Dispatcher, a single process on each system. The only cost is that there is a separate table per server.

The primary improvement by MeterMaid is that it maintains a single repository of the throttling information that can be accessed by all systems and processes within the Messaging Server environment. It continues to maintain an in-memory database to store this data to maximize performance. Restarting MeterMaid will lose all information previously stored, but since the data is typically very short lived, the cost of such a restart (done infrequently) is very low.

How MeterMaid Works

MeterMaid's configuration is maintained by the **msconfig** command in Unified Configuration or the **configutil** command in legacy configuration.

MeterMaid is accessed from the MTA through a mapping table callout using **check_metermaid.so**. It can be called from any of the ***_ACCESS** tables. When called from the **PORT_ACCESS** table, it can be used to check limits based on the IP address of the connection which will be the most common way to implement MeterMaid as a replacement for the older **conn_throttle.so**. If called from other ***_ACCESS** tables, MeterMaid can also be used to establish limits on other data such as the envelope from or envelope to addresses as well as IP addresses.

This chapter only describes the throttle entry point in **check_metermaid.so**. See "MeterMaid Reference" for a complete list of **check_metermaid.so** entry points. The throttle routine contacts MeterMaid providing two subsequent arguments separated by commas. The first is the name of the table against which the data will be checked, and the second is the data to be checked.

If the result from the probe is that the particular data being checked has exceeded its quota in that table, **check_metermaid.so** returns **success** so that the mapping engine will continue processing this entry. The remainder of the entry would then be used to handle this connection that has exceeded its quota.

```
PORT_ACCESS

  *|*|*|*|* $C$|INTERNAL_IP;$3|$Y$E
  *|*|*|*|* $C$:A$[/opt/sun/comms/messaging64/lib/ check_
metermaid.so,throttle,tablename,$3]\
 $N421$ Connection$ declined$ at$ this$ time$E
   * $YEXTERNAL
```

Note the **$:A** flag test in the mapping table entry before the call to check_ metermaid.so. This is to ensure that we only do the MeterMaid probe when **PORT_ ACCESS** is being checked by the dispatcher as it will set the **A** flag for its probe.

Options for MeterMaid

MeterMaid's configuration is maintained by setting options using the **msconfig** command in Unified Configuration or using the **configutil** command in legacy configuration. Table 9–1 describes some of the settings currently supported by MeterMaid.

Table 9–1 Options for MeterMaid

Unified Configuration Option	Legacy Configuration Option	Description
metermaid.enable	**local.metermaid.enable**	This setting must be set to 1 on the system that will run the MeterMaid daemon so that the Watcher will start and control MeterMaid.
metermaid.logfile.*	**logfile.metermaid.***	These settings are the same as those used by imap, pop, and other services. By default MeterMaid writes its log file into *DataRoot*/**log/metermaid**.
metermaid.listenaddr	**metermaid.config.listenaddr**	The address to which MeterMaid should bind. On most systems, the default would not need to be changed, but for multi-homed or HA systems, specifying the appropriate address here is recommended. Default: **(INADDR_ANY)**
metermaid.maxthreads	**metermaid.config.maxthreads**	The MeterMaid server is multithreaded and maintains a pool of threads onto which its tasks are scheduled. This value sets the maximum number of threads that will be used by MeterMaid. On systems with more than 4 CPUs, increasing this value may increase overall throughput. Default: 20

Table 9–1 (Cont.) Options for MeterMaid

Unified Configuration Option	Legacy Configuration Option	Description
metermaid.port	**metermaid.config.port**	This is the port to which MeterMaid listens for connections and to which MeterMaid clients will connect. Default: 63837
metermaid.secret	**metermaid.config.secret**	In order to authenticate incoming connections, MeterMaid uses a shared secret that the clients send once they connect to MeterMaid. No default. Value must be supplied.
metermaid.sslusessl	**service.metermaid.sslusessl**	Requires the use of SSL for all incoming connections to the MeterMaid server. Default: 0

Table 9–2 describes the options used by the **check_metermaid** client:

Table 9–2 check_metermaid Options

Unified Configuration Option	Legacy Configuration Option	Description
metermaid_client.connectfrequency	**metermaid.mtaclient.connectfrequency**	Attempt a connection every *connectfrequency* seconds. When the client needs to connect to MeterMaid, it uses this as an internal throttle to prevent constant connection attempts when MeterMaid isn't available. During the time that the client is unable to communicate with MeterMaid, it will return a "fail" status to the MTA mapping engine indicating that MeterMaid has not blocked this connection. For example, if **check_metermaid.so** attempts to connect to MeterMaid, but it fails for some reason, during the next N seconds as specified by **metermaid_client.connectfrequency** (or **metermaid.mtaclient.connectfrequency** in legacy configuration), no additional attempts will be attempted. It prevents **check_metermaid.so** from trying to connect to MeterMaid too frequently if it is not working. Default: 15
metermaid_client.connecttimeout	**metermaid.mtaclient.connectwait**	When the client is waiting for a connection to MeterMaid (either an initial connection or to reuse another already established connection), it will wait for *connecttimeout* seconds before returning a fail status and allowing this connection to continue. Default: 5
metermaid_client.debug	**metermaid.mtaclient.debug**	If this option is enabled, debugging information from the client will be printed into either the server or thread-specific log file for the SMTP server. Default: no

Table 9–2 (Cont.) check_metermaid Options

Unified Configuration Option	Legacy Configuration Option	Description
metermaid_client.max_conns	**metermaid.mtaclient.maxconns**	In order to support multithreaded servers, the client can maintain a pool of connections to MeterMaid. By doing this, there can be increased concurrency during communications. However, due to internal locking done by MeterMaid, access to a particular table is limited to one request at a time, so multiple connections from a single process may provide limited benefit. Default: 5
metermaid_client.timeout	**metermaid.mtaclient.readwait**	When communicating with MeterMaid, the client will wait *timeout* seconds before returning a fail status and allowing this connection to continue. Default: 10
metermaid_client.server_host	**metermaid.config.serverhost**	This is the host name or IP address to which the clients will connect. It may be the same as **metermaid.listenaddr** but will most likely have a particular value to direct clients to one system in particular in the Messaging Server environment. No default. Value must be supplied.
metermaid_client.sslusessl	**metermaid.mtaclient.sslusessl**	Enables SSL communication with an SSL-enabled MeterMaid server. Default: 0

Lastly, the throttling tables are also defined in Unified and legacy configurations. Table 9–3 describes the options defining the throttling tables. The * in each configuration option is the name of the particular table being defined. For example, for a table called internal, the first option would be called **metermaid.local_table:internal.data_type** in Unified Configuration or **metermaid.table.internal.data_type** in legacy configuration.

Table 9–3 Options Defining Throttling Tables

Unified Configuration Option	Legacy Configuration Option	Valid for Table Types	Description
metermaid.local_ table:*.data_type	**metermaid.table.* .data_type**	throttle simple greylisting	MeterMaid can support two kinds of data in its tables, string and ipv4. String data is limited to 255 bytes per entry and can be compared using case-sensitive or case-insensitive functions (see **metermaid.local_ table:*.options** below). Default: string
metermaid.local_ table:*.max_entries	**metermaid.table.* .max_entries**	throttle simple greylisting	When MeterMaid initializes each table, it pre-allocates this many entries. MeterMaid automatically recycles old entries, even if they haven't yet expired. When a new connection is received, MeterMaid will reuse the least recently accessed entry. A site should specify a value high enough to cache the connections received during *quota_time*. Default: 1000
metermaid.local_ table:*.options	**metermaid.table.* .options**	throttle	This option is a comma-separated list of keywords that defines behavior or characteristics for the table. Valid keywords are: ■ **nocase** - When working with the data, all comparisons are done using a case-insensitive comparison function. (This option is valid only for string data). ■ **penalize** - After *quota_time* seconds, throttle will normally reset the connection count to 0, but if the **penalize** option is enabled, throttle will decrement the connection count by quota (but not less than 0) so that additional connection attempts will penalize future *quota_time* periods. For example, if *quota* were 5 with a *quota_time* of 60, and the system received 12 connection attempts during the first minute, the first 5 connections would be accepted and the remaining 7 would be declined. After 60 seconds has passed, the number of connections counted against the particular address would be reduced to 7, still keeping it above quota and declining connection attempts. Assuming no additional connection attempts were made, after another 60 second period, the number of connections would be further reduced down to 2, and MeterMaid would permit connection attempts again.
metermaid.local_ table:*.quota	**metermaid.table.* .quota**	throttle	When a connection is received, it is counted against quota. If the number of connections received in *quota_ time* seconds exceeds this value, MeterMaid will decline the connection. (The actual effect on the incoming connection is controlled by the mapping table and could result in additional scrutiny, a delay, or denying the connection.) Default: 100
metermaid.local_ table:*.quota_time	**metermaid.table.* .quota_time**	throttle	This specifies the number of seconds during which connections will be counted against quota. After this many seconds, the number of connections counted against the incoming address will be reduced depending on the type of this table. Default: 60

Table 9–3 (Cont.) Options Defining Throttling Tables

Unified Configuration Option	Legacy Configuration Option	Valid for Table Types	Description
metermaid.local_table:*.storage	**metermaid.table.*.storage**	throttle simple greylisting	MeterMaid can use two different storage methods, hash and splay. The default hash table method is recommended, but under some circumstances a splay tree may provide faster lookups. Default: hash
metermaid.local_table:*.table_type	**metermaid.table.*.type**	NA	MeterMaid supports three table types: ■ throttle (default) - This type of table keeps track of the data, typically IP addresses, and will throttle the incoming connections to quota connections during a period of quota_time seconds. ■ simple - This type of table may be used to store arbitrary data referenced by a key. ■ greylisting - This type of table may be used to provide an anti-spam/anti-virus technique. More information about setting up this type of table can be found in the "Implementing Greylisting by Using MeterMaid" chapter.
metermaid.local_table:*.block_time	**metermaid.table.*.block_time**	greylisting	Specifies the ISO 8601 duration for how long we temporarily reject each delivery attempt based on sender and recipient information. Default: pt5m (5 minutes)
metermaid.local_table:*.resubmit_time	**metermaid.table.*.resubmit_time**	greylisting	Specifies the ISO 8601 duration during which, but after *block_time*, we must receive a subsequent delivery attempt based on the same sender and recipient information previously blocked. This sender and recipient combination is now flagged as permitted. Default: pt4h (4 hours)
metermaid.local_table:*.inactivity_time	**metermaid.table.*.inactivity_time**	greylisting	Specifies the ISO 8601 duration for how long we will continue to accept messages based on the sender and recipient information previously permitted. This permission expires after *inactivity_time* from the last allowed delivery. Default: p7d (7 days)

Table 9–4 describes the options used to enable access to multiple MeterMaid Servers from **check_metermaid.so**.

Table 9–4 Options Used to Enable Access to Multiple MeterMaid Servers from check_metermaid.so

Unified Configuration	Legacy Configuration	Description
metermaid_client.remote_table:table.server_nickname	**metermaid.mtaclient.remote_table.table.server_nickname**	Specifies the nickname for a particular MeterMaid server that is responsible for the referenced table. The nickname is a short keyword, consisting only of letters, numbers, and underscores, that will be used in the **remote_server** option names. No default. Value must be supplied.
metermaid_client.remote_server:nickname.max_conns	**metermaid.mtaclient.remote_server.nickname.max_conns**	Specifies the maximum number of concurrent connections to the MeterMaid server referenced by nickname. Default: 3
metermaid_client.remote_server:nickname.server_host	**metermaid.mtaclient.remote_server.nickname.server_host**	Specifies the host name of the MeterMaid server referenced by nickname. Defaults to the value of **metermaid_client.server_host**.
metermaid_client.remote_server:nickname.server_port	**metermaid.mtaclient.remote_server.nickname.server_port**	Specifies the port number of the MeterMaid server referenced by nickname. Default: 63837
metermaid_client.remote_server:nickname.sslusessl	**metermaid.mtaclient.remote_server.nickname.sslusessl**	Specifies whether or not to use SSL for the connection to the MeterMaid server referenced by nickname. Defaults to the value of **metermaid_client.sslusessl**.

Limit Excessive IP Address Connections Using Metermaid – Example

This example uses MeterMaid to throttle IP addresses at 10 connections per minute. For reference, the equivalent **conn_throttle.so** setup in the mappings table would be as follows:

```
PORT_ACCESS

   *|*|*|*|* $C$|INTERNAL_IP;$3|$Y$E
   *|*|*|*|* $C$[/opt/sun/comms/messaging/lib/conn_throttle.so,throttle,$3,10]\
   $N421$ Connection$ declined$ at$ this$ time$E
   * $YEXTERNAL
```

This **PORT_ACCESS** mapping table implements **conn_throttle.so** to restrict connections to a rate of no more than 10 connections per minute for non-INTERNAL connections.

One fundamental difference between the two technologies is that instead of configuring details such as the rate-limit for throttling directly into the mapping table, MeterMaid uses **msconfig** for these settings, as the following example shows.

On systems running the MeterMaid server:

1. Enable MeterMaid by running the following command:

   ```
   msconfig set metermaid.enable 1
   ```

   or in legacy configuration:

   ```
   configutil -o local.metermaid.enable -v TRUE
   ```

2. Set an authentication password used to verify communications between the client and MeterMaid server:

```
msconfig set metermaid.secret password
```

or in legacy configuration:

```
configutil -o metermaid.config.secret -v password
```

3. Define a throttling table

   MeterMaid's throttling behavior is determined by the use of named throttling tables that define operating characteristics. To define a table that throttles at a rate of 10 connections per minute, set the following options in **msconfig**:

   ```
   set metermaid.local_table:ext_throttle.data_type ipv4
   set metermaid.local_table:ext_throttle.quota 10
   ```

   or the following for legacy configuration using **configutil**:

   ```
   configutil -o metermaid.table.ext_throttle.data_type -v ipv4
   configutil -o metermaid.table.ext_throttle.quota -v 10
   ```

   ext_throttle is the name of the throttling table. **ipv4** is the data type Internet Protocol version 4 address representation. 10 is the quota (connection limit).

4. On the MeterMaid system, start MeterMaid.

   ```
   start-msg metermaid
   ```

5. On systems where the MTA will use MeterMaid to do throttling, specify the MeterMaid host and password.

   These are required:

   ```
   msconfig set metermaid.secret MeterMaid_Password
   msconfig set metermaid_client.server_host name_or_ipaddress_of_MetermaidHost
   ```

   or the following for legacy configuration:

   ```
   configutil -o metermaid.config.secret -v MeterMaid_Password
   configutil -o metermaid.config.serverhost -v name_or_ipaddress_of_
   MetermaidHost
   ```

6. Set up the MeterMaid **PORT_ACCESS** table.

   This table is similar to the equivalent **conn_throttle.so** setup:

   ```
   PORT_ACCESS

   *|*|*|*|* $C$|INTERNAL_IP;$3|$Y$E
   *|*|*|*|* $C$:A$[/opt/sun/comms/messaging/lib/check_metermaid.so,throttle,\
   ext_throttle,$3]$N421$ Connection$ declined$ at$ this$ time$E
   * $YEXTERNAL
   ```

   The first line checks to see if the IP address attempting a connection is internal. If it is, it allows the connection. The second line runs the IP address through MeterMaid and if it has connected too frequently, it declines the connection. The third line allows any other connections through, but flagged as **EXTERNAL**.

   This call to **check_metermaid.so** is very similar to the callout to **conn_throttle.so**. The function in **check_metermaid.so** is the same. **throttle** and its arguments are simply the table name as configured using **metermaid.local_table:***tablename* and the IP address to check (**$3**). Like **conn_throttle.so**, this function returns *success* when the limit (as specified in **metermaid.local_table:ext_throttle.quota**) has been reached. This allows the remainder of the mapping entry line is processed, which

sends a message (421 SMTP code, transient negative completion, *Connection not accepted at this time*) to the remote SMTP client, and tells the Dispatcher to close the connection.

$:A ensures that this line will only be processed when being called from the Dispatcher. Without this, the call to **check_metermaid.so** would also happen in the context of the **tcp_smtp_server** processes which also probes the **PORT_ACCESS** mapping table. This would cause MeterMaid to count each incoming connection twice.

This is the basic configuration to set up MeterMaid as a **conn_throttle.so** replacement. Fore more information, see the discussion on mapping operations in *Messaging Server Reference* for information on these topics.

Configuring check_metermaid.so Clients to Access Multiple MeterMaid Servers

The topics in this section include:

- Considerations for Distributing Load Across Multiple MeterMaid Servers
- Configuring check_metermaid.so to Access Multiple MeterMaid Servers

Considerations for Distributing Load Across Multiple MeterMaid Servers

If you have multiple MeterMaid tables in your deployment, you may be able to improve overall performance by distributing them across multiple MeterMaid servers. The **check_metermaid.so** client supports associations between MeterMaid tables and the servers that are responsible for their respective tables. The client also supports per server options for concurrency and use of SSL.

Configuring check_metermaid.so to Access Multiple MeterMaid Servers

To configure **check_metermaid.so** to access multiple MeterMaid Servers:

1. Define the list of all tables and associate them with nicknames for each server using **msconfig** (Unified Configuration):

   ```
   set metermaid_client.remote_table:table1.server_nickname "alpha"
   set metermaid_client.remote_table:table2.server_nickname "alpha"
   set metermaid_client.remote_table:table3.server_nickname "beta"
   ```

   or using **configutil** (legacy configuration):

   ```
   configutil -o metermaid.mtaclient.remote_table.table1.server_nickname -v
   "alpha"
   configutil -o metermaid.mtaclient.remote_table.table2.server_nickname -v
   "alpha"
   configutil -o metermaid.mtaclient.remote_table.table3.server_nickname -v
   "beta"
   ```

   where *table1*, *table2*, and *table3* are tables defined in your deployment's MeterMaid configuration. The servers' nicknames allow the **check_metermaid.so** client to look for remote servers associated with those nicknames. Since nicknames become part of the configuration option names for the remote server definitions, they must include only letters, numbers, and underscores.

2. Create configuration entries for each server nickname for the host name and port for the server, the maximum number of connections, and whether it uses SSL. Run the following commands if you are using **msconfig** (Unified Configuration):

```
set metermaid_client.remote_server:alpha.max_conns 3
set metermaid_client.remote_server:alpha.server_host "alpha.example.com"
set metermaid_client.remote_server:alpha.server_port 63837
set metermaid_client.remote_server:alpha.sslusessl 0
set metermaid_client.remote_server:beta.max_conns 3
set metermaid_client.remote_server:beta.server_host "beta.example.com"
set metermaid_client.remote_server:beta.server_port 63837
set metermaid_client.remote_server:beta.sslusessl 0
```

   Or run the following commands if you are using **configutil** (legacy configuration):

```
configutil -o metermaid.mtaclient.remote_server.alpha.max_conns -v 3
configutil -o metermaid.mtaclient.remote_server.alpha.server_host -v
"alpha.example.com"
configutil -o metermaid.mtaclient.remote_server.alpha.server_port -v 63837
configutil -o metermaid.mtaclient.remote_server.alpha.sslusessl -v 0
configutil -o metermaid.mtaclient.remote_server.beta.max_conns -v 3
configutil -o metermaid.mtaclient.remote_server.beta.server_host -v
"beta.example.com"
configutil -o metermaid.mtaclient.remote_server.beta.server_port -v 63837
configutil -o metermaid.mtaclient.remote_server.beta.sslusessl -v 0
```

   See "MeterMaid Reference" for descriptions of these options.

3. If you are using Unified Configuration and your configuration is compiled, recompile your configuration. This step is not necessary if you are using legacy configuration:

```
imsimta cnbuild
```

4. Restart components that are using **check_metermaid.so**. You would typically do this by restarting the dispatcher:

```
stop-msg dispatcher
start-msg dispatcher
```

10

Implementing Greylisting by Using MeterMaid

This chapter describes how to implement greylisting in Oracle Communications Messaging Server by using MeterMaid.

About Greylisting

Greylisting is a technique used by some MTAs as a way to reduce the number of undesirable spam messages they receive. Simply put, greylisting initially gives a temporary rejection to all incoming mail the first time it sees it, but then permits it upon subsequent attempts. It works by making the assumption that most spam is sent by spambots, PCs that have been compromised by a virus or trojan software, that act as mass-mailing clients. In order to send out as much spam as possible, these systems will connect to a mail server and attempt to deliver the spam to the recipient. If it should encounter a failure, it is extremely unlikely to retry delivery. Proper MTA clients will reschedule and reattempt delivery upon receiving a temporary failure, thus allowing the greylisting mail server a second chance to permit the message to be received.

Greylisting matches messages by using a combination of the source IP address, the envelope **FROM:** address, and the envelope **TO:** address. By using this triplet, greylisting works before the message body is sent during the SMTP transaction, making the temporary rejections happen in response to the **RCPT TO:** command. When the same triplet is presented to the mail server during a subsequent delivery attempt, the **RCPT TO:** command returns a successful response and the mail server will then accept the message for delivery.

In some setups, greylisting has been seen to reduce the number of spam messages received by 80-90%.

The downside to greylisting, however, is that it introduces an artificial delay to incoming mail from previously unknown triplets. The length of this delay varies depending on the originating MTA, but could range from thirty minutes to several hours. It is now expected by many that e-mail is nearly instantaneous, and greylisting can have a negative impact on customer's expectations. Table 10–1 provides an example of greylisting in action.

Table 10–1 Greylisting at a Glance: Example

Time	Action	SMTP Result	Explanation
9:45	Incoming SMTP transaction from **john@example.com** to **susan@example.com** (local user)	451 4.5.1 Temporary failure - retry later	This is the first time that Messaging Server has seen mail from **john@example.com** going to **susan@example.com** so Messaging Server responds with a temporary rejection.
9:47	Another transaction from **john@example.com** to **susan@example.com**	451 4.5.1 Temporary failure - retry later	Because this attempt happened within a short window after the first attempt, it is also temporarily rejected.
10:15	A bit later, the same transaction is retried	250 OK	Now that a subsequent attempt was made within the resubmit window, Messaging Server consider this combination of sender and recipient permitted.
10:20	Mail from **stephen@example.com** to **susan@example.com**	451 4.5.1 Temporary failure - retry later	This is a different sender, so it is handled independently from the previously permitted combination. This combination would be permitted after the block period has passed.
The next day	A new message from **john@example.com** to **susan@example.com**	250 OK	Since this combination is permitted, it remains valid for an extended period.

MeterMaid supports a greylisting table with additional related tuning options.

Table 10–2 describes the different implementations of greylisting.

Table 10–2 Greylisting Features

Feature	Implementation
Rejects previously unseen triplet	Yes
Continues rejecting for an initial blocking period (to block spambots that automatically retry within a very short period)	Yes
Once accepted, continues to accept messages from that triplet	Yes
Requires subsequent attempt within a specified period of time to register the triplet as permitted	Yes
Allows pre-registration of triplets based on outgoing mail, permitting wildcard source IP address	Yes
Expiration of existing triplets can be extended by recent use	Yes

Greylisting has more functionality than using a throttle table.

Basic Greylisting Implementation

Setting up greylisting with Messaging Server is easy due to MeterMaid's built-in support for greylisting tables. Instead of calling the **throttle** routine in **check_metermaid.so**, you can use the **greylisting** routine which handles the appropriate return values for allowing Messaging Server to block transactions when the routine returns success.

First, the MeterMaid table definition:

```
metermaid.table.greylist.type = greylisting
metermaid.table.greylist.data_type = string
metermaid.table.greylist.max_entries = 50000
metermaid.table.greylist.options = nocase
metermaid.table.greylist.block_time = pt5m
metermaid.table.greylist.resubmit_time = pt4h
metermaid.table.greylist.inactivity_time = p7d
```

Note that the time formats can now be in ISO 8601 duration format.

This table defines a greylisting table with the following characteristics:

- All triplets are rejected during the first 5 minutes, even if multiple attempts occur.
- In order for a triplet to be recognized and permitted, a subsequent attempt must be made after that 5 minute window, but within 4 hours of the initial attempt.
- Once a triplet is permitted, it will remain as permitted in the table for 7 days after its last use.

The corresponding access control mapping table is simpler when using a greylisting table as no special handling by the mapping table is required:

```
ORIG_MAIL_ACCESS

! Check the source IP address, sender, and recipient in MeterMaid's greylist
table.
! If the call to greylisting() returns success, then Messaging Server should
return
!  a temporary rejection. If the call fails, then the greylisting check has passed
! and other access control checks can continue.

  TCP|$@*|$@*|*|$@*|SMTP$@*|MAIL|tcp_local|*|1|* \
$C$[IMTA_LIB:check_metermaid.so,greylisting,greylist,$0|$1|$2]\
$N$X4.5.1|Temporary$ failure$ -$ retry$ later$E
```

(Note that the suggested form of the string being used in a greylisting table is *source-ip* | *env-sender* | *env-recipient*.)

After the MTA has received the **MAIL FROM:** and **RCPT TO:** SMTP commands, it will use the envelope addresses as well as the source IP address in a **greylisting** call to MeterMaid. Using the table configuration above, MeterMaid will determine whether or not this particular transaction should be permitted. If the MTA should send a temporary rejection at this point, the call to **check_metermaid.so** will succeed, and the **$N** part of this entry will be returned indicating a rejection. The **$X4.5.1** flags this rejection as a temporary condition so that a **4xx** SMTP response will be given.

Enhancing Greylisting Functionality

However greylisting is configured, there are additional steps one can take to make further improvements to its functionality. Several possibilities are listed here. They can be used individually or combined together to make more powerful setups.

Preloading the Greylisting Table with Outbound Transactions

Since it is often the case that one can expect to receive mail from addresses to which the local users are already sending, it may be useful to preload such address combinations into the greylist table. Since the future source IP address is not known, MeterMaid supports a special address of * that will match any other supplied address.

To preload the address combinations, one needs to add an entry to the access control mapping table:

```
X-IS_INTERNAL_CHANNEL

  tcp_intranet          $Y
  tcp_submit            $Y
  tcp_auth              $Y
  *                     $N

ORIG_MAIL_ACCESS

! For mail that is coming from a local user and going to an external recipient, we
! can save that user/recipient combination and store it into the greylist table
for
! future permission.

  TCP|$@*|$@*|$@*|$@*|SMTP$@*|MAIL|*|*|tcp_local|*  $C$|X-IS_INTERNAL_CHANNEL;$0|\
$[IMTA_LIB:check_metermaid.so,store,greylist,*|$2|$1,1]
```

(Note that the **store** routine requires a value although it is not used by a greylisting table. Any value may be specified here and is ignored by MeterMaid.)

This mapping table entry first checks to see whether the source channel is considered a channel used by our local users. If the channel is in the list provided by the **X-IS_INTERNAL_CHANNEL** mapping table, then processing continues with the call to the **store** routine of **check_metermaid.so** to store this new combination into the **greylist** table. The combination is stored so that it will match incoming messages from the current recipient going to the current sender, and these messages may come from any source IP address and be permitted.

Matching a Range of IP Addresses

A complication that can occur with greylisting is dealing with remote MTAs that use several different hosts to process deliveries. This can mean that one attempt may occur from 192.168.12.1, but a subsequent attempt may come from a different host like 192.168.12.5. Additional delays may be introduced until an attempt is repeated from a host that had tried it previously.

One way to help address this is to limit IP address matching to the first three octets, allowing more hosts to be considered to be the same source. This would match addresses coming from the same A.B.C.D/24 (class C) subnet. The mapping table setup would be very similar to the examples above, but with a change to the IP address wildcard matching.

```
ORIG_MAIL_ACCESS

! When checking the source IP address, only use the first three octets in the
string
! passed to MeterMaid.

  TCP|$@*|$@*|$D*.$D*.$D*.$@*|$@*|SMTP$@*|MAIL|tcp_local|*|1|* \
$C$[IMTA_LIB:check_metermaid.so,greylisting,greylist,$0.$1.$2|$3|$4]\
$N$X4.5.1|Temporary$ failure$ -$ retry$ later$E
```

Simplifying the Sender Address

Some sender addresses will be more complex than a simple **user@example.com** including such features as subaddresses or VERP (variable envelope return path)

notation. For more information see https://en.wikipedia.org/wiki/Variable_envelope_return_path. It may be useful to help greylisting recognize the base form of the address using some basic canonicalization in order to keep track of the basic, simplified address form. This simplification can be done by using a nested mapping table call out to perform the canonicalization.

```
X-CORRESPONDENT

! Subsidiary mapping for removing any subaddress or VERP style material from
! the local-part of an address.

  $_*$[+=\-]%*@*    $0@$3$Y
  *                 $0$Y

ORIG_MAIL_ACCESS

  TCP|$@*|$@*|*|$@*|SMTP$@*|MAIL|tcp_local|*|l|* \
$C$[IMTA_LIB:check_metermaid.so,greylisting,greylist,$0|$|X-CORRESPONDENT;$1||$2]\
$N$X4.5.1|Temporary$ failure$ -$ retry$ later$E
```

Here, the **X-CORRESPONDENT** table is used to reconstruct the sender address into the simpler **user@example.com** form. The result from this is then used in the call to the **greylisting** function.

Providing an Opt-In Mechanism

> **Note:** This section requires Messaging Server 7 Update 2 or later for the necessary **INCLUDE_SPARES** option.

It may be desirable to allow users to choose whether to have MeterMaid perform greylisting on their incoming mail. This could be especially useful when considering some local mail recipients who may not want to be subject to delays in receiving incoming mail from unknown senders, such as recipients like **sales** or **customer_service**. For these local users, one can set up additional LDAP attributes to be used in conjunction with the existing **ORIG_MAIL_ACCESS** mapping table processing.

For this example, let us assume that one creates a new LDAP attribute **mailUserGreyListOptIn** that will be set to **true** or **false**. Those users who have this attribute set to **true** will have their incoming mail checked with greylisting, while those who have it set for **false** will skip this check and receive their mail immediately.

In order to have the MTA look at this extra attribute, it must be configured into the **option.dat** configuration file.

```
LDAP_SPARE_5=mailUserGreyListOptIn
! Include LDAP_SPARE_5 in ORIG_MAIL_ACCESS probes by setting bit 22 (counting from
0)
! of INCLUDE_SPARES. Bit 22 has the value 4194304.
INCLUDE_SPARES=4194304
```

This will add the value of the user's **mailUserGreyListOptIn** attribute to the probe string used in the **ORIG_MAIL_ACCESS** mapping table.

```
ORIG_MAIL_ACCESS

! This example assumes INCLUDE_SPARES=4194304 is set, so that probes corresponding
! to submissions from remote (tcp_local) senders to local recipients have the
form:
```

```
!
! TCP|host-ip|host-port|source-ip|source-port|SMTP-app-info|MAIL|tcp_local|
! remote-sender-address|l|local-recipient-address|recipient-mailUserGreyListOptIn
!
  TCP|$@*|$@*|*|$@*|SMTP$@*|MAIL|tcp_local|$_*|l|$_*|true \
$C$[IMTA_LIB:check_metermaid.so,greylisting,greylist,$0|$1|$2]\
$N$X4.5.1|Temporary$ failure$ -$ retry$ later$E
```

The key difference in this mapping table entry is the addition of **|true** to the matching string. Since the **INCLUDE_SPARES** option will append the content of the **mailUserGreyListOptIn** attribute to the probe string, this mapping entry can match against only those where the recipient's **mailUserGreyListOptIn** attribute has been set to **true**, thus skipping others who may be opting out of greylisting.

Whitelisting Based on User's Addressbook

The goal of greylisting is to allow mail from remote senders to local recipients once they are known. In most cases, this happens when a transaction presents this combination on a subsequent delivery attempt. It is also possible to use the recipient's LDAP-based address book to check for the sender's address to determine whether to bypass greylisting for an already recognized address. In order to do this, another LDAP attribute must be added to the **ORIG_MAIL_ACCESS** probe string by including these values into the **option.dat** file:

```
LDAP_SPARE_6=psroot
! Include LDAP_SPARE_6 in ORIG_MAIL_ACCESS probes by setting bit 23 (counting from
0)
! of INCLUDE_SPARES. Bit 23 has the value 8388608.
INCLUDE_SPARES=8388608
```

Furthermore, if appropriate, the MTA's **LDAP_PAB_*xyz*** options should be set to the proper values for accessing the PAB LDAP server. (However, the usual **pab.*** (Unified Configuration) or **local.service.pab.*** (legacy configuration) settings are usually adequate, and usually do not need to be overridden for MTA purposes via the MTA-specific **LDAP_PAB_xyz** options.)

```
ORIG_MAIL_ACCESS

!
This example assumes INCLUDE_SPARES=4194304 is set, so that probes corresponding
! to submissions from remote (tcp_local) senders to local recipients have the
form:
!
! TCP|host-ip|host-port|source-ip|source-port|SMTP-app-info|MAIL|tcp_local|
! remote-sender-address|l|local-recipient-address|recipient-psroot

!
! Matches on this line mean that the sender was found in the recipient's address
book.
! "Whitelist" those addresses, bypassing the greylisting check.
!
  TCP|$@*|$@*|*|$@*|SMTP$@*|MAIL|tcp_local|$_*|l|$_*|* \
$C$]pabldap:///$3?piEmail1?sub?(|(piEmail1=$1)(piEmail2=$1)(piEmail3=$1))[$E$Y
!
! Now, for all other senders, do the normal greylisting check.
!
  TCP|$@*|$@*|*|$@*|SMTP$@*|MAIL|tcp_local|$_*|l|$_*|* \
$C$[IMTA_LIB:check_metermaid.so,greylisting,greylist,$0|$1|$2]\
$N$X4.5.1|Temporary$ failure$ -$ retry$ later$E
```

This mapping table example shows an LDAP callout being done to check to see whether the sender is already known to the local recipient. This allows users to add their correspondents to their address book as a way to whitelist those entries, allowing those senders to bypass the greylisting when sending mail to these local recipients.

Combining Functionality: A Complex Example

It is possible to combine many of these elements together into a much more comprehensive setup. This example makes use of the preloading, opt-in, and address book whitelisting features together.

First, the two LDAP attributes must be available to the mapping table probe. They can be added with these options in **option.dat**:

```
LDAP_SPARE_5=mailUserGreyListOptIn
LDAP_SPARE_6=psroot
! Include LDAP_SPARE_5 and LDAP_SPARE_6 in ORIG_MAIL_ACCESS probes by
! setting bits 22 and 23 (counting from 0) of INCLUDE_SPARES; that is,
! INCLUDE_SPARES=12582912=4194304+8388608=(1<<22)+(1<<23)
INCLUDE_SPARES=12582912
```

Then, the **mappings** file excerpt below shows how the above elements may be combined.

```
! Subsidiary mapping for checking incoming port and channel against a
! list of "internal submission" channels.
!
! Probe format is
!    port.channel
!
X-INTERNAL-CHANNELS

  587.tcp_submit    $Y
  25.tcp_auth       $Y
  25.tcp_intranet   $Y

! Subsidiary mapping for removing any subaddress or VERP style
! material from the local-part of an address.
! This mapping also performs LDAP URL style quoting of the retained
! portion of the address.
!
X-CORRESPONDENT

   $_*$[+=\-]%*@*    $=$0@$3$_$Y
   *                 $=$0$_$Y

ORIG_MAIL_ACCESS

! This example assumes INCLUDE_SPARES=12582912 (or some superset of bits) is
! set, so that probes corresponding to submissions from local senders to
! remote recipients have a form of:
!
! TCP|host-ip|host-port|source-ip|source-port|SMTP-app-info|MAIL|source-channel|
! local-sender-address|tcp_local|remote-recipient-address|
! sender-mailUserGreyListOptIn|sender-psroot
!
! while probes corresponding to SMTP MAIL submissions from remote
! (tcp_local) senders to local recipients have the form:
!
```

```
!  TCP|host-ip|host-port|source-ip|source-port|SMTP-app-info|MAIL|tcp_local|
!  remote-sender-address|1|local-recipient-address|
!  recipient-mailUserGreyListOptIn|recipient-psroot
!
! The overall logic includes pre-population of the "greylist" table at (1)
! with *|remote-correspondent|local-user on outgoing messages from local users
! who have opted-in to greylisting (have mailUserGreyListOptIn: true) (0),
! and then checks of incoming messages to local users who want greylisting (2)
! against:
!   (i) the local-user's PAB (3)
!   (ii) the pre-populated entries in the "greylist" table (4) or (5)
!   (iii) the "greylist" table tracking "recent" submission attempts from
!         not-otherwise-known (not pre-populated due to local-user sending
!         to them, nor recognized in local-user's PAB) remote senders (4) or (5)
! Note that (ii) and (iii) are done by one probe to the greylist table, as
! MeterMaid first performs the (ii) check automatically due to the format of
! probe.  This probe the greylist table is either done at (4) (for IPv4
! source IPs) or at (5) (for IPv6 source IPs).
!
! For outgoing messages, from local users to remote correspondents,
! pre-populate the "greylist" table using the "store" entry point with
! *|simplified-quoted-remote-correspondent|local-user
! This is so that replies from that remote-correspondent (from whatever
! source-IP) to that local-user will be accepted.
! The subsidiary mapping table X-INTERNAL-CHANNELS is used to check
! (based on the host-port and source-channel) whether the message is
! one from a local user to a remote correspondent.  The subsidiary
! mapping table X-CORRESPONDENT is used to canonicalize the
! recipient-address.
! (0)
!
  TCP|$@*|*|$@*|$@*|SMTP$@*|MAIL|*|*|tcp_local|*|true|*   \
     $C$|X-INTERNAL-CHANNELS;$0.$1|PREPOPULATE|$|X-CORRESPONDENT;$3||$2
!
! If the message was indeed from a local user who wants grey-listing,
! then the above entry matched and reset the probe to now be:
! PREPOPULATE|quoted-simplified-remote-correspondent|local-user
! Then the entry below pre-populates the greylist table with an
! entry for
! *|quoted-simplified-remote-correspondent|local-user
! (1)
!
  PREPOPULATE|*|*     \
     $C$[IMTA_LIB:check_metermaid.so,store,greylist,*|$0|$1,1]$E
!
! For incoming submission attempts from remote correspondents (tcp_local
! submission attempts):
!   (2) Entry matches remote senders to recipients that
!       want grey-listing (mailUserGreyListOptIn: true).  Entry constructs
!       a new GREYLIST... probe retaining relevant fields, namely:
!       GREYLIST|source-ip|quoted-simplified-sender|recipient|psroot
!       where the quoted-simplified-sender field is processed (simplified
!       and LDAP quoted) using the subsidiary X-CORRESPONDENT mapping table
!   (3) For the recipients who want grey-listing, the probe is now
!       GREYLIST|source-ip|quoted-simplified-sender|recipient|psroot
!       Look up the (simplified) sender address in the
!       recipients PAB. Accept submission if found, fall through otherwise.
!   (4) If the sender address wasn't found at (3), fall-through and now
!       attempt a MeterMaid "greylist" table lookup.  The probe to this
!       "greylist" table will have the form:
```

```
!       source-IP-subnet|quoted-simplified-sender|recipient
!       This entry matches on IPv4 incoming source IPs, and ignores the last
!       eight bits to give an IPv4 subnet.
!       Because the probe has the form A|B|C, and it is a probe to a
!       greylisting entrypoint, MeterMaid will automatically initially attempt
!       a *|B|C probe, only bothering with the A|B|C probe if its initial,
!       automatic probe fails.  Thus any pre-populated, generic source IP
!       entry will match first, prior to MeterMaid attempting a lookup of
!       the specific source IP subnet.  If the specific triad is
!       found in the "greylist" table as being due to be greylisted (rejected
!       temporarily), then the probe "succeeds" -- set a new probe string
!       FIRSTATTEMPT and continue so that (6) will match and the greylist
!       response will be issued.
!       Otherwise, the MeterMaid probe "fails" -- as for the cases
!       where the probe matches a "good" (pre-populated, or resubmitted
!       after block_time) entry.
!   (5) The same as (4), but matching on IPv6 incoming source IPs, ignoring
!       the last 64 bits to give an IPv6 subnet.
!   (6) Issue the temporary rejection when the greylist probe of (4) or (5)
!       "succeeded".
!
! For remote senders (source channel tcp_local) to local recipients with the
! LDAP_SPARE_5 attribute "true", reset the probe to the GREYLIST|...form
! (2)
!
  TCP|$@*|$@*|*|$@*|SMTP$@*|MAIL|tcp_local|$_*|1|$_*|true|*  \
$CGREYLIST|$0|$|X-CORRESPONDENT;$1||$2|$=$3$_
!
! If a recipient wants grey-listing, the probe has been rebuilt to be:
! GREYLIST|ip-source|simplified-sender-address|recipient-address|psroot
! where simplified-sender-address omits any subaddress/VERP-y sorts of fluff
! and has had any LDAP URL required quoting applied, and where psroot has also
! had any LDAP URL required quoting applied.
! So next check whether the simplified-sender-address can be found in
! the recipient-address user's PAB (found under psroot); if the sender is
! found, then accept this message -- this sender is "known".
! (3)
!
  GREYLIST|*|*|*|*   \
$C$]pabldap:///$3?piEmail1?sub?(|(piEmail1=$1)(piEmail2=$1)(piEmail3=$1))[$E$Y
!
! Otherwise, if the sender was not known to this recipient, then fall down
! to the subsequent entry which performs the MeterMaid grey-list check.
!
! Match on IPv4 addresses and probe the greylist table.
! If this sender matches an entry in the greylist table, whether
! pre-populated or due to a recent sending attempt, then let their message in.
! If the greylist table probe says the sender needs greylisting,
! (that is, never seen before, or seen before but only within block_time),
! then continue with the probe changed to "FIRSTATTEMPT" so that it'll fall
! through and match the temporary rejection entry below at (6).
! Otherwise, if the sender was in the greylist table but after block_time,
! then MeterMaid "fails" this probe, so the check ends; the table processing
! "falls-through" and, if no other entry matches, the submission is
! permitted.
! (4)
!
  GREYLIST|$D*.$D*.$D*.$D*|*|*|*      \
$[IMTA_LIB:check_metermaid.so,\
greylisting,greylist,$0.$1.$2|$4|$5]$CFIRSTATTEMPT
```

```
!
! (5)
!
  GREYLIST|$H*:$H*:$H*:$H*:$@H*:$@H*:$@H*:$@H*|*|*|*      \
$[IMTA_LIB:check_metermaid.so,\
greylisting,greylist,$0:$1:$2:$3|$4|$5]$CFIRSTATTEMPT
!
! Must be a "new" sending attempt -- give it a temporary rejection
! (6)
!
  FIRSTATTEMPT     $N$X4.5.1|Temporary$ failure$ -$ retry$ later
```

Mapping Table Notes

Table 10–3 describes some mapping table features that might be unfamiliar to casual users of the MTA's mapping table.

Table 10–3 UPDATE TABLE

Strings	Explanation
`$@`	Disables saving the following wildcard match that will not be needed for right-hand side substitutions. This permits sufficient saved wildcards (of which there can be at most ten) to be available for matching fields of more interest.
`$_`	Specifies "non-greedy" (minimal) matching of the portion of the string; used for the local-part of the sender address prior to the first occurrence of a special character possibly indicating a subaddress or VERP address variation.
`$[+=\-]%`	Matches an occurrence of any one of the specified characters. Note that the hyphen character must be quoted with a backslash character to be interpreted as a literal hyphen character rather than indicating a character range.
`$D*`	Matches only decimal digits; used for parsing IP addresses.
`$H*`	Matches only hexadecimal digits; used for parsing IPv6 addresses.

11

MeterMaid Reference

This chapter contains MeterMaid reference information.

configutil Options

For MeterMaid-specific **configutil** options, search for options containing the string "metermaid" in *Messaging Server Reference*.

Table Types

The possible table types allowed by the **metermaid.table.*.type** options are:

- greylisting Tables
- simple Tables
- throttle Tables

greylisting Tables

Greylisting tables may be used to provide an anti-spam/anti-virus technique. For more information about setting up these tables, see "Implementing Greylisting by Using MeterMaid".

simple Tables

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**. We now use a set of new routines to use data stored in **simple** tables (please refer to the

msconfig metermaid.table:*_type option, or the **configutil metermaid.table.*_type** option). These routines now permit one to store, retrieve, and test arbitrary data stored in MeterMaid's ephemeral data store. Oracle Communications Messaging Server also uses the **greylisting** routine that works with **greylisting** tables.

Table 11–1 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 options are provided to these routines, the routines will simply return FALSE and the shared library callout will fail.

Table 11–1 check_metermaid.so Available Routines

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 options used by the routine, the value returned, if any, to the calling environment, and sample usage.

adjust Routine

This section describes the **adjust** routine.

Description

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

Table 11–2 describes the **adjust** routine options.

Table 11–2 adjust Routine Options

Option	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.

adjust Routine 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*.

Table 11–3 describes the **adjust** routine supported table and value types.

Table 11–3 adjust Routine 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@example.org** will be increased by 35 after the **adjust** is completed. If **fred@example.org** 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@example.org,+35]
```

adjust_and_test Routine

This section describes the **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.

Table 11–4 descries the **adjust_and_test** routine options.

Table 11–4 adjust_and_test Routine Options

Option	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

adjust_and_test Routine 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*.

Table 11–5 describes the **adjust_and_test** routine supported table and value types.

Table 11–5 adjust_and_test Routine 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@example.org** will be decreased by 2 after the **adjust** is completed. If **fred@example.org** 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@example.org,-2,>=20]
```

fetch Routine

This section describes the **fetch** routine.

Description

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

Table 11–6 describes the **fetch** routine options.

Table 11–6 fetch Routine Options

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

fetch Routine 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.

Table 11–7 describes the **fetch** routine supported table and value types.

Table 11–7 fetch Routine Supported Table and Value Types

Table	Value	Supported?
greylisting	-	**X**
simple	**integer**	**X**
simple	**string**	**X**

Table 11–7 (Cont.) fetch Routine Supported Table and Value Types

Table	Value	Supported?
throttle	-	-

Example

The following example retrieves the current score for **fred@example.org** 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@example.org]
```

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 (see "Implementing Greylisting by Using MeterMaid" for details).

Table 11–8 describes the **greylisting** routine options.

Table 11–8 greylisting Routine Options

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

greylisting Routine 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@example.org. 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@example.com|fred@example.org]
```

remove Routine

This section describes the **remove** routine.

Description

The **remove** routine removes an entry from table.

Table 11–9 describes the **remove** routine options.

Table 11–9 remove Routine Options

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

remove Routine 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.

Table 11–10 describes the **remove** routine supported table and value types.

Table 11–10 remove Routine 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@example.org** is no longer needed.

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

store Routine

This section describes the **store** routine.

Description

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

Table 11–11 describes the **store** routine options.

Table 11–11 store Routine Options

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

store Routine 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.

Table 11–12 describes the **store** routine supported table and value types.

Table 11–12 store Routine 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@example.org,quarry.example.org]
```

test Routine

This section describes the **test** routine.

Description

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

Table 11–13 describes the **test** routine options.

Table 11–13 test routine Options

Option	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

test Routine 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.

Table 11–14 describes the test routine supported table and value types.

Table 11–14 test Routine 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@example.org,>5]
```

throttle Routine

This section describes the **throttle** routine.

Description

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

Table 11–15 describes the **throttle** routine options.

Table 11–15 throttle Routine Options

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

throttle Routine 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 *Messaging Server Security Guide*.

Table 11–16 describes the throttle routine supported table and value types.

Table 11–16 throttle Routine Supported Table and Value Types

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

12

Administering Event Notification Service

This chapter describes how to enable the Event Notification Service Publisher (ENS Publisher) and how to administer the Event Notification Service (ENS) in Unified Configuration.

See "Messaging Server Specific Event Notification Service Information" and "Event Notification Service API Reference" for more information on ENS and ENS APIs.

ENS Publisher in Messaging Server

The Event Notification Service (ENS) is the underlying publish-and-subscribe service. ENS acts as a dispatcher used by Communications Suite applications as a central point of collection for certain types of *events* that are of interest to them. Events are changes to the value of one or more properties of a resource. Any application that wants to know when these types of events occur registers with ENS, which identifies events in order and matches notifications with subscriptions. ENS and the ENS publisher are bundled with Oracle Communications Messaging Server.

Configuring the ENS Publisher in Unified Configuration

ENS has the following default behavior:

- ENS is enabled by default. The initial configuration sets the **ens.enable** option to 1.
- No configuration is required to load the ENS publisher because the **ms-internal** instance is automatically loaded and configured. Therefore, you do not need to create a separate **ms-internal** instance.
- If you want to configure options for the pre-loaded **ms-internal** default instance, set them with the **ms-internal** instance name. For example, **notifytarget:ms-internal.***settings*.
- IMAP IDLE now only works using ENS, because the ability to use IMAP IDLE with JMQ has been removed.
- The **notifytarget:ms-internal.enshost** defaults to **base.listenaddr** if it is not set.

Administering Event Notification Service

Administering ENS consists of starting and stopping the service, and changing the options to control the behavior of the ENS publisher.

Starting and Stopping ENS

If desired, you can use the **start-msg ens** and **stop-msg ens** commands to start and stop the ENS server.

Event Notification Service Configuration Options

The **notifytarget:***target***.*** options control the behavior of the publisher. Use the **msconfig set** command to set these options. For a list of options, see *Messaging Server Reference*.

To enable ENS, make sure that the **ens.enable** option is set to **1**, for example:

```
/opt/sun/comms/messaging64/bin/msconfig set ens.enable 1
/opt/sun/comms/messaging64/bin/msconfig show ens.enable
role.ens.enable = 1
```

ENS SSL Support

ENS supports SSL in a separate default port 8997. Use the following configuration options to manage ENS SSL support.

To enable or disable SSL support for ENS:

- **ens.enablesslport** (Unified Configuration)
- **local.ens.enablesslport** (legacy configuration)

To change the ENS **sslport**:

- **ens.sslport** (Unified Configuration)
- **local.ens.sslport** (legacy configuration)

To add **sslnicknames**:

- **ens.sslnicknames**(Unified Configuartion)
- **local.ens.sslnicknames** (legacy configuration)

To make a notification target to use TLS/SSL:

- **notifytarget:***target-name***.ensusessl** (Unified Configuration)
- **local.store.notifyplugin.***target-name***.ensusessl** (legacy configuration)

Both SSL and non-SSL ports can be enabled for ENS at the same time. The ENS notification targets can use TLS/SSL for it's communication with the ENS broker.

The notification targets of type, ENS, can use TLS/SSL to communicate with the ENS broker specified.

The default value of the option **notifytarget:***target-name***.ensusessl** will be 1, if **ens.enablesslport** is 1 and one of the following conditions is satisfied:

- The **notifytarget** is the **ms-internal** plugin
- The value of the **notifytarget:***target-name***.enshost**is not set
- The value of **notifytarget:***target-name***.enshost**is equal to the value of **service.listenaddr**
- The value of **notifytarget:***target-name***.enshost** is the loopback address, "127.0.0.1" or "::1"

If the **ensusessl** option of the **notifytarget** is set, then the TLS/SSL will be used to communicate with the host defined by the options **ensHost** and **ensPort**, in the **notifytarget** plugin.

The default value of the option:

- **notifytarget:***target-name***.ensport** (Unified Configuration)
- **local.store.notifyplugin.***target-name***.ensport** (legacy configuration)

will be equal to the value of **ens.sslport** if the value of **notifytarget:***target-name*.**ensusessl** is 1. Otherwise, it will be equal to the value of **ens.port**.

ENS Support for Password Based Authentication

Use the following configuration options to support password-based authentication to the ENS server.

1. Option to Enable/Disable authentication (**ens.mustauthenticate**).
2. Option to change the secret for authentication (**ens.secret**).
3. Option to specify username for the ENS **notifyplugin** (**notifytarget:***target-name***.ensuser**)
4. Option to specify password for the ENS **notifyplugin** (**notifytarget:***target-name***.enspwd**)

Password based authentication of the ENS clients to the broker is enabled, by setting the option, **mustauthenticate**.

The option:

local.ens.mustauthenticate (legacy configuration)

or

ens.mustauthenticate (Unified Configuration)

enables or disables whether authentication is required by the ENS broker. The default value of the **ens.mustauthenticate** option is 0.

If **mustauthenticate** option is set, authentication is required by the ENS broker in both SSL and non-SSL ports.

The ENS broker accepts any user name but the password for authentication is set by the option:

local.ens.secret (legacy configuration)

or

ens.secret (Unified Configuration)

There is no default value for **ens.secret**. If **mustauthenticate** is set, authentication is required by the ENS Broker on both SSL and non-SSL ports. The password for authentication can be set with the **ens.secret**option. All connections to the ENS Broker will fail unless the **ens.secret** is set with a password.

The notification targets of type ENS can be made to use password based authentication.

The userid for authentication is set using the option:

local.store.notifyplugin.*target-name***.ensuser** (legacy configuration)

or

notifytarget:*target-name***.ensuser** (Unified Configuration)

The default value of the option, **ensuser** is "guest."

The password for authentication to ENS broker in a notification target is set using the option:

local.store.notifyplugin.*target-name***.enspwd** (legacy configuration)

or

notifytarget:*target-name***.enspwd** (Unified Configuration)

There is no default value for **enspwd**. The value of the option **enspwd** will be equal to the value of option **ens.secret**, if one of the following conditions satisfies:

1. The notification is the **ms-internal** plugin.
2. The value of **notifytarget:***target-name***.enshost** is NULL.
3. The value of **notifytarget:***target-name***.enshost** is equal to the value of **service.listenaddr**
4. The value of **notifytarget:***target-name***.enshost** is the loopback address, "127.0.0.1" or "::1".

If the **ensuser** and **enspwd** are provided, then the **notifytarget** figures out whether the ENS broker that it connects to require password based authentication or not. If the ENS broker that the notify target connects requires a password, then the password provided will be used or else it won't be used.

With the newer version of the ENS broker that uses authentication with **ens.mustauthenticate** set to 1, you must set a password using the **ens.secret** option. Otherwise all connections to the ENS broker will fail. If authentication is disabled with ens.mustauthenticate set to 0, the older version of the ENS broker which does not have authentication will be used. By default, authentication is disabled.

Note: If you use the older ENS Client APIs with the newer ENS broker (i.e. authentication enabled), it will not work. When authentication is enabled, using ENS will require setting of the option, **local.ens.secret** and use of newer API, **ens_sopen**.

13

Messaging Server Specific Event Notification Service Information

This chapter describes the Oracle Communications Messaging Server specific items that are necessary to use the Event Notification Service (ENS) APIs.

Event Notification Types and Options

For Messaging Server, there is only one event reference, which can be composed of several options. There are various types of event notifications. Table 13–1 lists the event types supported by Messaging Server and gives a description of each. Event notifications are also generated when a user creates, deletes, or renames a folder.

Event Types

Table 13–1 Event Types

Event Types	Description
AnnotateMsg	Shows when annotations or notes are added to a message or deleted from a message.
ChangeFlag	Shows change status as "1" add, "2" remove, or "3" replace.
Copy	Copies one or more messages from one mailbox to another mailbox. If the **CopyMsg** event is not set, **UpdateMsgs** event is triggered when messages are copied.
DeleteMsg	When a message is deleted by a user, IMAP client of the user flags this message with **\Deleted**. It means, IMAP has moved the message to the trash folder.
ExpungeMsg	Messages are deleted permanently from the mailbox which were flagged with **\Deleted** by using **DeleteMsg** event.
Login	User logged in from IMAP, HTTP, or POP.
Logout	User logged out from IMAP, HTTP, or POP.
MsgFlags	Shows when flags on a message are changed. For example, when a read message is flagged as unread.
NewMsg	New message was received by the system into the user's mailbox. Can have a payload of message headers and body.
OverQuota	Operation failed because the user's mailbox exceeded one of the quotas (diskquota, msgquota). The MTA channel holds the message until the quota changes or the user's mail box count goes below the quota. If the message expires while it is being held by the MTA, it will be expunged.

Table 13–1 (Cont.) Event Types

Event Types	Description
PurgeMsg	Message expunged (as a result of an expired date) from the mailbox by the server process imexpire. This is a server side expunge, whereas DeleteMsg is a client side expunge. This is not a purge in the true sense of the word.
ReadMsg	Message in the mailbox was read (in the IMAP protocol, the message was marked Seen).
TrashMsg	Message was marked for deletion by IMAP or HTTP. The user may still see the message in the folder, depending on the mail client's configuration. The messages are to be removed from the folder when an expunge is performed.
UnderQuota	Quota went back to normal from OverQuota state.
UpdateMsg	Message was appended to the mailbox (other than by NewMsg). for example, the user copied an email message to the mailbox. Can have a payload of message headers and body.

The following applies to the above supported event types:

- For **NewMsg** and **UpdateMsg**, message pay load is turned off by default to prevent overloading ENS. See "Payload" for information on how to enable the payload. No other event types support a payload.
- Event notifications can be generated for changes to the **INBOX** alone, or to the **INBOX** and all other folders. The following configuration variable allows for **INBOX** only (value = **0**), or for both the **INBOX** and all other folders (value = **1**):

  ```
  local.store.notifyplugin.noneInbox.enable
  ```

  The default setting is for **INBOX** only (value = **0**).

  Note: There is no mechanism to select folders; all folders are included when the variable is enabled (value = **1**).

- The **NewMsg** notification is issued only after the message is deposited in the user mailbox (as opposed to "after it was accepted by the server and queued in the message queue").
- Every notification carries several pieces of information (called options) depending on the event type, for example, **NewMsg** indicates the IMAP **uid** of the new message. See "Available Options for Each Event Type" for details on the options each event type takes
- Events are not generated for POP3 client access.
- All event types can be suppressed by issuing **XNOTNOTIFY**. For example, an IMAP script used for housekeeping only (the users are not meant to be notified) might issue it to suppress all events.

Options

iBiff uses the following format for the ENS event reference:

```
enp://127.0.0.1/store_?param_=_value&amp;param1_=_value1&amp;param2_=_value2_
```

The event key **enp://127.0.0.1/store** has no significance other than its uniqueness as a string. For example, the hostname portion of the event key has no significance as a hostname. It is simply a string that is part of the URI. However, the event key is user configurable. The list of iBiff event reference options is listed in tables "Mandatory Event Reference Options" and "Optional Event Reference Options" that follow.

The second part of the event reference consists of option-value pairs. This part of the event reference is separated from the event key by a question mark (**?**). The option and value are separated by an equals sign (**=**). The option-value pairs are separated by an ampersand (**&**). Note that there can be empty values, for which the value simply does not exist.

Mandatory Event Reference Options

Table 13–2 describes the mandatory event reference options that must be included in every notification.

Table 13–2 Mandatory Event Reference Options

Option	Data Type	Description
evtType	string	Specifies the event type.
hostname	string	The hostname of the machine that generated the event.
mailboxName	string	Specifies the mailbox name in the message store. The mailboxName has the format **uid@domain**, where **uid** is the user's unique identifier, and **domain** is the domain the user belongs to. The **@domain** portion is added only when the user does not belong to the default domain (i.e. the user is in a hosted domain).
pid	integer	ID of the process that generated the event.
process	string	Specifies the name of the process that generated the event.
timestamp	64-bit integer	Specifies the number of milliseconds since the epoch (midnight GMT, January 1, 1970).

Optional Event Reference Options

Table 13–3 describes optional event reference options, which might be seen in the event depending on the event type (see "Available Options for Each Event Type" for more information.)

Table 13–3 Optional Event Reference Options

Option	Data Type	Description
attrn	string	Specifies an attribute of the *n*th annotation. This attribute can be either **value.shared** or **value.priv**.
authid	string	Specifies the original user name passed by a client.
ctx	integer	Specifies the context within a process which generates an event.
client	IP address	The IP address of the client logging in or out.
diskquota	signed 32-bit integer	Specifies the disk space quota in kilobytes. The value is set to -1 to indicate no quotas.
diskquotaused	signed 64-bit integer	Specifies the volume of disk space in kilobytes that is being used by a user associated with the event.
entryn	string	Specifies the entry of the *n*th annotation. For example, **/comment**.
frommailboxName	string	Specifies the name of the mailbox from which messages were copied.

Table 13–3 (Cont.) Optional Event Reference Options

Option	Data Type	Description
fromUidList	string	Specifies the list of UIDs as a comma separated list. This list shows messages that were copied from the original mailbox.
fromuidValidity	unsigned 32-bit integer	Specifies **uidValidty** from the original mailbox.
hdrLen	unsigned 32-bit integer	Specifies the size of the message header. Note that this might not be the size of the header in the payload, because it might have been truncated.
imapUid	unsigned 32-bit integer	Specifies the IMAP uid option.
identifier	string	Specifies the identifier in the **Set ACL** command for setting rights to access mailboxes.
internaldate	64-bit integer	Specifies the date when a message arrives at the store. **Note**: The time is in milliseconds since the epoch. For example, midnight GMT, January 1, 1970.
mechanism	string	Specifies the authorization type or action performed depending on the event occurred that is Login or Logout.
modseq_sec	long integer	Specifies internal variables associated with the event.
modseq_usec	long unsigned integer	
msgflags	string	Specifies changed flags on messages.
msgquota	unsigned 32-bit integer	Specifies the message quota of a user.
newflags	string	Sets a new flag after the following operations: ■ **A**: answered flag ■ **F**: flagged flag ■ **D**: deleted flag ■ **S**: seen flag ■ **R**: draft flag
NewName	string	Specifies the name of a mailbox after the rename event.
numDeleted	signed 32-bit integer	Specifies the number of messages in a mailbox with the **/deleted** flag set.
numDeletedn	signed 32-bit integer	Specifies the number of messages that are belonged to type **n** in a mailbox with the **/deleted** flag set.
numMsgs	unsigned 32-bit integer	Specifies the number of total messages in a mailbox.
numMsgsn	signed 32-bit integer	Specifies the number of messages that are belonged to type **n** in a mailbox presently.
numSeen	unsigned 32-bit integer	Specifies the number of messages in a mailbox which are marked as seen or read.
numSeenn	signed 32-bit integer	Specifies the total number of messages in a mailbox which are marked as seen or read for each message type.
numSeenDeleted	signed 32-bit integer	Specifies the number of message in a mailbox which are marked as seen or read and deleted.
numSeenDeletedn	signed 32-bit integer	Specifies the total number of message in a mailbox which are marked as seen (read) and deleted for each message type.
oldflags	string	Specifies flags are set for messages before an operation.

Table 13–3 (Cont.) Optional Event Reference Options

Option	Data Type	Description
operation	integer	Specifies the following flag operations: ■ add flags ■ remove flags ■ replace flags
owner	string	Sets the value to **True** if the **userid** associated with the flag change event is the owner of the mailbox.
peruser_flags	signed 32-bit integer	Specifies an internal representation of the **peruser_flags** attribute.
quotaRoot	string	Specifies a user name, folder name, or a type.
rights	string	Sets ACL rights to access mailboxes.
system_flags	signed 32-bit integer	Specifies internal representation of system flags.
toUidList	string	Specifies the list of UIDs as a comma separated list. The list displays UID messages which are provided in the destination mailbox.
uidList	string	Specifies the UID sequence in the IMAP format. It lists messages of interest.
unchangedsince	64-bit integer	Specifies internal variables associated with an event.
userid	string	Specifies the **Userid** associated with an event.
lastUid	unsigned 32-bit integer	Specifies the last IMAP uid value that was used.
size	unsigned 32-bit integer	Specifies the size of the message. Note that this may not be the size of payload, since the payload is typically a truncated version of the message.
uidValidity	unsigned 32-bit integer	Specifies the IMAP uid validity option.

Note: Subscribers should allow for undocumented options when parsing the event reference. This allows for future compatibility when new options are added.

Available Options for Each Event Type

Table 13–4 shows the options that are available for each event type. For example, to see which options apply to a **TrashMsg** event, look in the column header for **ReadMsg**, **TrashMsg** and then note that these events can use **numDel**, **numMsgs**, **numSeen**, and **userValidity**.

Note: Oracle reserves the right to change **no** to **yes** at any time needed.

Table 13–4 Available Options for Each Event Type

Option	NewMsg,UpdateMsg	ReadMsg,TrashMsg	DeleteMsg,PurgeMsg	MsgFlags	ChangeFlag	Login,Logout	OverQuota,UnderQuota	ExpungeMsg	SetAcl	Create	Delete	Rename	AnnotateMsg	Copy
attrn	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No
authid	No	No	No	No	No	Yes	No	No	No	No	No	No	No	No
ctx	Yes	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes
diskquota	Yes	No	Yes	No	No	No	Yes	No	No	No	Yes	No	No	No
diskquotaused	Yes	No	Yes	No	No	No	Yes	No	No	No	Yes	No	No	No
entryn	No	No	No	No	No	No	No	No	No	No	No	No	Yes	No
frommailboxName	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes
fromUidList	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes
fromuidValidity	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes
fromuidValidity64	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No
identifier	No	No	No	No	No	No	No	No	Yes	No	No	No	No	No
internaldate	Yes	No	No	No	No	No	No	No	No	No	No	No	No	Yes
mechanism	No	No	No	No	No	Yes	No	No	No	No	No	No	No	No
modseq_sec	Yes	No	No	No	No	No	No	Yes	No	No	No	No	Yes	Yes
modseq_usec	Yes	No	No	No	No	No	No	Yes	No	No	No	No	Yes	Yes
msgflags	Yes	No	No	No	Yes	No	No	No	No	No	No	No	No	No
msgquota	No	No	No	No	No	No	Yes	No	No	No	Yes	No	No	No
newflags	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No
NewName	No	No	No	No	No	No	No	No	No	No	No	Yes	No	No
numDeleted	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	Yes
numDeletedn	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	Yes
numMsgs	Yes	Yes	Yes	No	No	No	Yes	Yes	No	No	No	No	No	Yes
numMsgsn	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	Yes
numSeen	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	Yes
numSeenn	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	Yes
numSeenDeleted	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	Yes
numSeenDeletedn	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	Yes
oldflags	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No
operation	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No

Table 13–4 (Cont.) Available Options for Each Event Type

Option	NewMsg,UpdateMsg	ReadMsg,TrashMsg	DeleteMsg,PurgeMsg	MsgFlags	ChangeFlag	Login,Logout	OverQuota,UnderQuota	ExpungeMsg	SetAcl	Create	Delete	Rename	AnnotateMsg	Copy
owner	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No
peruser_flags	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No
quotaRoot	No	No	No	No	No	No	Yes	No	No	No	No	No	No	No
rights	No	No	No	No	No	No	No	No	Yes	No	No	No	No	No
system_flags	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No
toUidList	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes
uidList	No	No	No	No	Yes	No	No	Yes	No	No	No	No	No	No
unchangedsince	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No
userid	No	Yes	No	Yes	Yes	Yes	No	No	No	No	No	No	Yes	No
client	No	No	No	No	No	Yes	No	No	No	No	No	No	No	No
diskQuota	Yes	No	Yes	No	No	No	Yes	No	No	No	Yes	No	No	No
hdrLen	Yes	No	Yes	Yes	No	No	No	No	No	No	No	No	No	No
imapUid	Yes	No	Yes	Yes	No	No	No	No	No	No	No	No	Yes	No
lastUid	No	No	Yes	No	No	No	No	Yes	No	No	No	No	No	No
size	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No
uidValidity	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	Yes	No	Yes
userid	No	Yes	No	Yes	Yes	Yes	No	No	No	No	No	No	Yes	No

Payload

ENS allows a payload for two event types: **NewMsg**, and **UpdateMsg**; the other event types do not carry a payload. The payload portion of these two notifications can contain any of the following data:

- No header or body data (default setting)
- Message header data only
- Message body data only
- Both message header and body data

The amount and type of data sent as the payload of the ENS event is determined by the configuration options found in "Payload Configuration Options".

Payload Configuration Options

Table 13–5 describes the payload configuration options.

Table 13–5 Payload Configuration Options

Configuration Options	Description
local.store.notifyplugin.*.maxbodysize	Specifies the maximum size (in bytes) of the body that will be transmitted with the notification. Syntax: **uint32** Default: **0**
local.store.notifyplugin.*.maxheadersize	Specifies the maximum size (in bytes) of the header that will be transmitted with the notification. Syntax: **uint32** Default: **0**

Note that both options are set to zero as the default so that no header or body data is sent with ENS notifications.

Examples

The following example shows a **NewMsg** event reference (it is actually a single line that is broken up to several lines for readability):

```
enp://127.0.0.1/store?evtType=NewMsg&timestamp=1047488403000&
hostname=eman&process=imta&pid=476&mailboxName=testuser&numMsgs=16
&uidValidity=1046993605&imapUid=62&size=877&hdrLen=814
```

In this example, for the **DeleteMsg** event. Messages marked as deleted by IMAP or HTTP were expunged. The user would not see the message in the folder any more.

```
enp://127.0.0.1/store?evtType=DeleteMsg&timestamp=1047488588000&
hostname=eman&process=imapd&pid=419&mailboxName=testuser&
numMsgs=6&uidValidity=1046993605&imapUid=61&lastUid=62
```

And a third example shows a **ReadMsg** event. Message was marked as Seen by IMAP or HTTP.

```
enp://127.0.0.1/store?evtType=ReadMsg&timestamp=1047488477000&
hostname=eman&process=imapd&pid=419&mailboxName=testuser&
uidValidity=1046993605&numSeen=11&numDel=9&numMsgs=16
```

Implementation Notes

The current implementation does not provide security on events that can be subscribed to. Thus, a user could register for all events, and portions of all other users' mail. Because of this it is strongly recommended that the ENS subscriber be on the "safe" side of the firewall at the very least.

The ENS server supports two options to control TCP access to the ENS server (**service.ens.domainallowed** and **service.ens.domainnotallowed**). These options work the same way as the equivalent options for POP, IMAP, and HTTP. See the discussion on configuring client access to POP, IMAP, and HTTP services in *Messaging Server Security Guide*. These options replace the functionality of the ENS_ACCESS environment variable that was included in the legacy ENS server.

14

Event Notification Service API Reference

This chapter details the ENS API reference.

ENS C API Overview

The ENS C API, *ens.h,* is located in the *MessagingServer_home*/**examples/enssdk/** directory. The **ens_pub.c** sample publisher and **ens_sub.c** sample subscriber demonstrate use of the ENS C API.

Here is the API header (**ens.h**):

```
====
// ens.h -- ENS C client API
//
// Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.

ifndef ENS_HEADER_INCLUDED
define ENS_HEADER_INCLUDED 1

ifdef __cplusplus
extern "C" {
endif

//
// Connecting
//

// an ENS client
struct ensclient_t;
typedef struct ensclient_t ensclient_t;

// callback invoked if the ENS connection dies
// subscriptions are no longer valid but must not be unsubscribed
typedef void (*lost_cnx_cb_t)(void*);

ensclient_t* ens_open(const char* host, int port, lost_cnx_cb_t
lost_cnx_cb, void* lost_cnx_arg);

// automatically cleans up all existing subscription handles
void ens_close(ensclient_t*);


//
// Publishing events
//
```

```
typedef void (*destructor_t)(void*);            // cleanup function

void ens_publish(ensclient_t*, const char* evt, char* body, size_t
bodysz, destructor_t body_delete);


//
// Receiving events
//

// a subscription handle
struct sub_t;
typedef struct sub_t subscription_t;

// handler called when a subscribed event is received
typedef void (*notify_cb_t)(void *rock, char *event, char *body, size_t
body_len);

subscription_t* ens_subscribe(ensclient_t*, const char* evt, notify_cb_t
cb, void* rock);
void ens_unsubscribe(ensclient_t*, subscription_t*);

ifdef __cplusplus
}
endif

endif // ENS_HEADER_INCLUDED
====
```

API Basic Usage

The client calls **ens_open()** to start a connection. If reliability across ENS connection outages is important, the client should provide a lost connection handler callback. The lost connection handler normally marks any client-specific subscription information as invalid, calls **ens_close**, and triggers a task to attempt a reconnect by using **ens_open** (possibly after a delay).

The client calls **ens_close()** on shut down.

The client calls **ens_subscribe()** to subscribe to events and gets a callback when a matching event is received. The client calls **ens_unsubscribe()** to unsubscribe from an event (usually not necessary as a client can just call **ens_close**).

To publish an event, use **ens_publish**. (In general, you do not need to do so and the sample code should be sufficient.)

To build the sample programs, link against the **libens** library, which is normally installed in **/opt/sun/comms/messaging64/lib/libens.so**.

The **ens_sub.c** sample program is helpful to see what events are generated and how the event strings and message payloads are formatted.

Both Oracle Communications Messaging Server publishers (that is, **imapd**) and the ENS server (**enpd**) are designed to drop events if an overload situation occurs.

Client API ens_sopen

The client API, **ens_sopen**, can connect to the ENS broker with authentication.

Customers who want to connect to the ENS broker use the API **ens_open** declared in **ens.h** to create a new client connection to the ENS broker. The API, **ens_open**, does not support authentication and TLS/SSL.

So we added a new API, **ens_sopen**, to create a secure connection to the ENS broker that supports authentication and TLS/SSL. The arguments to the API, **ens_sopen**, include all the arguments to the API, **ens_open**, and also includes arguments to accept an username and a password for authentication. It also includes an argument to specify whether to use TLS/SSL while making an connection to the ENS Broker specified.

The new API declared in *lib***/ens/ens.h** is:

ensclient_t* ens_sopen (**const char* host**, **int port**, **int use_ssl**, **const char* auth_user**, **const char* auth_secret**, **lost_cnx_cb_t lost_cnx_cb**, **void* lost_cnx_arg**);

In order to connect to the ENS brokers that support TLS/SSL and authentication, you must use the new API, **ens_sopen**. The API accepts the arguments **auth_user** and **auth_secret**, but it may or may not use them depending on whether or not the ENS broker it is connecting to requires authentication.

If the connection is made to the SSL port, then the value for the argument, **use_ssl**, must be 1. If the connection is made to the non-SSL port, then the value for the argument, **use_ssl**, must be 0.

API Usage Notes

The ENS C API is presently the recommended API for C-based software that needs to subscribe to Messaging Server events. Use of the Glassfish Message Queue, OpenMQ, or Java Enterprise System Message Queue C API is not recommended.

Event Notification Service Java (JMS) API

The ENS Java API is included with Messaging Server. The Java API conforms to the Java Message Service specification (JMS).

ENS acts as a provider to Java Message Service. Thus, it provides a Java API to ENS. The software consists of the base library plus a demo program.

The ENS-JMS API provides a way for Java programs to consume the messages generated by the Messaging Server. An application using JMS can either use the point-to-point approach (queues) or Publish/Subscribe (topic) approach.

The messaging server generates events and publish them to the ENS broker called "enpd". The ENS broker support the topic approach only and not the queue approach. It supports non-durable subcriptions only. With topics, multiple subscribers can subscribe to a single topic.

Note: The bundled **ens-jms.jar** and **jms.jar** should be on classpath.

Sample ENS-JMS Consumer Program

To use Sample ENS-JMS consumer program, the messaging server should be configured for ENS. See "Administering Event Notification Service" and "Messaging Server Specific Event Notification Service Information" for more information on ENS.

Below is the sample JMS consumer of ENS messages which consumes the messages from the default topic and prints the messages.

```
/**
* A consumer that can connect to a ENS Server and printMessages as JMS messages.
*/
public class JMSConsumer {

String hostname = "localhost";
int port = 7997;
boolean printMessages = true;
private String topicName = "store";

/**
* Constructor for creating a JMS consumer.
*/
JMSConsumer() {
}

/**
* start - starts the JMS consumer which waits to receive a event from the
* messaging server. This waits for the event and prints the message got.
*
* @throws JMSException
*/
public void start() throws JMSException {
/**
* Create a EnsTopicConnFactory which creates a connection to the ENS
* Broker. If you want to use the Same code for a different JMS
* provider, all you have to do is to change the ConnectionFactory to
* that of the corresponding broker.
* eg. TopicConnectionFactory connFactory = new
newcom.sun.messaging.ConnectionFactory();
* can be replaced by
* TopicConnectionFactory connFactory = new
EnsTopicConnectionFactory("ens-conn-factory", hostname, port);
* to use the OpenMQ as a broker to receive JMS Messages.
*/
TopicConnectionFactory connFactory = new
EnsTopicConnectionFactory("ens-conn-factory", hostname, port);
TopicConnection topicConn = connFactory.createTopicConnection();
TopicSession topicSession = topicConn.createTopicSession(false,
Session.AUTO_ACKNOWLEDGE);
Topic topic = topicSession.createTopic(topicName);
TopicSubscriber topicSubscriber = topicSession.createSubscriber(topic);
topicConn.start();
TextMessage message = (TextMessage) topicSubscriber.receive();
if (printMessages) {
printMessage(message);
}
topicSession.close();
topicConn.close();
}

public static void main(String[] args) throws JMSException {
JMSConsumer cons = new JMSConsumer();
cons.start();
}

/**
* printMessage - prints the Contents of the JMS messages received.
*
* @param message
```

```
* @throws JMSException
*/
private void printMessage(TextMessage message) throws JMSException {
system.out.println("----Start of Message------ \n");
System.out.println("JMSCorrelationID:+" + message.getJMSCorrelationID() + "\n");
System.out.println("JMSMessageID:+" + message.getJMSMessageID() + "\n");
System.out.println("JMSType:+" + message.getJMSType() + "\n");
System.out.println("JMSDeliveryMode:+" + message.getJMSDeliveryMode() + "\n");
System.out.println("JMSDestination:+" + message.getJMSDestination() + "\n");
System.out.println("JMSExpiration:+" + message.getJMSExpiration() + "\n");
System.out.println("JMSPriority:+" + message.getJMSPriority() + "\n");
System.out.println("JMSRedelivered:+" + message.getJMSRedelivered() + "\n");
System.out.println("JMSReplyTo:+" + message.getJMSReplyTo() + "\n");
System.out.println("JMSTimestamp:+" + message.getJMSTimestamp() + "\n");
System.out.println("Properties:\n");
// Print the Properties
Enumeration keys = message.getPropertyNames();
while (keys.hasMoreElements()) {
String key = (String) keys.nextElement();
System.out.println(key + "  -> " + message.getStringProperty(key) + "\n");
}
System.out.println("----End of Properties----\n");
System.out.println("Body:+" + message.getBody(String.class) + "\n");
System.out.println("----End of Message------ \n");
System.out.println("received: " + message.getText());
}
```

Sample ENS-JMS Consumer Using Automatic Failover and Properties File

Below is the file that has the mapping from the logical hostname to a list of physical hostnames:

```
bash-3.2$ cat /local/harokias/hostmap.properties

# The mapping from logical mailhost name to a list of physical hostnames
# mailhost1 mapping
mailhost1.hostlist = host1 host2 host3

# mailhost2 mapping
mailhost2.hostlist = host4 host5 host6
```

If automatic failover and properties file are to be used then you should made following change in the sample ENS-JMS consumer program:

```
TopicConnectionFactory connFactory = new
EnsTopicConnectionFactory("ens-conn-factory",
<"mailhost1">,<"local/harokias/hostmap.properties">, port);
```

15

Configuring IMAP IDLE

This chapter describes how to configure IMAP IDLE for Oracle Communications Messaging Server.

Benefits of Using IMAP IDLE

The IMAP IDLE extension to the IMAP specification, defined in RFC 2177, allows an IMAP server to notify the mail client when new messages arrive and other updates take place in a user's mailbox. The IMAP IDLE feature has the following benefits:

- Mail clients do not have to poll the IMAP server for incoming messages. Eliminating client polling reduces the workload on the IMAP server and enhances the server's performance. Client polling is most wasteful when a user receives few or no messages; the client continues to poll at the configured interval, typically every 5 or 10 minutes.
- A mail client displays a new message to the user much closer to the actual time it arrives in the user's mailbox. A change in message status is also displayed in near-realtime. The IMAP server does not have to wait for the next IMAP polling message before it can notify the client of a new or updated mail message. Instead, the IMAP server receives a notification as soon as a new message arrives or a message changes status. The server then notifies the client through the IMAP protocol.

Configuring IMAP IDLE with ENS in Unified Configuration

IMAP IDLE with ENS has the following default behavior:

- ENS is enabled by default. The initial configuration sets the **ens.enable** option to 1.
- Every message store has its own **enpd** server.
- The **imapd** process, store delivery channels, and store utilities report changes to the **enpd** server on the local store.
- Some additional configuration is helpful for improved security, HA, and flag updates, as explained in "To Configure IMAP IDLE with ENS".
- IMAP IDLE does not require that events be aggregated to a single **enpd** server and the IDLE event distribution is more efficient if each store uses its own **enpd** server.

Prerequisites for Configuring IMAP IDLE with ENS

Make sure ENS is enabled by setting the **ens.enable** option to 1:

```
msconfig set ens.enable 1
```

To Configure IMAP IDLE with ENS

1. Configure the **enpd** server to allow (or restrict) connections only from the hosts running the message stores by configuring the **ens.domainallowed** and **ens.domainnotallowed** options as necessary. For example, the following command allows access to the local host only:

   ```
   msconfig set ens.domainallowed enpd:127.0.0.1
   ```

   The following command allows access to the local host and all IP addresses 192.168.0.* except 192.168.0.17:

   ```
   msconfig set ens.domainallowed '"enpd:192.168.0.0/255.255.255.0,127.0.0.1
   EXCEPT 192.168.0.17"'
   ```

   These options work the same way as the equivalent options for POP, IMAP, and HTTP. These options replace the functionality of the **ENS_ACCESS** environment variable that was included in the legacy ENS server.

2. Stop, then restart Messaging Server.

   ```
   cd /opt/sun/comms/messaging/bin
   stop-msg
   start-msg
   ```

3. Verify that the IMAP services now include the IDLE feature. Use telnet to connect to the IMAP host and port.

   ```
   telnet IMAP_hostname port
   ```

   Example:

   ```
   telnet myhost imap
   trying 192.18.01.44 ...
   connected to myhost.example.com
   * OK [CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS
   CHILDREN BINARY UNSELECT SORT LANGUAGE STARTTLS IDLE XSENDER X-NETSCAPE
   XSERVERINFO X-SUN-SORT X-SUN-IMAP X-ANNOTATEMORE AUTH=PLAIN]
   myhost.example.com IMAP4 service (Oracle Communications Messaging Server
   7u5-4.07 64bit (built Mar 21 2012)
   ```

To Disable IMAP IDLE

- To disable IMAP IDLE, set the **ens.enable** option to 0 (default is 1). For example:

  ```
  msconfig set ens.enable 0
  ```

16

Lemonade Profile 1 Support

This chapter describes Oracle Communications Message Server's support for Lemonade Profile 1.

Absent from Messaging Server's support for Lemonade Profile 1 is SMTP BINARYMIME. In addition, Messaging Server's CONDSTORE and ANNOTATE implementations might cause performance issues, so use caution when working with these features. See the appropriate sections in this chapter for details.

Introduction to Lemonade

Lemonade refers to an IETF working group formed to address the requirements of supporting standards-based email in a mobile or other resource-constrained environment. A "resource-constrained" environment is one where any or all of the following might be encountered:

- Low bandwidth, high latency networks
- Intermittent network connectivity
- Scarce power and compute cycles
- Minimizing data usage is a goal

The Lemonade Profile (RFC 4550, http://tools.ietf.org/html/rfc4550) defines a set of IMAP and SMTP extensions that address these constraints. Messaging Server implements most of the extensions defined in RFC 4550 (Lemonade Profile 1) and some of the extensions defined in RFC 5550 (Lemonade Profile 2). This information describes the configurable extensions.

Note: The Lemonade standard is mostly intended for mobile clients. Its goal is to reduce network traffic (both in volume and number of interactions) and to move the CPU load from the client to the server. Clients must have support for Lemonade built into them.

Lemonade Features

Some of the more interesting features of Lemonade include the following:

- Forward a message without download (enabled by CATENATE, URLAUTH, and BURL)
- Quick resync (enabled by CONDSTORE and QRESYNC)
- Persistent sort and search (enabled by CONTEXT)

- Conversion

The following sections describe these features in more detail.

Support for BURL

> **Note:** Refer to the discussion on BURL support for SMTP SUBMIT in *Messaging Server Reference* for details on configuring and using BURL.

Messaging Server supports the BURL command, which extends the SMTP submission profile by adding a new command to fetch submission data from an IMAP server. This permits a mail client to inject content from an IMAP server into the SMTP infrastructure without downloading it to the client and uploading it back to the server. Thus, you could forward an email message without first downloading it to the client.

For more information, see http://www.ietf.org/rfc/rfc4468.txt.

Support is enabled in Messaging Server by the BURL_ACCESS mapping. The mapping receives two different probe strings:

```
port_access-probe-info|channel|uid|
port_access-probe-info|channel|uid|url
```

Here **port_access-probe-info** consists of all the information usually included in a PORT_ACCESS mapping table probe. It will be blank if BURL is being used in a "disconnected" context such as batch SMTP. The **channel** is the current source channel and **uid** is the user's authenticatd UID. The **uid** will be blank if no authentication has been performed. The **$:S** input flags will be set if SASL authentication has been performed and **$:T** will be set if TLS is in use.

The first probe is done when responding to EHLO. In order to offer BURL support the mapping must set **$Y** and optionally provide a space-separated list of supported URL types. The mapping assumes **imap** if no string is returned.

The second probe is performed when a BURL command is actually sent by the submit client. It includes the URL specified in the BURL command. Additionally, **$:|** will be set if the URL contains any vertical bars (which if present could possibly confuse some sorts of access checks). The mapping must set **$Y** for the URL to be accepted for processing. If **$D** is also set the string result of the mapping replaces the originally specified URL.

At an absolute minimum the mapping must verify that a proper type of URL has been specified. Typically only **imap:** URLs should be allowed. Additionally, in the case of "submit" IMAP URLs, a check needs to be made to insure that the URL belongs to the user, that is, the access user in the URL matches the authenticated UID for the submit session. Additionally, it is almost always essential to restrict access to an appropriate set of IMAP servers.

The default BURL settings for Unified Configuration are the following:

```
BURL_ACCESS

  *|tcp_*|%*|                    imap$Y
  *|*@*|imap://*;URLAUTH=submit+$1*:*  $:A$M$Y
```

The SMTP server has to have the ability to log in to the IMAP server as the submit user. The **imap_username** and **imap_password** MTA options are used to accomplish

this. **imap_username** specifies the submit user and defaults to the setting of the **imap.submituser** option if not specified. The **imap_password** option specifies the password which of course much match the value set for the submit user account. The **imap_password** option has no default value.

IMAP URLAUTH Support

Messaging Server supports the URLAUTH extension to IMAP and the IMAP URL Scheme (IMAPURL). This extension provides a means by which an IMAP client can use URLs carrying authorization to access limited message data on the IMAP server. An IMAP server that supports this extension indicates this with a capability name of "URLAUTH."

For more information, see http://www.ietf.org/rfc/rfc4467.txt.

IMAP CATENATE Support

Messaging Server supports the CATENATE extension to IMAP, which extends the APPEND command to allow clients to create messages on the IMAP server that may contain a combination of new data along with parts of (or entire) messages already on the server. Using this extension, the client can catenate parts of an already existing message onto a new message without having to first download the data and then upload it back to the server.

For more information, see http://www.ietf.org/rfc/rfc4469.txt.

IMAP Conditional Store Operation Support

Messaging Server supports IMAP Conditional Store Operations (CONDSTORE). IMAP CONDSTORE enables clients to coordinate changes to a common IMAP mailbox, for example, when multiple users are accessing shared mailboxes. The Conditional Store facility provides a protected update mechanism for message state information that can detect and resolve conflicts between multiple writing mail clients. The Conditional Store facility also allows a client to quickly resynchronize mailbox flag changes.

Warning: Use caution when enabling CONDSTORE with Convergence, as there might be degradation of IMAP performance. Our hope is to fix this problem in a future release. When this happens, we will remove this caution.

For more information, see http://www.ietf.org/rfc/rfc4551.txt.

IMAP ANNOTATE Support

Messaging Server supports the ANNOTATE extension to IMAP, which permits clients and servers to maintain "meta data" for messages, or individual message parts, stored in a mailbox on the server. For example, you could use IMAP ANNOTATE to attach comments and other useful information to a message, or to attach annotations to specific parts of a message, marking them as seen or important, or a comment added.

Warning: Use caution when enabling ANNOTATE, as there might be degradation of IMAP performance. Our hope is to fix this problem in a future release. When this happens, we will remove this caution.

For more information, see http://www.ietf.org/rfc/rfc5257.txt. Of note in this document is Section 3.4, "Access Control," which summarizes access control restrictions, including the new ACL "n" right.

Controlling IMAP CAPABILITIES Vector

When migrating a multi-system deployment from Messaging Server 6.3 to 7, it is important that the systems advertise consistent IMAP extension sets, especially with respect to **CONDSTORE**. During migration you can configure Messaging Server 7 to display the same capability set as the older Messaging Server version. You can also turn on the new features on all back systems simultaneously. This feature is only of real significance with **CONDSTORE** and if you have lemonade-aware clients.

You can also filter the initial capability vector advertised in the IMAP banner in the MMP.

Set the IMAP capability options to **1** to control the **CAPABILITIES** vector. To see a list of these options, run the following command:

```
msconfig
msconfig> help option capability_*
```

You can also refer to *Messaging Server Reference* to see the IMAP capability options.

The default values for most of these options is **1**. The exceptions are **IMAP4** and **CONDSTORE**. The default for **IMAP4** is 0 unless **obsoleteimap** is set, in which case it is **1**. These options only affect whether a particular feature is advertised, except for **imap.capability_condstore** which also enables the feature.

Turning on (or not turning off) a capability does not necessarily mean that the feature will be advertised. **IDLE**, **STARTTLS**, and **XREFRESH** are only advertised if enabled by these options and other condition exist that make it appropriate for them to be advertised.

Support for SMTP Submission Service Extension for Future Message Release

Messaging Server supports Lemonade Profile 1, which is an extension to the SMTP submission protocol for a client to indicate a future time for the message to be released for delivery. This extension permits a client to use server-based storage for a message that should be held in queue until an appointed time in the future. This is useful for clients that do not have local storage or are otherwise unable to release a message for delivery at an appointed time. This functionality is useful for sending announcements to be read at the beginning of a work day, to send birthday greetings a day or so ahead, or to use as a lightweight facility to build a personal reminder service.

For more information, see http://tools.ietf.org/rfc/rfc4865.txt.

Support is enabled in Messaging Server by placing the **futurerelease** channel option on the source channel used for initial message submission. The option takes a single integer argument: the maximum number of seconds a message can be held.

17

Managing Logging

This chapter provides overview information on the logging facilities for the Oracle Communications Messaging Server MTA, the Message Store, and services. It also provides procedures for how to manage these logging facilities.

Overview of Logging

This section contains the following subsections:

What Is Logging and How Do You Use it?

Logging is the means by which a system provides you with time-stamped and labeled information about the system's services. Logging provides both a current snapshot of the system as well as a historical view.

By understanding and using Messaging Server log files, you can:

- Gather message statistics, such as message size, rate of message delivery, and how many messages are passing through the MTA
- Perform trend determination
- Correlate capacity planning
- Troubleshoot problems

For example, if your site needs to add more disk storage due to an increase in the number of users, you can use the Messaging Server log files to see what percentage your system demand has increased by and plan for the amount of new disk storage you need.

You can also use Messaging Server logs to understand what your messaging pattern looks like across one day. Understanding when your daily peak loads occur helps you conduct capacity planning.

Logging is also helpful for troubleshooting user problems. For example, if a user isn't receiving expected mail messages, you can use the Messaging Server logging facilities to trace the user's mail messages. In so doing, you might find out that the messages didn't arrive because they were automatically filtered and sent to a SPAM folder.

Types of Logging Data

In general, logging provides you with two types of information:

- Operational data
- Error conditions, also known as event logging

For the most part, Messaging Server logging provides operational data. This operational data contains information such as: the date and time a message entered the system; the sender and recipient of the message; when the message was written to disk; and at a later point in time, when the message was removed from disk and inserted into user's mailbox.

However, Messaging Server logging does also provide some event logging data. To obtain event logging data, you must pull together multiple items from different log files. You could then use a unique constant, such as message ID, to search and correlate the life cycle of a message as it passed from point to point through the system.

Types of Messaging Server Log Files

Messaging Server logging consists of three types of log files:

1. *MTA logs.* These logs provide operational data previously described for the Message Transfer Agent.
2. *Error logs.* These are the MTA debug logs, and the MTA subcomponent logs (that is, job controller, dispatcher and so on).
3. *Message Store and Service logs.* These logs provide messages from the http server, mshttpd, imap, and pop services, as well as the Admin service. The format of these logs differs from that of the first two types of logs.

Table 17–1 lists the different types of log files. By default, log files are located in the *DataRoot*/**log** directory. You can customize and view each type of log file individually.

Table 17–1 Messaging Server Log Files

Type of Log File	Log File Description	Default Name
Message Transfer Agent	Show information about message traffic through the MTA including date and time information, enqueue and dequeue information, and so on.	**mail.log_current**, **mail.log_yesterday**, **mail.log**
Connections	Contains remote machines (MTAs) that connect to this system to send email.	**connection.log**
Counters	Contains message trends in terms of messages sent and received on a per channel basis.	**counters**
Job Controller	Contains data on the master, job controller, sender, and dequeue channel programs.	**job_controller.log**
Dispatcher	Contains errors pertaining to the dispatcher. Turning on dispatcher debugging will increase the information.	**dispatcher.log**

Table 17–1 (Cont.) Messaging Server Log Files

Type of Log File	Log File Description	Default Name
Channel	Records errors pertaining to the channel. Channel options **master_debug** and **slave_debug** turn on channel debugging, which increases the verbosity of the channel log files. Level and type of information is controlled with the various *_DEBUG MTA options.	*channel-name*_master.log* (example: **tcp_local_master.log***)*channel-name*_slave.log* (example: **tcp_local_slave.log***)
IMAP	Contains logged events related to IMAP4 activity of this server	**imap**, **imap.***sequenceNum.timeStamp*
POP	Contains logged events related to POP3 activity of this server	**pop**, **pop.***sequenceNum.timeStamp*
HTTP	Contains logged events related to HTTP activity of this server	**http**, **http.***sequenceNum.timeStamp*
Default	Contains logged events related to other activity of this server, such as command-line utilities and other processes	**default**, **default.***sequenceNum.timeStamp*
transactlog	Contains machine readable (XML format) information about actions performed by the message store. For more details on the Store Transaction Log Format, see *Reference Guide*.	**transactlog**
watcher	Monitors process failures and unresponsive services (see Table 3–5, " Services Monitored by watcher and msprobe") and will log error messages indicating specific failures.	**watcher**

where:

sequenceNum - Specifies an integer that specifies the order of creation of this log file compared to others in the log-file directory. Log files with higher sequence numbers are more recent than those with lower numbers. Sequence numbers do not roll over. They increase monotonically for the life of the server (beginning at server installation).

timeStamp - Specifies a large integer that specifies the date and time of file creation. (Its value is expressed in standard UNIX time: the number of seconds since midnight January 1, 1970.)

For example, a log file named **imap.63.915107696** would be the 63rd log file created in the directory of IMAP log files, created at 12:34:56 PM on December 31, 1998.

The combination of open-ended sequence numbering with a timestamp gives you more flexibility in rotating, expiring, and selecting files for analyzing. See "Defining and Setting Service Logging Options" for more specific suggestions.

Tools for Managing Logging

You can customize the policies for creating and managing Messaging Server log files by using the **msconfig** command.

For Message Store, the settings you specify affect which and how many events are logged. You can use those settings and other characteristics to refine searches for logged events when you are analyzing log files.

The MTA uses a separate logging facility you configure MTA logging by specifying information in configuration files.

For log analysis and report generation beyond the capabilities of Messaging Server, you must use other tools. You can manipulate log files on your own with text editors or standard system tools.

With a scriptable text editor supporting regular-expression parsing, you can potentially search for and extract log entries based on any of the criteria discussed in this information, and possibly sort the results or even generate sums or other statistics.

In UNIX environments you might also be able to modify and use existing report-generation tools that were developed to manipulate UNIX **syslog** files. If you want to use a public-domain **syslog** manipulation tool, remember that you might need to modify it to account for the different date/time format and for the two extra components (*facility* and *logLevel*) that appear in Messaging Server log entries but not in **syslog** entries.

Tracking a Message Across the Various Log Files

The following describes how a message flows through the system, and at what point information gets written to the various log files. This description is meant to aid you in your understanding of how to use Message Server's log files to troubleshoot and resolve problems. See Figure 6–2, "MTA Architecture" to follow along.

1. A remote host makes a connection to the TCP socket on your messaging host, requesting SMTP service.
2. The MTA dispatcher responds to the request, and hands off the connection to your messaging host's SMTP service. As the MTA is modular in design, it consists of a set of processes, including the job controller and the SMTP service dispatcher. The dispatcher takes the incoming TCP connection and sends it to the SMTP service. The SMTP service writes the message to disk to a channel area. The SMTP service understands the message's envelope options, such as sender and recipient. Configuration entries in the system tell what destination channel it belongs to.
3. The dispatcher writes to the **dispatcher.log** file that it forked a thread and made the thread available to incoming connection from a certain IP address.
4. The SMTP server writes to its **tcp_smtp_server.log** file, recording the dialog of what happens when the remote host connected to it and sent a message. This log file gets created when dispatcher hands off to SMTP server on the host's IP.
5. The SMTP server writes the message to a queue area on disk for a channel program such as **tcp_intranet**, and informs the job controller.
6. The job controller contacts the channel program.
7. The channel program delivers the message. Each channel has its own log file. However, these logs usually show the starting and stopping of the channel. To get more information, you must enable debug level for the channel. However, as this can slow down your system and actually make problems more obscure if left on, you should only enable debug level when an actual problem is occurring.

Note: For efficiency, if a channel is already running for an existing process, and a new message comes in, the system does not spawn a new channel process. The currently running process picks up the new message.

---

8. The message is delivered to its next hop, which could be another host, another TCP connection, and so forth. This information is written to the **connection.log** file when **SEPARATE_CONNECTION_LOG** is enabled. At the same time that the SMTP server writes the message to a queue area on disk, the channel responsible for the message writes a record in the **mail.log_current** file. The record shows such information as the date and time the message was enqueued, the sender, the recipient, so forth. See "MTA Message Logging Examples" for more information. The most useful file for tracing the message is the **mail.log_current** file.

Managing MTA Message and Connection Logs

The MTA provides facilities for logging each message as it is enqueued and dequeued. It also provides dispatcher error and debugging output.

You can control logging on a per-channel basis or you can specify that message activity on all channels be logged. In the initial configuration, logging is disabled on all channels.

See "Enabling MTA Logging" for more information.

Enabling logging causes the MTA to write an entry to the *DataRoot*/**log/mail.log_current** file each time a message passes through an MTA channel. Such log entries can be useful for gathering statistics on how many messages are passing through the MTA (or through particular channels). You can also use these log entries to investigate other issues, such as whether and when a message was sent or delivered.

The message return job, which runs every night around midnight, appends any existing **mail.log_yesterday** to the cumulative log file, **mail.log**, renames the current **mail.log_current** file to **mail.log_yesterday**, and then begins a new **mail.log_current** file. The message return job also performs the analogous operations for any **connection.log*** files.

While the MTA performs automatic rollovers to maintain the current file, you must manage the cumulative **mail.log** file by determining policies for tasks such as backing up the file, truncating the file, deleting the file, and so on.

When considering how to manage the log files, note that the MTA periodic return job will execute a site-supplied *DataRoot*/**site-programs/bin/daily_cleanup** script, if one exists. Thus some sites might choose to supply their own cleanup procedure that, for instance, renames the old **mail.log** file once a week (or once a month), and so on.

Note: With logging is enabled, the **mail.log** file steadily grows and, if left unchecked, consumes all available disk space. Monitor the size of this file and periodically delete unnecessary contents. You can also delete the entire file as another version will be created as needed.

Understanding the MTA Log Entry Format

The MTA log file is written as ASCII text. By default, each log file entry contains eight or nine fields as shown in the example below.

```
16-Feb-2007 14:54:13.72 tcp_local ims-ms EE 1 adam@example.com
rfc822;marlowe@example.org marlowe@ims-ms-daemon
```

The log entry shows:

1. The date and time the entry was made (in the example, 16-Feb-2007 14:54:13.72).
2. The channel name for the source channel (in the example, **tcp_local**).

3. The channel name for the destination channel (in the example, **ims-ms**). For SMTP channels, when **LOG_CONNECTION** is enabled, a plus (+) indicates inbound to the SMTP server; a minus (-) indicates outbound via the SMTP client.
4. The type of entry (in the example, *EE*). Entries can consist of a single action code (see Table 17–2, " Logging Entry Action Codes") or an action code and one or more modifier codes (see Table 17–3, " Logging Entry Modifier Codes"). The format for entries is as follows

   :<*action_code*><*zero or more optional modifiers*>

   For example a logging entry code of **EEC** means that the email was Enqueued (action-code **E**) using ESMTP (modifier **E**) and SMTP Chunking (modifier **C**). Please refer to the tables below for details on the currently used action and modifier codes.
5. The size of the message (in the example, *1*). This is expressed in kilobytes by default, although this default can be changed by using the **BLOCK_SIZE** MTA option. The SMS channel can be configured to log a page count rather than file size in this field. See Table 21–5, " SMS Channel Options" for information on **LOG_PAGE_COUNT**.
6. The envelope **From:** address (in the example, **adam@example.com**). Note that for messages with an empty envelope **From:** address, such as notification messages, this field is blank.
7. The original form of the envelope *To:* address (in the example, *marlowe@example.org*).
8. The active (current) form of the envelope **To:** address (in the example, **marlowe@ims-ms-daemon**).
9. The delivery status (SMTP channels only).

Table 17–2 describes the logging entry action codes.

Table 17–2 Logging Entry Action Codes

Entry	Description
B	Bad command sent to the SMTP server. The recipient address field will contain the command that was rejected while the diagnostic field will contain the response the SMTP server gave. MTA channel option, MAX_B_ENTRIES, controls how many bad commands will be logged in a given session. Default is 10.
D	Successful dequeue
E	Successful enqueue
J	Rejection of attempted enqueue (rejection by slave channel program)
K	Recipient message rejected. If the sender requests **NOTIFY=NEVER** DSN flag set or if the message times out or if the message is manually returned (for example: **imsimta qm** "delete" command always generates a "K" record for each recipient, while a qm "return" command will generate a "K" record rather than an "R" record). This indicates that there was no notification sent to the sender per the sender's own request.This can be compared with "R" records, which are the same sort of rejection/time-out, but where a new notification message (back to the original sender) is also generated regarding this failed message.

Table 17–2 (Cont.) Logging Entry Action Codes

Entry	Description
P	Request to generate a Delivery Status Notification. "P" records are only generated in cases where the error causing the DSN isn't recorded in any other log entry. The various cases where this happens are further detailed by the presence of a modifier character on the action. Currently the defined modifiers are: ■ F - address errors detected during alias or mailing list expansion operations ■ X - capture operations ■ J - journal operations ■ Y - Sieve syntax or evaluation errors ■ D - success delivery receipts
Q	Temporary failure to dequeue
R	Recipient address rejected on attempted dequeue (rejection by master channel program), or generation of a failure/bounce message
S	LMTP deposit into the message store. This action code is used on the LMTP server side.
V	Warning message that will appear whenever a transaction is abnormally aborted. There will be one "V" record per enqueued recipient address.
W	Warning message sent to notify original sender that the message has not been delivered yet, but it is still in the queue being retried.
Z	Some successful recipients, but this recipient was temporarily unsuccessful; the original message file of all recipients was dequeued, and in its place a new message file for this and other unsuccessful recipients will be immediately enqueued

Table 17–3 describes the logging entry modifier codes.

Table 17–3 Logging Entry Modifier Codes

Entry	Description
A	SASL authentication used.
B	SMTP BINARYMIME extension used (RFC 3030).
C	Chunking was used. Note that ESMTP has to be used for chunking to work, so you'll typically see field values like *EEC* or *DEC*.
E	An EHLO command was issued/accepted and therefore ESMTP was used.
L	LMTP was used.
Q	SMTP PIPELINING extension used (RFC 2920).
S	TLS/SSL used. S transaction log entries now increment the various submitted message counters associated with the channel.
U	BURL used (RFC 4468).

If **LOG_CONNECTION** is enabled, then an additional set of action codes is used. See the discussion on LOG_CONNECTION in *Messaging Server Reference*. Table 17–4 describes the LOG_CONNECTION action codes.

Table 17–4 SMTP LOG_CONNECTION Action Codes + or - Entries

Entry	Description
C	Connection closed. A diagnostic field will follow. Written to **connection.log_current (or {{mail.log_current** if a single log file is being used). Used to record the reason why the connection was closed. In particular, if the connection was closed due to some session disconnect limit being reached, that fact will show up in the diagnostics field.
O	Connection opened.
T	**PORT_ACCESS** log entry.
U	Logs SMTP authentication successes and failures. Format is the same as other O and C entries. In particular, the same application and transport information fields appear in same order. The username will be logged in the username field if it is known. Bit 7 (value 128) of the **LOG_CONNECTION** MTA option controls this.
X	Connection rejected.
Y	Connection attempt failed before being established.
I	ETRN command received.

With **LOG_CONNECTION**, **LOG_FILENAME**, **LOG_MESSAGE_ID**, **LOG_NOTARY**, **LOG_PROCESS**, and **LOG_USERNAME** MTA options all enabled, the format becomes as shown in the example below. (The sample log entry line has been wrapped for typographic reasons; the actual log entry would appear on one physical line.)

```
16-Feb-2007 15:04:01.14 2bbe.5.3 tcp_local ims-ms
EE 1 service@example.org rfc822;adam@example.com
adam@ims-ms-daemon 20
/opt/sun/comms/messaging64/data/queue/ims-ms/000/ZZf0r2i0HIaY1.01
&lt;0JDJ00803FAON200@mailstore.example.org> mailsrv
example.org (example.org [192.160.253.66])
```

Where the additional fields, beyond those already discussed above, are:

1. The process ID (expressed in hexadecimal), followed by a period (dot) character and a count. If this had been a multithreaded channel entry (that is, a **tcp_*** channel entry), there would also be a thread ID present between the process ID and the count. In the example, the process ID is 2bbe.5.3.
2. The **NOTARY** (delivery receipt request) flags for the message, expressed as an integer (in the example, 20).
3. The file name in the MTA queue area (in the example, **/opt/sun/comms/messaging64/data/queue/ims-ms/000/ZZf0r2i0HIaY1.01**).
4. The message ID (in the example, **0JDJ00803FAON200@mailstore.example.org**).
5. The name of the executing process (in the example, **mailsrv**). On UNIX, for dispatcher processes such as the SMTP server, this will usually be **mailsrv** (unless SASL was used, in which case it will be the authenticated user name, for example, ***service@example.org**).
6. The connection information (in the example, **example.org (example.org [192.160.253.66])**. The connection information consists of the sending system or channel name, such as the name presented by the sending system on the HELO/EHLO line (for incoming SMTP messages), or the enqueuing channel's official host name (for other sorts of channels). In the case of TCP/IP channels, the sending system's real name, that is, the symbolic name as reported by a DNS

reverse lookup and/or the IP address, can also be reported within parentheses as controlled by the **ident*** channel options. See the discussion on the IDENT Lookups in the *Messaging Server Reference* for an instance of the default **identnone** option, that selects display of both the name found from the DNS and IP address.

Enabling MTA Logging

To gather statistics for just a few particular MTA channels, enable the logging channel option on just those MTA channels of interest. Many sites prefer to enable logging on all MTA channels. In particular, if you are trying to track down problems, the first step in diagnosing some problems is to notice that messages are not going to the channel you expected or intended, and having logging enabled for all channels can help you investigate such problems.

To Enable MTA Logging on a Specific Channel

1. Run the **msconfig edit channels** command.
2. To enable logging for a particular channel, add the **logging** option to the channel definition. For example:

   channel-name option1 option2 **logging**

   In addition, you can also set several configuration options such as directory path for log files, log levels, and so on. See "Managing Message Store, Admin, and Default Service Logs" for more information.

> **Note:** The message return job, which runs every night around midnight, appends any existing **mail.log_yesterday** to the cumulative log file, **mail.log**, renames the current **mail.log_current** file to **mail.log_yesterday**, and then begins a new **mail.log_current**file. It also performs the analogous operations for any **connection.log*** files. It is possible that **mail.log_yesterday** contains time stamps which have already passed over rotation time.

To Enable MTA Logging on All Channels

1. Run the **msconfig edit channels** command.
2. Add the **logging** option to your **defaults** channel configuration file. For example:

```
defaults notices 1 2 4 7 copywarnpost copysendpost postheadonly
noswitchchannel \
immnonurgent maxjobs 7 defaulthost example.org example.org logging

!
! delivery channel to local /var/mail store
l subdirs 20 viaaliasrequired maxjobs 7
mailhost.example.org
```

Specifying Additional MTA Logging Options

In addition to the basic information always provided when logging is enabled, you can specify that additional, optional information fields be included by setting various **LOG_*** MTA options.

To Send MTA Logs to syslog

1. Enter the following command to set **log_messages_syslog** to 1:

```
msconfig set log_messages_syslog 1
```

To write MTA message log file entries to syslog, you must set the **log_messages_ syslog** option to a non-zero value. The absolute value of the non-zero value sets the syslog priority and facility mask. Negative values disable the generation of the regular **mail.log*** entries. Positive values mean that the syslog entries are generated in addition to the regular **mail.log*** entries. 0 is the default and means no syslog or event logging is performed.

Facility and priority numbers are located in the **/usr/include/sys/syslog.h** file.

To Control Formatting of Log Entries

1. Set the **LOG_FORMAT** option by running **msconfig set log_format** *n* where *n* corresponds to one of the settings below
   - 1 (default) the standard format.
   - 2 requests non-null formatting: empty address fields are converted to the string "<>"
   - 3 requests counted formatting: all variable length fields are preceded by **N**, where **N** is a count of the number of characters in the field.
   - 4 causes log entries to be written in an XML-compatible format. Entry log entry appears as a single XML element containing multiple attributes and no sub-elements. Three elements are currently defined, **en** for enqueue/dequeue entries, **co** for connection entries, and **he** for header entries.

     Enqueue/dequeue (**en**) elements can have the following attributes:

```
ts - time stamp (always present)
no - node name (present if LOG_NODE=1)
pi - process id (present if LOG_PROCESS=1)
sc - source channel (always present)
dc - destination channel (always present)
ac - action (always present)
sz - size (always present)
so - source address (always present)
od - original destination address (always present)
de - destination address (always present)
rf - recipient flags (present if LOG_NOTARY=1)
fi - filename (present if LOG_FILENAME=1)
ei - envelope id (present if LOG_ENVELOPE_ID=1)
mi - message id (present if LOG_MESSAGE_ID=1)
us - username (present if LOG_USERNAME=1)
ss - source system (present if bit 0 of LOG_CONNECTION
is set and source system information is available)
se - sensitivity (present if LOG_SENSITIVITY=1)
pr - priority (present if LOG_PRIORITY=1)
in - intermediate address (present if LOG_INTERMEDIATE=1)
ia - initial address (present if bit 0 of LOG_INTERMEDIATE
is set and intermediate address information is available)
fl - filter (present if LOG_FILTER=1 and filter information
is available)
re - reason (present if LOG_REASON=1 and reason string is set)
di - diagnostic (present if diagnostic info available)
tr - transport information (present if bit 5 of LOG_CONNECTION
is set and transport information is available)
ap - application information (present if bit 6 of LOG_CONNECTION
is set and application information is available)
qt - the number of seconds the message has spent in the queue (LOG_QUEUE_
TIME=1)
```

Here is a sample **en** entry:

```
<en ts="2004-12-08T00:40:26.70" pi="0d3730.10.43" sc="tcp_local"
dc="l" ac="E" sz="12" so="info-E8944AE8D033CB92C2241E@whittlesong.com"
od="rfc822;ned+2Bcharsets@mauve.sun.com"
de="ned+charsets@mauve.sun.com" rf="22"
fi="/path/ZZ01LI4XPX0DTM00IKA8.00" ei="01LI4XPQR2EU00IKA8@mauve.sun.com"
mi="&lt;11a3b401c4dd01$7c1c1ee0$1906fad0@elara>" us=""
ss="elara.whittlesong.com ([208.250.6.25])"
in="ned+charsets@mauve.sun.com" ia="ietf-charsets@innosoft.com"
fl="spamfilter1:rvLiXh158xWdQKa9iJ0d7Q==, addheader, keep"/>
```

Note that this entry has been wrapped for clarity; actual log file entries always appear on a single line. Connection (**co**) entries can have the following attributes:

```
ts - time stamp (always present, also used in en entries)
no - node name (present if LOG_NODE=1, also used in en entries)
pi - process id (present if LOG_PROCESS=1, also used in en entries)
sc - source channel (always present, also used in en entries)
dr - direction (always present)
ac - action (always present, also used in en entries)
tr - transport information (always present, also used in en entries)
ap - application information (always present, also used in en entries)
mi - message id (present only if message id info available,
also used in en entries)
us - username (present only if username information available, also
used in en entries)
di - diagnostic (present only if diagnostic information available,
also used in en entries)
ct - the length of the connection, in seconds. (LOG_QUEUE_TIME=1,
also used in en entries)
```

Here is a sample *co* entry:

```
<co ts="2004-12-08T00:38:28.41" pi="1074b3.61.281" sc="tcp_local" dr="+"
ac="O" tr="TCP|209.55.107.55|25|209.55.107.104|33469" ap="SMTP"/>
```

Header (**he**) entries have the following attributes:

```
ts - time stamp (always present, also used in en entries)
no - node name (present if LOG_NODE=1, also used in en entries)
pi - process id (present if LOG_PROCESS=1, also used in en entries)
va - header line value (always present)
```

Here is a sample **he** entry:

```
<he ts="2004-12-08T00:38:31.41" pi="1074b3.61.281" va="Subject: foo"/>
```

To Correlate Log Message Entries

- Set the **LOG_MESSAGE_ID** option to 1 by running **msconfig set log_message_id 1**.

  A value of 0 is the default and indicates that message IDs are not saved in the **mail.log*** files.

To Log Amount of Time Messages Have Spent in the Queue

- Set the **LOG_QUEUE_TIME** option to 1 by running **msconfig set log_queue_time 1**. This option logs the amount of time messages spent in the queue. The queue time is logged as an integer value in seconds. It appears immediately after the

application information string in non-XML format logs. The attribute name in XML formatted logs for this value is **qt**.

To Identify Message Delivery Retries

- Set the **LOG_FILENAME** option to 1 by running **msconfig set log_filename 1** This option makes it easier to immediately spot how many times the delivery of a particular message file has been retried. This option can also be useful in understanding when the MTA does or does not split a message to multiple recipients into separate message file copies on disk.

To Log TCP/IP Connections

- Set the **LOG_CONNECTION** option by running **msconfig set log_connection 1** This option causes the MTA to log TCP/IP connections, as well as message traffic. The connection log entries are written to the **mail.log*** files by default. Optionally, the connection log entries can be written to **connection.log*** files. See the **SEPARATE_CONNECTION_LOG** option for more information.

To Write Entries to the connection.log File

- Set the **SEPARATE_CONNECTION_LOG** option to 1 by running **msconfig set separate_connection_log 1**. Use this option to specify that connection log entries instead be written to **connection.log** files. The default value of 0 causes the connection logging to be stored in the MTA log files.

To Correlate Log Messages by Process ID

- Set the **LOG_PROCESS** option by running **msconfig set log_process 1**. When used in conjunction with **LOG_CONNECTION**, this option enables correlation by process ID of which connection entries correspond to which message entries.

To Save User Names Associated with a Process That Enqueues Mail to the mail.log File

- Set the **LOG_USERNAME** option by running **msconfig set log_username 1**. This option controls whether or not the user name associated with a process that enqueues mail is saved in the **mail.log** file. For SMTP submissions where SASL (SMTP AUTH) is used, the user name field will be the authenticated user name (prefixed with an asterisk character).

MTA Message Logging Examples

The exact field format and list of fields logged in the MTA message files vary according to the logging options set. This section shows a few examples of interpreting typical sorts of log entries.

See "Specifying Additional MTA Logging Options" for a description of additional, optional fields.

> **Note:** For typographic reasons, log file entries will be shown folded onto multiple lines. Actual log file entries are one line per entry.

When reviewing a log file, keep in mind that on a typical system many messages are being handled at once. Typically, the entries relating to a particular message will be interspersed among entries relating to other messages being processed during that same time. The basic logging information is suitable for gathering a sense of the overall numbers of messages moving through the MTA.

If you want to correlate particular entries relating to the same message to the same recipient(s), enable **LOG_MESSAGE_ID**. To correlate particular messages with particular files in the MTA queue area, or to see from the entries how many times a particular not-yet-successfully-dequeued message has had delivery attempted, enable **LOG_FILENAME**. For SMTP messages (handled via a TCP/IP channel), if you want to correlate TCP connections to and from remote systems with the messages sent, enable **LOG_PROCESS** and some level of **LOG_CONNECTION**.

MTA Logging Example: User Sends an Outgoing Message

The following example shows a basic example of the sort of log entries one might see if a local user sends a message out an outgoing TCP/IP channel, for example, to the Internet. In this example, **LOG_CONNECTION** is enabled. The lines marked with (1) and (2) are one entry---they would appear on one physical line in an actual log file. Similarly, the lines marked with (3) - (7) are one entry and would appear on one physical line.

Example MTA Logging: A Local User Sends An Outgoing Message

```
16-Feb-2007 15:41:32.36 tcp_intranet tcp_local EE 1 (1)
adam@example.com rfc822;marlowe@example.org marlowe@example.org (2)
example.org (example.org [192.160.253.66])

16-Feb-2007 15:41:34.73 tcp_local DE 1 (3)
adam@example.com rfc822;marlowe@example.org marlowe@example.org (4)
thor.example.org dns;thor.example.org

(TCP|206.184.139.12|2788|192.160.253.66|25) (5)

(thor.example.org ESMTP Sendmail ready Thu 15 Feb 2007 21:37:29 -0700 [MST]) (6)

smtp;250 2.1.5 &lt;marlowe@example.org>... Receipt ok (7)
```

1. This line shows the date and time of an enqueue with ESMTP (EE) from the **tcp_intranet** channel to the **tcp_local** channel of a one (1) block message.
2. This is part of the same physical line of the log file as (1), presented here as a separate line for typographical convenience. It shows the envelope **From:** address, in this case adam@example.com, and the original version and current version of the envelope **To:** address, in this case **marlowe@example.org**.
3. This shows the date and time of a dequeue with ESMTP (DE) from the **tcp_local** channel of a one (1) block message that is, a successful send by the **tcp_local** channel to some remote SMTP server.
4. This shows the envelope **From:** address, the original envelope **To:** address, and the current form of the envelope **To:** address.
5. This shows that the actual system to which the connection was made is named **thor.example.org** in the DNS, that the local sending system has IP address 206.184.139.12 and is sending from port 2788, that the remote destination system has IP address 192.160.253.66 and the connection port on the remote destination system is port 25.
6. This shows the SMTP banner line of the remote SMTP server.
7. This shows the SMTP status code returned for this address; 250 is the basic SMTP success code and in addition, this remote SMTP server responds with extended SMTP status codes and some additional text.

MTA Logging Example: Including Optional Logging Fields

This example shows a logging entry similar to that shown in "Example MTA Logging: Sending to a List" with **LOG_FILENAME=1** and **LOG_MESSAGE_ID=1** showing the file name (1 and 3 below) and message ID (2 and 4 below). The message ID in particular can be used to correlate which entries relate to which message.

Example MTA Logging: Including Optional Logging Fields

```
16-Feb-2007 15:41:32.36 tcp_intranet tcp_local EE 1
adam@example.com rfc822;marlowe@example.org marlowe@example.org
/opt/sun/comms/messaging64/data/queue/tcp_local/002/ZZf0r4i0Wdy51.01 (1)
&lt;0JDJ00D02IBWDX00@example.com> (2)
example.org (example.org [192.160.253.66])

16-Feb-2007 15:41:34.73 tcp_local DE 1
adam@example.com rfc822;marlowe@example.org marlowe@example.org
/opt/sun/comms/messaging64/data/queue/tcp_local/002/ZZf0r4i0Wdy51.01 (3)
&lt;0JDJ00D02IBWDX00@example.com> (4)
thor.example.org dns;thor.example.org
(TCP|206.184.139.12|2788|192.160.253.66|25)
(thor.example.org ESMTP Sendmail ready at Thu, 15 Feb 2007 21:37:29 -0700 [MST])
smtp;250 2.1.5 &lt;marlowe@sexample.org>... Recipient ok
```

MTA Logging Example: Sending to a List

This example illustrates sending to multiple recipients with **LOG_FILENAME=1**, **LOG_MESSAGE_ID=1**, and **LOG_CONNECTION=1** enabled. Here user *adam@example.com* has sent to the MTA mailing list *test-list@example.com*, which expanded to *bob@example.com*, *carol@example.org*, and *david@example.org*. Note that the original envelope To: address is **test-list@example.com** for each recipient, though the current envelope *To:* address is each respective address. Note how the message ID is the same throughout, though two separate files (one for the **l** channel and one going out the **tcp_local** channel) are involved.

Example MTA Logging: Sending to a List

```
20-Feb-2007 14:00:16.46 tcp_local tcp_local EE 1
adam@example.com rfc822;test-list@example.com carol@example.org
/opt/sun/comms/messaging64/data/queue/tcp_local/004/ZZf0r2D0yuej4.01
&lt;0JDQ00706R0FX100@example.com>
example.org (example.org [192.160.253.66])

20-Feb-2007 14:00:16.47 tcp_local tcp_local EE 1
adam@example.com rfc822;test-list@example.com david@example.org
/opt/sun/comms/messaging64/data/queue/tcp_local/004/ZZf0r2D0yuej4.01
&lt;0JDQ00706R0FX100@example.com>
example.org (example.org [192.160.253.66])

20-Feb-2007 14:00:16.48 tcp_local ims-ms EE 1
adam@example.com rfc822;test-list@example.com bob@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/008/ZZf0r2D0yuej6.01
&lt;0JDQ00706R0FX100@example.com>
example.org (example.org [192.160.253.66])

20-Feb-2007 14:00:16.68 ims-ms D 1
adam@example.com rfc822;test-list@example.com bob@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/008/ZZf0r2D0yuej6.01
&lt;0JDQ00706R0FX100@example.com>

20-Feb-2007 14:00:17.73 tcp_local DE 1
adam@example.com rfc822;test-list@example.com carol@example.org
/opt/sun/comms/messaging64/data/queue/tcp_local/004/ZZf0r2D0yuej4.01
```

```
&lt;0JDQ00706R0FX100@example.com>
gw.example.org dns;gw.example.org (TCP|206.184.139.12|2788|192.160.253.66|25)
(gw.example.org -- SMTP Sendmail)
smtp;250 2.1.5 &lt;carol@example.org >... Recipient ok

20-Feb-2007 14:00:17.75 tcp_local DE 1
adam@example.com rfc822;test-list@example.com david@example.org
/opt/sun/comms/messaging64/data/queue/tcp_local/004/ZZf0r2D0yuej4.01
&lt;0JDQ00706R0FX100@example.com>
gw.example.org dns;gw.example.org (TCP|206.184.139.12|2788|192.160.253.66|25)
(gw.example.org -- SMTP Sendmail)
smtp;250 2.1.5 &lt;david@example.org>... Recipient ok
```

MTA Logging Example: Sending to a Nonexistent Domain

This example illustrates an attempt to send to a nonexistent domain (here **very.bogus.com**); that is, sending to a domain name that is not noticed as nonexistent by the MTA's rewrite rules and that the MTA matches to an outgoing TCP/IP channel. This example assumes the MTA option settings of **LOG_FILENAME=1** and **LOG_MESSAGE_ID=1**.

When the TCP/IP channel runs and checks for the domain name in the DNS, the DNS returns an error that no such name exists. Note the "rejection" entry (R), as seen in (5), with the DNS returning an error that this is not a legal domain name, as seen in (6).

Because the address is rejected after the message has been submitted, the MTA generates a bounce message to the original sender. The MTA enqueues the new rejection message to the original sender (1), and sends a copy to the postmaster (4) before deleting the original outbound message (the R entry shown in (5)).

Notification messages, such as bounce messages, have an empty envelope From: address--as seen, for instance, in (2) and (8)--in which the envelope From: field is shown as an empty space. The initial enqueue of a bounce message generated by the MTA shows the message ID for the new notification message followed by the message ID for the original message (3). (Such information is not always available to the MTA, but when it is available to be logged, it allows correlation of the log entries corresponding to the outbound failed message with the log entries corresponding to the resulting notification message.) Such notification messages are enqueued to the process channel, which in turn enqueues them to an appropriate destination channel (7).

Example MTA Logging: Sending to a Nonexistent Domain

```
20-Feb-2007 14:17:07.77 tcp_intranet tcp_local E 1
adam@example.com rfc822;user@very.bogus.com user@very.bogus.com
/opt/sun/comms/messaging64/data/queue/tcp_local/008/ZZf0r2D0CVaL0.00
&lt;0JDQ00903RS89T00@example.com>
example.org (example.org [192.160.253.66])

20-Feb-2007 14:17:08.24 tcp_local process E 1 (1)
rfc822;adam@example.com adam@example.com (2)
/opt/sun/comms/messaging64/data/queue/process/ZZf0r2D0CVbR0.00
&lt;0JDQ00904RSK9Z00@example.com>,&lt;0JDQ00903RS89T00@example.com> (3)
tcp-daemon.mailhost.example.com

20-Feb-2007 14:17:08.46 tcp_local process E 1 (4)
rfc822;postmaster@example.com postmaster@example.com
/opt/sun/comms/messaging64/data/queue/process/ZZf0r2D0CVbR1.00
&lt;0JDQ00906RSK9Z00@example.com>,&lt;0JDQ00903RS89T00@example.com>
tcp-daemon.mailhost.example.com
```

```
20-Feb-2007 14:17:08.46 tcp_local R 1 (5)
adam@example.com rfc822;user@very.bogus.com user@very.bogus.com
/opt/sun/comms/messaging64/data/queue/tcp_local/008/ZZf0r2D0CVaL0.00
&lt;0JDQ00903RS89T00@example.com>
Illegal host/domain name found (6)
(TCP active open: Failed gethostbyname() on very.bogus.com, resolver errno = 1)

20-Feb-2007 14:17:09.21 process ims-ms E 3 (7)
rfc822;adam@example.com adam@ims-ms-daemon (8)
/opt/sun/comms/messaging64/data/queue/ims-ms/018/ZZf0r2D0CVbS1.00
&lt;0JDQ00904RSK9Z00@example.com>
process-daemon.mailhost.example.com

20-Feb-2007 14:17:09.72 process ims-ms E 3
rfc822;postmaster@example.com postmaster@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/014/ZZf0r2D0CVbS2.00
&lt;0JDQ00906RSK9Z00@example.com>
process-daemon.mailhost.example.com

20-Feb-2007 14:17:09.73 ims-ms D 3
rfc822;adam@example.com adam@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/018/ZZf0r2D0CVbS1.00
&lt;0JDQ00904RSK9Z00@example.com>

20-Feb-2007 14:17:09.84 ims-ms D 3
rfc822;postmaster@example.com postmaster@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/014/ZZf0r2D0CVbS2.00
&lt;0JDQ00906RSK9Z00@example.com>
```

MTA Logging Example: Sending to a Nonexistent Remote User

This example illustrates an attempt to send to a bad address on a remote system. This example assumes MTA option settings of **LOG_FILENAME=1** and **LOG_MESSAGE_ID=1**, and channel option settings of **LOG_BANNER=1** and **LOG_TRANSPORTINFO=1**. Note the rejection entry (R), seen in (1). But in contrast to the rejection entry in "Example MTA Logging: Sending to a Nonexistent Domain", note that the rejection entry here shows that a connection to a remote system was made, and shows the SMTP error code issued by the remote SMTP server, (2) and (3). The inclusion of the information shown in (2) is due to setting the channel options **LOG_BANNER=1** and **LOG_TRANSPORTINFO=1**.

Example MTA Logging: Sending to a Nonexistent Remote User

```
26-Feb-2007 13:56:35.16 tcp_intranet tcp_local EE 1
adam@example.com rfc822;nonesuch@example.org nonesuch@example.org
/opt/sun/comms/messaging64/data/queue/tcp_local/000/ZZf0s690a3mf2.01
&lt;0JE100J08UU24H00@example.com>
example.org (example.org [192.160.253.66])

26-Feb-2007 13:56:35.19 tcp_local process E 1
rfc822;adam@example.com adam@example.com
/opt/sun/comms/messaging64/data/queue/process/ZZf0s690a3ml2.00
&lt;0JE100J09UUB4N00@example.com>,&lt;0JE100J08UU24H00@example.com>
tcp-daemon.mailhost.example.com

26-Feb-2007 13:56:35.20 tcp_local process E 1
rfc822;postmaster@example.com postmaster@example.com
/opt/sun/comms/messaging64/data/queue/process/ZZf0s690a3ml3.00
&lt;0JE100J0BUUB4N00@example.com>,&lt;0JE100J08UU24H00@example.com>
tcp-daemon.mailhost.example.com
```

```
26-Feb-2007 13:56:35.20 tcp_local RE 1 (1)
adam@example.com rfc822;nonesuch@example.org nonesuch@example.org

/opt/sun/comms/messaging64/data/queue/tcp_local/000/ZZf0s690a3mf2.01
&lt;0JE100J08UU24H00@example.com>
thor.example.org dns;thor.example.org
(TCP|206.184.139.12|2788|192.160.253.66|25) (2)
(thor.example.org -- Server ESMTP [Sun Java System Messaging
Server 6.2-8.01 [built Feb 16 2007]])
smtp;550 5.1.1 unknown or illegal alias: nonesuch@example.org (3)

26-Feb-2007 13:56:35.62 process ims-ms E 4
rfc822;adam@example.com adam@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/003/ZZf0s690a3mm5.00
&lt;0JE100J09UUB4N00@example.com>
process-daemon.mailhost.example.com

26-Feb-2007 13:56:36.07 process ims-ms E 4
rfc822;postmaster@example.com postmaster@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/016/ZZf0s690a3nm7.01
&lt;0JE100J0BUUB4N00@example.com>
process-daemon.mailhost.example.com

26-Feb-2007 13:56:35.83 ims-ms D 4
rfc822;adam@example.com adam@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/003/ZZf0s690a3mm5.00
&lt;0JE100J09UUB4N00@example.com>

26-Feb-2007 13:56:36.08 ims-ms D 4
rfc822;postmaster@example.com postmaster@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/016/ZZf0s690a3nm7.01
&lt;0JE100J0BUUB4N00@example.com>
```

MTA Logging Example: Rejecting a Remote Side's Attempt to Submit a Message

This example illustrates the sort of log file entry resulting when the MTA rejects a remote side's attempt to submit a message. (This example assumes that no optional **LOG_*** options are enabled, so only the basic fields are logged in the entry. Note that enabling the **LOG_CONNECTION** option, in particular, would result in additional informative fields in such J entries.) In this case, the example is for an MTA that has set up SMTP relay blocking with an **ORIG_SEND_ACCESS** mapping, including:

```
ORIG_SEND_ACCESS

! ...numerous entries omitted...
!
tcp_local|*|tcp_local|* $NRelaying$ not$ permitted
```

and where **alan@very.bogus.com** is not an internal address. Hence the attempt of the remote user **harold@example.org** to relay through the MTA system to the remote user **alan@very.bogus.com** is rejected.

Example MTA Logging: Rejecting a Remote Side's Attempt to Submit a Message

```
26-Feb-2007 14:10:06.89 tcp_local JE 0 (1)
harold@example.org rfc822; alan@very.bogus.com (2)
530 5.7.1 Relaying not allowed: alan@very.bogus.com (3)
```

1. This log shows the date and time the MTA rejects a remote side's attempt to submit a message. The rejection is indicated by a J record. (Cases where an MTA channel is attempting to send a message which is rejected is indicated by R

records, as shown in "Example MTA Logging: Sending to a Nonexistent Domain" and "MTA Logging Example: Sending to a Nonexistent Remote User").

> **Note:** The last J record written to the log will have an indication stating that it is the last for a given session. Also, the current version of Messaging Server does not place a limit on the number of J records.

2. The attempted envelope **From:** and **To:** addresses are shown. In this case, no original envelope **To:** information was available so that field is empty.
3. The entry includes the SMTP error message the MTA issued to the remote (attempted sender) side.

MTA Logging Example: Multiple Delivery Attempts

This example illustrates the sort of log file entries resulting when a message cannot be delivered upon the first attempt, so the MTA attempts to send the message several times. This example assumes option settings of **LOG_FILENAME=1** and **LOG_MESSAGE_ID=1**.

Example MTA Logging: Multiple Delivery Attempts

```
26-Feb-2007 14:38:16.27 tcp_intranet tcp_local EE 1 (1)
adam@example.com rfc822;user@some.org user@some.org
/opt/sun/comms/messaging64/data/queue/tcp_local/001/ZZf0s690kN_y0.00
&lt;0JE100L05WRJIC00@example.com>

26-Feb-2007 14:38:16.70 tcp_local Q 1 (2)
adam@example.com rfc822;user@some.org user@some.org
/opt/sun/comms/messaging64/data/queue/tcp_local/001/ZZf0s690kN_y0.00 (3)
&lt;0JE100L05WRJIC00@example.com>
TCP active open: Failed connect() 192.1.1.1:25 Error: no route to host (4)

...several hours worth of entries...

26-Feb-2007 16:58:11.20 tcp_local Q 1 (5)
adam@example.com rfc822;user@some.org user@some.org
/opt/sun/comms/messaging64/data/queue/tcp_local/001/ZYf0s690kN_y0.01 (6)
&lt;0JE100L05WRJIC00@example.com>
TCP active open: Failed connect() 192.1.1.1:25 Error: no route to host

...several hours worth of entries...

26-Feb-2007 19:15:12.11 tcp_local Q 1
adam@example.com rfc822;user@some.org user@some.org
/opt/sun/comms/messaging64/data/queue/tcp_local/001/ZXf0s690kN_y0.00 (7)
&lt;0JE100L05WRJIC00@example.com>
TCP active open: Failed connect() 192.1.1.1:25 Error: Connection refused (8)

...several hours worth of entries...

26-Feb-2007 22:41:12.63 tcp_local DE 1 (9)
adam@example.com rfc822;user@some.org user@some.org
/opt/sun/comms/messaging64/data/queue/tcp_local/001/ZXf0s690kN_y0.00
&lt;0JE100L05WRJIC00@example.com>
host.some.org dns;host.some.org (TCP|206.184.139.12|2788|192.1.1.1|25)
(All set, fire away)
smtp;250 2.1.5 &lt;user@some.org >... Recipient ok
```

1. The message comes in the **tcp_internal** channel---perhaps from a POP or IMAP client, or perhaps from another host within the organization using the MTA as an SMTP relay; the MTA enqueues it to the outgoing **tcp_local** channel.
2. The first delivery attempt fails, as indicated by the Q entry.
3. That this is a first delivery attempt can be seen from the **ZZ*** filename.
4. This delivery attempt failed when the TCP/IP package could not find a route to the remote side. As opposed to "MTA Logging Example: Sending to a Nonexistent Domain", the DNS did not object to the destination domain name, **some.org**; rather, the "no route to host" error indicates that there is some network problem between the sending and receiving side.
5. The next time the MTA periodic job runs it reattempts delivery, again unsuccessfully.
6. The file name is now **ZY***, indicating that this is a second attempt.
7. The file name is **ZX*** for this third unsuccessful attempt.
8. The next time the periodic job reattempts delivery the delivery fails, though this time the TCP/IP package is not complaining that it cannot get through to the remote SMTP server, but rather the remote SMTP server is not accepting connections. (Perhaps the remote side fixed their network problem, but has not yet brought their SMTP server back up---or their SMTP server is swamped handling other messages and hence was not accepting connections at the moment the MTA tried to connect.)
9. Finally the message is dequeued.

MTA Logging Example: Incoming SMTP Message Routed Through the Conversion Channel

This example illustrates the case of a message routed through the conversion channel. The site is assumed to have a CONVERSIONS mapping table such as:

```
CONVERSIONS

IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT Yes
```

This example assumes option settings of **LOG_FILENAME=1** and **LOG_MESSAGE_ID=1**.

Example MTA Logging: Incoming SMTP Message Routed Through the Conversion Channel

```
26-Feb-2007 15:31:04.17 tcp_local conversion EE 1 (1)
amy@example.edu rfc822;bert@example.com bert@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/conversion/ZZf0s090wFwx2.01
&lt;0JE100206Z7J5F00@example.edu>

26-Feb-2007 15:31:04.73 conversion ims-ms E 1 (2)
amy@example.edu rfc822;bert@example.com bert@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/007/ZZf0s090wMwq1.00
&lt;0JE100206Z7J5F00@example.edu>

26-Feb-2007 15:31:04.73 conversion D 1 (3)
amy@example.edu rfc822;bert@example.com bert@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/conversion/ZZf0s090wFwx2.01
&lt;0JE100206Z7J5F00@example.edu>

26-Feb-2007 15:31:04.73 ims-ms D 1 (4)
```

```
amy@example.edu rfc822;bert@example.com bert@ims-ms-daemon
/opt/sun/comms/messaging64/data/queue/ims-ms/007/ZZf0s090wMwq1.00
&lt;0JE100206Z7J5F00@example.edu>
```

1. The message from external user amy@example.edu comes in addressed to the **ims-ms** channel recipient bert@example.com. The **CONVERSIONS** mapping entry, however, causes the message to be initially enqueued to the conversion channel (rather than directly to the **ims-ms** channel).
2. The conversion channel runs and enqueues the message to the **ims-ms** channel.
3. Then the conversion channel can dequeue the message (delete the old message file).
4. And finally the **ims-ms** channel dequeues (delivers) the message.

MTA Logging Example: Outbound Connection Logging

This example illustrates log output for an outgoing message when connection logging is enabled, via **LOG_CONNECTION=3**. **LOG_PROCESS=1**, **LOG_MESSAGE_ID=1** and **LOG_FILENAME=1** are also assumed in this example. The example shows the case of user **adam@example.com** sending the same message (note that the message ID is the same for each message copy) to three recipients, **bobby@hosta.example.com**, **carl@hosta.example.com**, and **dave@hostb.example.com**. This example assumes that the message is going out a **tcp_local** channel marked (as such channels usually are) with the **single_sys** channel option. Therefore, a separate message file on disk will be created for each set of recipients to a separate host name, as seen in (1), (2), and (3), where the **bobby@hosta.example.com** and **carl@hosta.example.com** recipients are stored in the same message file, but the **dave@hostb.example.com** recipient is stored in a different message file.

Example MTA Logging: Outbound Connection Logging

```
28-Feb-2007 09:13:19.18 409f.3.1 tcp_intranet tcp_local EE 1
adam@example.com rfc822;bobby@hosta.example.com bobby@hosta.example.com
/opt/sun/comms/messaging64/data/queue/tcp_local/000/ZZf0s4g0G2Zt0.00 (1)
&lt;0JE500C0371HRJ00@example.com>
example.org (example.org [192.160.253.66])

28-Feb-2007 09:13:19.18 409f.3.1 tcp_intranet tcp_local EE 1
adam@example.com rfc822;carl@hosta.example.com carl@hosta.example.com
/opt/sun/comms/messaging64/data/queue/tcp_local/000/ZZf0s4g0G2Zt0.00 (2)
&lt;0JE500C0371HRJ00@example.com>
example.org (example.org [192.160.253.66])

28-Feb-2007 09:13:19.19 409f.3.2 tcp_intranet tcp_local EE 1
adam@example.com rfc822;dave@hostb.example.com dave@hostb.example.com
/opt/sun/comms/messaging64/data/queue/tcp_local/004/ZZf0s4g0G2Zt1.00 (3)
&lt;0JE500C0371HRJ00@example.com>
example.org (example.org [192.160.253.66])

28-Feb-2007 09:13:19.87 40a5.2.0 tcp_local - O (4)
TCP|206.184.139.12|5900|206.184.139.66|25
SMTP/hostb.example.com/mailhub.example.com (5)

28-Feb-2007 09:13:20.23 40a5.3.4 tcp_local - O (6)
TCP|206.184.139.12|5901|206.184.139.70|25
SMTP/hosta.example.com/hosta.example.com (7)

28-Feb-2007 09:13:20.50 40a5.2.5 tcp_local DE 1
adam@example.com rfc822;bobby@hosta.example.com bobby@hosta.example.com
```

```
/opt/sun/comms/messaging64/data/queue/tcp_local/000/ZZf0s4g0G2Zt0.00
&lt;0JE500C0371HRJ00@example.com>
hosta.example.com dns;hosta.example.com (8)
(TCP|206.184.139.12|5901|206.184.139.70|25)
(hosta.example.com -- Server ESMTP [Sun Java System Messaging Server
6.2-8.01 [built Feb 16 2007]])
smtp;250 2.1.5 bobby@hosta.example.com and options OK.

28-Feb-2007 09:13:20.50 40a5.2.5 tcp_local DE 1
adam@example.com rfc822;carl@hosta.example.com carl@hosta.example.com
/opt/sun/comms/messaging64/data/queue/tcp_local/000/ZZf0s4g0G2Zt0.00
&lt;0JE500C0371HRJ00@example.com>
hosta.example.com dns;hosta.example.com
(TCP|206.184.139.12|5901|206.184.139.70|25)
(hosta.example.com -- Server ESMTP [Sun Java System Messaging Server
6.2-8.01 [built Feb 16 2007]])
smtp;250 2.1.5 carl@hosta.example.com and options OK.

28-Feb-2007 09:13:20.50 40a5.2.6 tcp_local - C (9)
TCP|206.184.139.12|5901|206.184.139.70|25
SMTP/hosta.example.com/hosta.example.com

28-Feb-2007 09:13:21.13 40a5.3.7 tcp_local DE 1
adam@example.com rfc822;dave@hostb.example.com dave@hostb.example.com
/opt/sun/comms/messaging64/data/queue/tcp_local/004/ZZf0s4g0G2Zt1.00
&lt;0JE500C0371HRJ00@example.com>
mailhub.example.com dns;mailhub.example.com
(TCP|206.184.139.12|5900|206.184.139.66|25)
(mailhub.example.com ESMTP Sendmail ready at Tue, 27 Feb 2007 22:19:40 GMT)
smtp;250 2.1.5 &lt;dave@hostb.example.com>... Recipient ok

28-Feb-2007 09:13:21.33 40a5.3.8 tcp_local - C (10)
TCP|206.184.139.12|5900|206.184.139.66|25
SMTP/hostb.example.com/mailhub.example.com
```

1. The message is enqueued to the first recipient...
2. ....and to the second recipient...
3. ....and to the third recipient.
4. Having **LOG_CONNECTION=3** set causes the MTA to write this entry. The minus, **-**, indicates that this entry refers to an outgoing connection. The **O** means that this entry corresponds to the opening of the connection. Also note that the process ID here is the same, 40a5, since the same process is used for the multithreaded TCP/IP channel for these separate connection opens, though this open is being performed by thread 2 vs. thread 3.
5. As there are two separate remote systems to which to connect, the multithreaded SMTP client in separate threads opens up a connection to each---the first in this entry, and the second shown in 7. This part of the entry shows the sending and destination IP numbers and port numbers, and shows both the initial host name, and the host name found by doing a DNS lookup. In the **SMTP/***initial-host/dns-host* clauses, note the display of both the initial host name, and that used after performing a DNS MX record lookup on the initial host name: **mailhub.example.com** is apparently an MX server for **hostb.example.com**.
6. The multithreaded SMTP client opens up a connection to the second system in a separate thread (though the same process).

7. As there are two separate remote systems to which to connect, the multithreaded SMTP client in separate threads opens up a connection to each---the second in this entry, and the first shown above in 5. This part of the entry shows the sending and destination IP numbers and port numbers, and shows both the initial host name, and the host name found by doing a DNS lookup. In this example, the system **hosta.example.com** apparently receives mail directly itself.
8. Besides resulting in specific connection entries, **LOG_CONNECTION=3** also causes inclusion of connection related information in the regular message entries, as seen here for instance.
9. Having **LOG_CONNECTION=3** causes the MTA to write this entry. After any messages are dequeued, (the bobby and carl messages in this example), the connection is closed, as indicated by the **C** in this entry.
10. The connection mailhub.example.com is closed now that the delivery of the message (dave in this example) is complete.

MTA Logging Example: Inbound Connection Logging

This example illustrates log output for an incoming SMTP message when connection logging is enabled, via **LOG_CONNECTION=3**.

Example MTA Logging: Inbound Connection Logging

```
28-Feb-2007 11:50:59.10 tcp_local + O (1)
TCP|206.184.139.12|25|192.160.253.66|1244 SMTP (2)

28-Feb-2007 11:51:15.12 tcp_local ims-ms EE 1
service@example.org rfc822;adam@example.com adam@ims-ms-daemon
THOR.EXAMPLE.ORG (THOR.EXAMPLE.ORG [192.160.253.66]) (3)

28-Feb-2007 11:51:15.32 ims-ms D 1
service@example.org rfc822;adam@example.com adam@ims-ms-daemon

28-Feb-2007 11:51:15.66 tcp_local + C (4)
TCP|206.184.139.12|25|192.160.253.66|1244 SMTP
```

1. The remote system opens a connection. The **O** character indicates that this entry regards the opening of a connection; the + character indicates that this entry regards an incoming connection.
2. The IP numbers and ports for the connection are shown. In this entry, the receiving system (the system making the log file entry) has IP address 206.184.139.12 and the connection is being made to port 25; the sending system has IP address 192.160.253.66 and is sending from port 1244.
3. In the entry for the enqueue of the message from the incoming TCP/IP channel (**tcp_local**) to the **ims-ms** channel recipient, note that information beyond the default is included since **LOG_CONNECTION=3** is enabled. Specifically, the name that the sending system claimed on its HELO or EHLO line, the sending system's name as found by a DNS reverse lookup on the connection IP number, and the sending system's IP address are all logged.
4. The inbound connection is closed. The **C** character indicates that this entry regards the closing of a connection; the + character indicates that this entry regards an incoming connection.

Enabling Dispatcher Debugging

Dispatcher error and debugging output (if enabled) are written to the file **dispatcher.log** in the MTA log directory. A default dispatcher configuration is created at installation time and can be used without any changes made. However, if you want to modify the default configuration for security or performance reasons, you can do so by running the **msconfig** command.

Table 17–5 describes the dispatcher debugging bits.

Table 17–5 Dispatcher Debugging Bits

Bit	Hexadecimal value	Decimal value	Usage
0	x 00001	1	Startup, initialization, shutdown message
1	x 00002	2	Thread increment/decrement messages
2	x 00004	4	Configuration loading messages
3	x 00008	8	Process creation messages
4	x 00010	16	Process activity messages
5	x 00020	32	PORT_ACCESS mapping, connection messages
6	x 00040	64	Process calculation messages
7	x 00080	128	Process shutdown messages
8	x 00100	256	Socket listen, cookie messages
9	x 00200	512	Dispatcher internal I/O messages
10	x 00400	1024	Dispatcher internal read messages
11	x 00800	2048	Not used
12	x 01000	4096	Process management messages
13	x 02000	8192	Process handoff messages
14	x 04000	16384	Dispatcher message processing messages
15	x 08000	32768	Not used.
16	x 10000	65536	Successful connection message
17	x 20000	131072	Connection accept, TLS messages
18	x40000	262144	Not used
19	x80000	524288	Not used
20	x 100000	1048576	Statistics messages
21	x 200000	2097152	Extra statistics debugging.
22	x400000	4194304	Not used
23	x800000	8388608	Not used
24	x 1000000	16777216	Connection rejection message (subset of bits 4 and 5)

To Enable Dispatcher Error Debugging Output

1. Run the **msconfig set dispatcher.debug -1** (see "Restricted Options"). You can also set the logical or environmental variable **IMTA_DISPATCHER_DEBUG** (UNIX), which defines a 32-bit debug mask in hexadecimal, to the value **FFFFFFFF**. The preceding table describes the meaning of each bit.

To Set Dispatcher options (Oracle Solaris)

The dispatcher services offered in the dispatcher configuration affect requirements for various system options. The system's heap size (**datasize**) must be enough to accommodate the dispatcher's thread stack usage.

1. To display the heap size (that is, default **datasize**), use one of the following: The **csh** command:

   ```
   limit
   ```

   The **ksh** command:

   ```
   ulimit -a
   ```

   The Solaris utility:

   ```
   sysdef
   ```

2. For each dispatcher service compute **STACKSIZE*MAX_CONNS**, and then add up the values computed for each service. The system's heap size needs to be at least twice this number.

Note: Refer to the **Monitoring the MTA** chapter in the *Messaging Server Reference* for updated options and log format information.

Managing Message Store, Admin, and Default Service Logs

This section describes logging for the Message Store (POP, IMAP, and HTTP), Admin, and Default services. (See Table 17–1, " Messaging Server Log Files" for more information.)

For these services, you specify log settings and to view logs. The settings you specify affect which and how many events are logged. You can use those settings and other characteristics to refine searches for logged events when you are analyzing log files.

msconfig Logging Options

To control the location of log files, use the options described in Table 17–6 for specifying directory paths.

Note: The location of MTA log files, which are in the *DataRoot*/**log** directory, cannot be modified, but you can change the *log* subdirectory to symbolically link to another location. To separate the MTA logs from the rest of the log files, use **msconfig** options to specify non-default locations for non-MTA log files.

Table 17–6 msconfig Directory Paths for Log Files

Option	Description
***.logfile.logdir** **base.logfile.logdir** **dispatcher.logfile.logdir** **ens.logfile.logdir** **http.logfile.logdir** **imap.logfile.logdir** **imapproxy.logfile.logdir** **mta.logfile.logdir** **job_controller.logfile.logdir** **metermaid.logfile.logdir** **mmp.logfile.logdir** **msadmin.logfile.logdir** **messagetrace.logfile.logdir** **pop.logfile.logdir** **popproxy.logfile.logdir** **snmp.logfile.logdir** **submitproxy.logfile.logdir** **tcp_lmtp_server.logfile.logdir**	Directory path for log files. If this is not specified, log files will be placed in the *DataRoot***/log** directory. For the MTA, this option is only used by Message Store insertion tasks Directory path to the imta log file used for Message Store insertion (ims_master, LMTP). It is not used by other parts of the MTA which always log to the default location. The default location is *DataRoot***/log**. Changing that path to a soft-link is supported. (Restart of all services required). Syntax: dirpath

Understanding Service Log Characteristics

This section describes the following log characteristics for the message store and administration services: logging levels, categories of logged events, filename conventions for logs, and log-file directories.

Logging Levels

The level, or priority, of logging defines how detailed, or verbose, the logging activity is to be. A higher priority level means less detail; it means that only events of high priority (high severity) are logged. A lower level means greater detail; it means that more events are recorded in the log file.

You can set the logging level separately for each service---POP, IMAP, HTTP, Admin, and Default by setting the *service*.**logfile.loglevel** configuration option (see "Defining and Setting Service Logging Options"). You can also use logging levels to filter searches for log events. Table 17–7 describes the available levels. These logging levels are a subset of those defined by the UNIX **syslog** facility.

Table 17–7 Logging Levels for Store and Administration Services

Level	Description
Critical	The minimum logging detail. An event is written to the log whenever a severe problem or critical condition occurs---such as when the server cannot access a mailbox or a library needed for it to run.
Error	An event is written to the log whenever an error condition occurs---such as when a connection attempt to a client or another server fails.
Warning	An event is written to the log whenever a warning condition occurs---such as when the server cannot understand a communication sent to it by a client.

Table 17–7 (Cont.) Logging Levels for Store and Administration Services

Level	Description
Notice	An event is written to the log whenever a notice (a normal but significant condition) occurs---such as when a user login fails or when a session closes. This is the default log level.
Information	An event is written to the log with every significant action that takes place---such as when a user successfully logs on or off or creates or renames a mailbox.
Debug	The most verbose logging. Useful only for debugging purposes. Events are written to the log at individual steps within each process or task, to pinpoint problems.

When you select a particular logging level, events corresponding to that level and to all higher (less verbose) levels are logged. The default level of logging is **Notice**.

Note: The more verbose the logging you specify, the more disk space your log files will occupy; See "Defining and Setting Service Logging Options" for guidelines.

Categories of Logged Events

Within each supported service or protocol, Messaging Server further categorizes logged events by the facility, or functional area, in which they occur. Every logged event contains the name of the facility that generated it. These categories aid in filtering events during searches. Table 17–8 lists the categories that Messaging Server recognizes for logging purposes.

Table 17–8 Categories in Which Log Events Occur

Facility	Description
General	Undifferentiated actions related to this protocol or service
LDAP	Actions related to Messaging Server accessing the LDAP directory database
Network	Actions related to network connections (socket errors fall into this category)
Account	Actions related to user accounts (user logins fall into this category)
Protocol	Protocol-level actions related to protocol-specific commands (errors returned by POP, IMAP, or HTTP functions fall into this category)
Stats	Actions related to the gathering of server statistics
Store	Low-level actions related to accessing the message store (read/write errors fall into this category)

See "Searching and Viewing Service Logs" for examples of using categories as filters in log searches.

Service Log File Directories

Every logged service is assigned a single directory, in which its log files are stored. All IMAP log files are stored together, as are all POP log files, and log files of any other service. You define the location of each directory, and you also define how many log files of what maximum size are permitted to exist in the directory.

Make sure that your storage capacity is sufficient for all your log files. Log data can be voluminous, especially at lower (more verbose) logging levels.

It is important also to define your logging level, log rotation, log expiration, and server-backup policies appropriately so that all of your log-file directories are backed up and none of them become overloaded; otherwise, you may lose information. See "Defining and Setting Service Logging Options" for more information.

Understanding Service Log File Format

All message store and administration service log files created by Messaging Server have identical content formats. Log files are multiline text files, in which each line describes one logged event. All event descriptions, for each of the supported services, have the general format:

```
dateTime hostName processName[pid]: category logLevel: eventMessage
```

Store and Administration Log File Components

Table 17–9 lists the log file components for POP and IMAP. Note that this format of event descriptions is identical to that defined by the UNIX **syslog** facility, except that the date/time format is different and the format includes two additional components (*category* and *logLevel*).

Table 17–9 POP and IMAP Log File Formats

Component	Definition
dateTime	The date and time at which the event was logged, expressed in *dd/mm/yyyy hh:mm:ss* format, with a time-zone field expressed as *+/-hhmm* from GMT. For example:**02/Jan/1999:13:08:21 -0700**
hostName	The name of the host machine on which the server is running: for example, **showshoe**. Note: If there is more than one instance of Messaging Server on the host, you can use the process ID (pid) to separate logged events of one instance from another.
processName	The name of the process that generated the event: for example, **cgi_store**.
pid	The process ID of the process that generated the event: for example, **18753**.
category	The category that the event belongs to: for example, **General** (see "Example MTA Logging: Sending to a Nonexistent Remote User").
logLevel	The level of logging that the event represents: for example, **Notice** (see "Example MTA Logging: Sending to a Nonexistent Domain").
eventMessage	An event-specific explanatory message that may be of any length: for example, **Log created (894305624)**.

Here are three examples of logged events:

```
02/May/1998:17:37:32 -0700 showshoe cgi_store[18753]: General Notice:
Log created (894155852)

04/May/1998:11:07:44 -0400 xyzmail cgi_service[343]: General Error:
function=getserverhello|port=2500|error=failed to connect

03/Dec/1998:06:54:32 +0200 ExamplePost imapd[232]: Account Notice: close
[127.0.0.1]
[unauthenticated] 1998/12/3 6:54:32 0:00:00 0 115 0
```

IMAP and POP event entries may end with three numbers. The example above has 0 115 0. The first number is bytes sent by client, the second number is the bytes sent by the server, and third number is mailboxes selected (always 1 for POP).

When viewing a log file in the Log Viewer window, you can limit the events displayed by searching for any specific component in an event, such as a specific logging level or category, or a specific process ID. See "Searching and Viewing Service Logs" for more information.

The event message of each log entry is in a format specific to the type of event being logged, that is, each service defines what content appears in any of its event messages. Many event messages are simple and self-evident; others are more complex.

Defining and Setting Service Logging Options

You can define the message store and administration service logging configurations that best serve your administration needs. This section discusses issues that may help you decide on the best configurations and policies, and it explains how to implement them.

Flexible Logging Architecture

The naming scheme for log files (*service.sequenceNum.timeStamp*) helps you to design a flexible log-rotation and backup policy. The fact that events for different services are written to different files makes it easier for you to isolate problems quickly. Also, because the sequence number in a filename is ever-increasing and the timestamp is always unique, later log files do not simply overwrite earlier ones after a limited set of sequence numbers is exhausted. Instead, older log files are overwritten or deleted only when the more flexible limits of age, number of files, or total storage are reached.

Messaging Server supports automatic rotation of log files, which simplifies administration and facilitates backups. You are not required to manually retire the current log file and create a new one to hold subsequent logged events. You can back up all but the current log file in a directory at any time, without stopping the server or manually notifying the server to start a new log file.

In setting up your logging policies, you can set options (for each service) that control limits on total log storage, maximum number of log files, individual file size, maximum file age, and rate of log-file rotation.

Planning the Options You Want

Keep in mind that you must set several limits, more than one of which might cause the rotation or deletion of a log file. Whichever limit is reached first is the controlling one. For example, if your maximum log-file size is 3.5 MB, and you specify that a new log be created every day, you may actually get log files created faster than one per day if log data builds up faster than 3.5 MB every 24 hours. Then, if your maximum number of log files is 10 and your maximum age is 8 days, you may never reach the age limit on log files because the faster log rotation may mean that 10 files will have been created in less than 8 days.

The following default values, provided for Messaging Server administration logs, may be a reasonable starting point for planning:

Maximum number of log files in a directory: 10

Maximum log-file size: 2 MB

Total maximum size permitted for all log files: 20 MB

Minimum free disk space permitted: 5 MB

Log rollover time: 1 day

Maximum age before expiration: 7 days

Level of logging: Notice

You can see that this configuration assumes that server-administration log data is predicted to accumulate at about 2 MB per day, backups are weekly, and the total space allotted for storage of admin logs is at least 25 MB. (These settings may be insufficient if the logging level is more verbose.)

For POP, IMAP or HTTP logs, the same values might be a reasonable start. If all services have approximately the same log-storage requirements as the defaults shown here, you might expect to initially plan for about 150 MB of total log-storage capacity. (Note that this is meant only as a general indication of storage requirements; your actual requirements may be significantly different.)

Understanding Logging Options

You can set options that control the message store logging configuration by the command line.

The optimal settings for these options depend on the rate at which log data accumulates. It may take between 4,000 and 10,000 log entries to occupy 1 MB of storage. At the more verbose levels of logging (such as Notice), a moderately busy server may generate hundreds of megabytes of log data per week. Here is one approach you can follow:

- Set a level of logging that is consistent with your storage limits---that is, a level that you estimate will cause log-data accumulation at approximately the rate you used to estimate the storage limit.
- Define the log file size so that searching performance is not impacted. Also, coordinate it with your rotation schedule and your total storage limit. Given the rate at which log entries accumulate, you might set a maximum that is slightly larger than what you expect to accumulate by the time a rotation automatically occurs. And your maximum file size times your maximum number of files might be roughly equivalent to your total storage limit. For example, if your IMAP log rotation is daily, your expected accumulation of IMAP log data is 3 MB per day, and your total storage limit for IMAP logs is 25 MB, you might set a maximum IMAP log-file size of 3.5 MB. (In this example, you could still lose some log data if it accumulated so rapidly that all log files hit maximum size and the maximum number of log files were reached.)
- If server backups are weekly and you rotate IMAP log files daily, you might specify a maximum number of IMAP log files of about 10 (to account for faster rotation if the individual log-size limit is exceeded), and a maximum age of 7 or 8 days.
- Pick a total storage limit that is within your hardware capacity and that coordinates with the backup schedule you have planned for the server. Estimate the rate at which you anticipate that log data will accumulate, add a factor of safety, and define your total storage limit so that it is not exceeded over the period between server backups. For example, if you expect to accumulate an average of 3 MB of IMAP log-file data per day, and server backups are weekly, you might specify on the order of 25 - 30 MB as the storage limit for IMAP logs (assuming that your disk storage capacity is sufficient).
- For safety, pick a minimum amount of free disk space that you will permit on the volume that holds the log files. That way, if factors other than log-file size cause the volume to fill up, old log files will be deleted before a failure occurs from attempting to write log data to a full disk.

Searching and Viewing Service Logs

The log files provide the basic interface for viewing message store and administration log data. For a given service, log files are listed in chronological order. Once you have chosen a log file to search, you can narrow the search for individual events by specifying search options.

Search Options

These are useful search options you can specify for viewing log data:

- *A time period*. You can specify the beginning and end of a specific time period to retrieve events from, or you can specify a number of days (before the present) to search. You might typically specify a range to look at logged events leading up to a server crash or other occurrence whose time you know of. Alternatively, you might specify a day range to look at only today's events in the current log file.
- *A level of logging*. You can specify the logging level (see "Logging Levels" example, Critical to see why the server went down, or Error to locate failed protocol calls.
- *A facility.* You can specify the facility (see "Categories of Logged Events" that contains the problem; for example, Store if you believe a server crash involved a disk error, or Protocol if the problem lies in an IMAP protocol command error.
- *A text search pattern.* You can provide a text search pattern to further narrow the search. You can include any component of the event (see "Understanding Service Log File Format" search, such as event time, process name, process ID, and any part of the event message (such as remote host name, function name, error number, and so on) that you know defines the event or events you want to retrieve. Your search pattern can include the following special and wildcard characters:

* Any set of characters (example: ***.com**)

? Any single character (example: **199?**)

[nnn] Any character in the set *nnn* (example: **[aeiou]**)

[] Any character not in the set *nnn* (example: **[Managing Logging^aeiou]**)

[] Any character in the range *n-m* (example: **[A-Z]**)

[Managing Logging^n-m*]* Any character not in the range *n-m* (example: **[Managing Logging^0-9]**)

\ Escape character: place before *****, **?**, **[**, or **]** to use them as literals

> **Note:** Searches are case-sensitive.

Examples of combining logging level and facility in viewing logs might include the following:

- Specifying Account facility (and Notice level) to display failed logins, which may be useful when investigating potential security breaches
- Specifying Network facility (and all logging levels) to investigate connection problems
- Specifying all facilities (and Critical logging level) to look for basic problems in the functioning of the server

Working With Service Logs

This section describes how to work with service logs by using the **msconfig** command for searching and viewing logs.

To Send Service Logs to syslog

1. Run the **msconfig** command with the **syslogfacility** option:

   msconfig logfile.*service***.syslogfacility***value*

   where *service* is **admin**, **pop**, **imap**, **mta**, **base** or **http** and *value* is **user**, **mail**, **daemon**, **local0** to **local7**, or none.

   Once the value is set, messages are logged to the **syslog** facility corresponding to the set value and all the other log file service options are ignored. When the option is not set or the value is **none**, logging uses the Messaging Server log files.

To Disable HTTP Logging

If your system does not support HTTP message access, that is, Webmail, you can disable HTTP logging by setting the following variables. Do not set these variables if your system requires Webmail support (for example, mshttpd).

1. Run the following **msconfig** commands:

```
msconfig
msconfig> set http.enable 0
msconfig set http.enablesslport 0
msconfig write
```

To Set the Server Log Level

1. Run the following **msconfig** command:

```
msconfig set service.logfile.loglevel loglevel
```

   where *service* is **admin**, **pop**, **imap**, **mta**, **base** or **http** and *loglevel* is **Nolog**, **Critical**, **Error**, **Warning**, **Notice**, **Information**, or **Debug**.

To Specify a Directory Path for Server Log Files

1. Run the following **msconfig** command:

```
msconfig service.logfile.logdir dirpath
```

To Specify a Maximum File Size for Each Service Log

1. Run the following **msconfig** command:

```
msconfig set service.logfile.maxlogfilesize size
```

   where *size* specifies a number of bytes.

To Specify a Service Log Rotation Schedule

1. Run the following **msconfig** command:

```
msconfig set service.logfile.rollovertime number
```

   where *number* specifies a number of seconds.

To Specify a Maximum Number of Service Log Files Per Directory

1. Run the following **msconfig** command:

```
msconfig service.logfile.maxlogsize number
```

where *number* specifies the maximum number of log files.

To Specify a Storage Limit

1. Run the following **msconfig** command:

   ```
   msconfig service.logfile.maxlogsize number
   ```

   where *number* specifies a number in bytes.

To Specify the Minimum Amount of Free Disk Space to Reserve

1. Run the following **msconfig** command:

   ```
   msconfig service.logfile.minfreediskspace number
   ```

   where *number* specifies a number in bytes.

To Specify an Age for Logs at Which They Expire

```
msconfig service.logfile.expirytime number
```

where *number* specifies a number in seconds.

Implementing and Configuring Message Store Transaction Logging

This section describes how to configure message store transaction logging.

Overview of Message Store Transaction Logging

You can use Message Store transaction logging to record Messaging Server user actions and events, tracing messages similar to the way the MTA traces messages. Tracing messages in this fashion allows you to track critical events of a message's life cycle.

Message Store transaction logging uses the same XML format used in the MTA **mail.log*** and **connection.log*** files. This XML format is the default. It provides the same transaction information previously logged in the process log at the **notice** and **information** log levels as well as information in **transactlog** logging. It also includes IMAP and POP context numbers in all IMAP and POP log messages. You do not set log levels for Message Store transaction logs. You instead configure settings to determine which events and attributes to log. Message Store transaction logging uses a log roll over daemon which rolls over all logs previously written using **nslogger**. Message Store transaction logging provides a backward-compatibility mode that behaves the way the previous store logging worked, combining process and action logging together.

Message Store Transaction Logging Log Entries

For a list of message store transaction logging user actions and attributes, see *Messaging Server Reference*.

An action log entry is an XML element with the following characteristics:

- The element tag is the two character action label, which follows XML syntax to start and end: <tag ... />
- The element has many attributes defined previously, and each action has own attributes.
- The attribute starts with a two character attribute label, and its value is quoted.
- The following attributes are required for each action entry: timestamp (**ts**), service name (**sn**), and process id (pi).

A typical log entry resembles the following:

```
<co ts="2016-11-04T16:05:39.71" sn="imapd" pi="7149" ac="C"
tr="TCP|192.0.2.0|25|192.0.2.1|33469" us="admin" nt="2015/2/9 13:43:52 0:01:02 158
1032 1"/>
```

where:

- **co** indicates the action label, which is connection closed.
- Three mandatory attributes, **ts** (time stamp), **sn** (service name), and **pi** (process id) are present.
- The action specific attributes are **us** (user name) and **ac** (action flags).

Configuring Message Store Transaction Logging

Transaction logging configuration is applied globally to IMAP, POP, and delivery Message Store components.

Enabling Message Store Transaction Logging

To trace transactions in the Message Store transaction log, you configure message tracing in addition to the logging configuration.

To enable Message Store transaction logging, run the following command:

```
msconfig set messagetrace.activate transactlog
```

To show the configuration, run the following command:

```
msconfig show messagetrace.activate
role.messagetrace.activate = transactlog
```

Enabling Message Store Transaction Log Actions and Attributes

The syntax to enable store actions and attributes is a matrix that you configure separately using actions and action attributes.

Table 17–10 shows the actions and their descriptions.

Table 17–10 Message Store Transaction Log Actions

Action	Description
all	All actions are enabled. This is the default.
+({li | lo | ma | ...})	Only actions in the list are enabled. Actions are separated by a space.
-({li | lo | ma | ...})	All actions are enabled except those in the list. Actions are separated by a space.

To enable all Message Store actions except for login and logout actions, run the following command.

```
msconfig set imap.actions '-(li lo)'
```

Running the **msconfig show imap.actions** command shows the **imap.actions** settings.

```
msconfig show imap.actions
role.imap.actions = -(li lo)
```

Table 17–11 shows the attributes and their descriptions.

Table 17–11 Message Store Attributes

Attributes	Description
all	All attributes are enabled. This is the default.
+({us | mi | ii | ...})	Only attributes in the list are enabled. Attributes are separated by a space.
-({us | mi | ii | ...})	All attributes are enabled except those in the list. Attributes are separated by a space.

To enable only the canonical user name and session ID attributes, run the following command.

```
msconfig set imap.actionattributes '+(us si)'
```

Running the **msconfig show imap.actionattributes** shows the **imap.actionattributes** settings.

```
msconfig show imap.actionattributes
role.imap.actionattributes = +(us si)
```

Message Store Transaction Log Examples

The examples in this section use the following Messaging Store transaction logging settings:

```
msconfig show messagetrace
role.messagetrace.activate = transactlog

msconfig show imap.actions
role.imap.actions = all

msconfig show imap.actionattributes
role.imap.actionattributes = all
```

Example Message Store Transction Logs for IMAP

Example 1: Log in as Joe.

```
<li ts="2016-11-04T16:05:39.71" sn="imapd" pi="26435" us="joe"
tr="tr="TCP|192.0.2.0|25|192.0.2.1|33469" at="plaintext" cs="noSSL" si="0"/>
```

Example 2: Create mailbox Folder1 and Folder2.

```
<mc ts="2016-11-04T16:05:39.71" sn="imapd" pi="26435" us="joe" ma="Folder1"/>
<mc ts="22016-11-04T16:05:39.98 -0700" sn="imapd" pi="26435" us="joe"
ma="Folder2"/>
```

Example 3: Delete mailbox Folder1.

```
<md ts="2016-11-04T16:05:39.71" sn="imapd" pi="26435" us="joe" ma="Folder1"/>
```

Example 4: Append message to Folder2.

```
<ma ts="2016-11-04T16:05:39.71" sn="imapd" pi="27080" us="" ma="user/joe/Folder2"
sz="163" ui="1" uv="1406234045" cx="0"/>
```

Example 5: Expunge 2 messages in the Inbox.

```
<ex ts="2016-11-04T16:05:39.71" sn="imapd" pi="30247" ma="user/joe"
mi="0N8S00A0B007SS00@host.example.com" no="[127.0.0.1:52383]"/>
<ex ts="2016-11-04T16:05:55.70" sn="imapd" pi="30247" ma="user/joe"
mi="0N8R00A06ZLSSS00@host.example.com" no="[127.0.0.1:52383]"/>
```

Example 6: Log out.

```
<co ts="2016-11-04T16:05:39.71" sn="imapd" pi="7149" ac="C"
tr="tr="TCP|192.0.2.0|25|192.0.2.1|33469" us="admin" nt="2015/2/9 13:43:52 0:01:02
158 1032 1"/>
```

Example 7: Connect to an email account.

```
<co ts="2016-11-04T16:05:39.71" sn="imapd" pi="27433"
tr="tr="TCP|192.0.2.0|25|192.0.2.1|33469" at="ssl"/>
```

Example 8: Select a folder.

```
<sl ts="2016-11-04T16:05:39.71" sn="imapd" pi="27433"
tr="TCP|192.0.2.0|25|192.0.2.1|33469" ma="user/admin" us="admin"/>
```

Example 9: Rename a folder.

```
imap command: A01 RENAME Folder1 Folder2
<mr ts="2016-11-04T16:05:39.71" sn="imapd" pi="27433" us="admin" ma="Folder2"
om="Folder1"/>
```

Example 10: Subscribe to a folder.

```
imap command: A01 SUBSCRIBE INBOX
<ms ts="2016-11-04T16:05:39.71" sn="imapd" pi="27433" us="admin" ma="inbox"/>
```

Example 11: Unsubscribe from a folder.

```
imap command: A01 UNSUBSCRIBE INBOX
<mu ts="2016-11-04T16:05:39.71" sn="imapd" pi="3613" us="admin" ma="inbox"/>
```

Example 12: Set acl to a mailbox.

```
imap command: A01 SETACL folder1 david lrstwiead
<ac ts="2016-11-04T16:05:39.71" sn="imapd" pi="27433" us="admin"
ma="user/admin/folder1" nt="admin lrstwiepkxancd :admin lrstwiepkxancd David
lrstwiead "/>
```

Example 13: Fetch rfc822.text.

```
imap command: A01 FETCH 1 RFC822.TEXT
<fe ts="2016-11-04T16:05:39.71" sn="imapd" pi="3613" us="admin" ma="user/admin"
sz="-1:0" mi="<54C145D2.7010202@shenmail.com>"/>
```

Example 14: Change quota.

```
imap command: A01 SETQUOTA user/admin/Folder1 (STORAGE 512)
<qc ts="2016-11-04T16:05:39.71" sn="imapd" pi="3613" us="admin"
ur="user/admin/Folder1" dq="512"/>
```

Example 15: Quota exceeds limit.

```
imquotacheck command: ./imquotacheck -n -l -r rulefile
<qe ts="2016-11-04T16:05:39.71" sn="imquotacheck" pi="26490" us="admin" dq="2"
du="43" mq="0" mc="0" qt="90" qr="rule3"/>
```

See "imquotacheck" for more information on syntax and options.

Example Message Store Transaction Logs for POP

Example 1: Log in as Joe.

```
<li ts="2016-11-04T16:05:39.71" sn="popd" pi="27522" us="joe"
```

```
"tr="TCP|192.0.2.0|25|192.0.2.1|33469" at="plaintext" cs="noSSL" si="21"/>
```

Example 2: Fetch message.

```
<fe ts="2016-11-04T16:05:39.71" sn="popd" pi="27522" us="joe" om="user/joe"
mi="<0N8S00A0E09CSS00@host.example.com>"/>
```

Example 3: Delete message.

```
<ex ts="2016-11-04T16:05:39.71" sn="popd" pi="27522" ma="user/joe"
mi="<0N8S00A0E09CSS00@host.example.com>" no="sc11136733"/>
```

Example 4: Connect to an email account.

```
<co ts="02016-11-04T16:05:39.71" sn="popd" pi="27443"
tr="TCP|192.0.2.0|25|192.0.2.1|33469" at=""/>
```

Example Message Store Transaction Logs for Message Delivery

Example 1: Send five messages to Joe.

```
<ma ts="2016-11-04T16:05:39.71" sn="ims_master" pi="9236" us="" ma="user/joe"
sz="404652" mi="<0N9800L0UQWP9600@host.example.com>" ui="5" uv="1392927327"
cx="0"/>
<ma ts="2016-11-04T16:05:39.75" sn="ims_master" pi="9236" us="" ma="user/joe"
sz="404652" mi="<0N9800L0WQWP9600@host.example.com>" ui="6" uv="1392927327"
cx="0"/>
<ma ts="2016-11-04T16:05:40.01" sn="ims_master" pi="9236" us="" ma="user/joe"
sz="404652" mi="<0N9800L0YQWP9600@host.example.com>" ui="7" uv="1392927327"
cx="0"/>
<ma ts="2016-11-04T16:05:40.34" sn="ims_master" pi="9240" us="" ma="user/joe"
sz="404652" mi="<0N9800L10QWP9600@host.example.com>" ui="8" uv="1392927327"
cx="0"/>
<ma ts="2016-11-04T16:05:40.77" sn="ims_master" pi="9240" us="" ma="user/joe"
sz="404652" mi="<0N9800L12QWQ9600@host.example.com>" ui="9" uv="1392927327"
cx="0"/>
```

Other Message Store Logging Features

Messaging Server provides a feature called telemetry that can capture a user's entire IMAP or POP session into a file. This feature is useful for debugging client problems. For example, if a user complains that their message access client is not working as expected, this feature can be used to trace the interaction between the access client and Messaging Server. See "Checking User IMAP/POP/Webmail Session by Using Telemetry" for more information.

Message Store Logging Examples

The exact field format and list of fields logged in the Message Store log files vary according to the logging options set. This section shows a few examples of interpreting typical sorts of log entries.

Message Store Logging Example: Bad Password

When a user types an invalid password, "authentication" failure is logged, as opposed to a "user not found" message. The message "user not found" is the text passed to the client for security reasons, but the real reason (invalid password) is logged.

Example Message Store Logging: Invalid Password

```
[31/Aug/2004:16:33:14 \-0700|] vadar imapd[13027]: Account Notice: badlogin:
[192.18.126.64:40718] plaintext user1 authentication failure
```

Message Store Logging Example: Account Disabled

The following example shows why a user cannot log in due to a disabled account. Furthermore, the disabled account is clarified as "(inactive)" or "(hold)."

Example Message Store Logging: Account Disabled

```
[31/Aug/2004:16:33:14 \-0700|] vadar imapd[13027]: Account Notice: badlogin:
[192.18.126.64:40720] plaintext user3 account disabled (hold)
```

Message Store Logging Example: Message Appended

The following example shows an append message, which occurs when whenever a message is appended to a folder. The Message Store log records all messages entering the Message Store through the **ims_master** and **lmtp** channels. Records the "append" of user ID, folder, message size, and message ID.

Example Message Store Logging: Append

```
[31/Aug/2004:16:33:14 \-0700|] vadar ims_master[13822]: Store Information:append:
user1:user/user1:659:<Roam.SIMC.2.0.6.1093995286.11265.user1@vadar.example.org>
```

Message Store Logging Example: Message Retrieved by a Client

The Message Store log writes a "fetch" message when a client retrieves a message. The Message Store log records all client fetches of at least one body part. Records the "fetch" of user ID, folder, and message-ID.

Example Message Store Logging: Message Retrieved by a Client

```
[31/Aug/2004:16:33:14 \-0700|] vadar imapd[13729]: Store Information:
fetch:user1:user/user1:<Roam.SIMC.2.0.6.1093051161.3655.user1@vad.example.org>
```

Message Store Logging Example: Message Removed from a Folder

The following example shows how to remove a message from a folder.

The Message Store writes an "expunge" message when an IMAP or POP message is removed from a folder (but not removed from the system). It is logged whether it is expunged by the user or a utility. Records the "expunge" of folder and message ID.

Example Message Store Logging Example: Message Removed from a Folder

```
[31/Aug/2004:16:33:14 \-0700|] vadar imexpire[13923]: Store Information:
expunge:user/user1:<Roam.SIMC.2.0.6.1090458838.2929.user1@vadar.example.org>
```

Example Message Store Logging: Login

```
[31/Aug/2004:16:33:14 \-0700|] vadar imapd[13027]: Account Information: login
[192.18.126.64:40718] user1 plaintext
```

Using Message Store Log Messages

For more information about message store log messages, refer to the following topics:

- Overview of Logging: Provides an introduction to Messaging Server logging concepts.
- Tools for Managing Logging: Lists available tools for managing logs.
- See the discussion on logging options in the *Messaging Server Reference*.

  Search for **logfile**. The information lists the configuration options that you set for Messaging Server by using the **configutil** command. You can view the

documentation for the Unified Configuration equivalent of a **configutil** option by clicking on the **configutil** name link.

- Managing Message Store, Admin, and Default Service Logs: Describes logging for the Message Store (POP, IMAP, and HTTP), Admin, and Default services.

MMP Logging

The logging format used by the MMP is unstable and subject to change at any time. It uses the **nslog** model and syntax used by the store, with similar configuration settings. The most interesting line in the MMP log is:

```
[04/Feb/2016:10:42:24 -0800] myhostname ImapProxyAService.cfg[17242]: General
Notice: (id 3) Proxy connect for client 10.0.0.5:55159 (cleartext) canonical
user 'admin@host.example.com' original user 'admin' auth 'PLAIN' via MMP
127.0.0.1:55162 to backend 127.0.0.1 (cleartext)
```

which can be used to correlate a connection from a client to a back end connection.

For debugging the MMP, the loglevel may be raised and the **debugkeys** option can be helpful.

For additional information, see the discussion on MMP in *Messaging Server Reference*.

18

Monitoring Messaging Server

This chapter describes how to monitor Oracle Communications Messaging Server. In most cases, a well-planned, well-configured server performs without extensive intervention from an administrator. As an administrator, however, it is your job to monitor the server for signs of problems.

In addition to this chapter, see the following chapters for more information about monitoring Messaging Server:

- Monitoring User Access to the Message Store
- Monitoring System Performance
- Monitoring Disk Space
- Monitoring the MTA
- Monitoring LDAP Directory Server
- Monitoring the Message Store
- SNMP Support

Automatic Monitoring and Restart

Messaging Server provides a way to transparently monitor services and automatically restart them if they crash or become unresponsive (the services hangs or freeze up). It can monitor all message store, MTA, and MMP services including the IMAP, POP, HTTP, job controller, dispatcher, and MMP servers. It does not monitor other services such as SMS or TCP/SNMP servers. (TCP/SNMP is monitored by the job controller.) See "Automatic Restart of Failed or Unresponsive Services" and "Monitoring Using msprobe and watcher Functions" for more information.

Daily Monitoring Tasks

The most important tasks you should perform on a daily basis are checking postmaster mail, monitoring the log files, and setting up the **stored** utility. These tasks are described below.

Checking Postmaster Mail

Messaging Server has a predefined administrative mailing list set up for postmaster email. Any users who are part of this mailing list will automatically receive mail addressed to postmaster.

The rules for postmaster mail are defined in RFC822, which requires every email site to accept mail addressed to a user or mailing list named postmaster and that mail sent to this address be delivered to a real person. All messages sent to postmaster@host.domain are sent to a postmaster account or mailing list.

Typically, the postmaster address is where users should send email about their mail service. As postmaster, you might receive mail from local users about server response time, from other server administrators who are encountering problems sending mail to your server, and so on. You should check postmaster mail daily.

You can also configure the server to send certain error messages to the postmaster address. For example, when the MTA cannot route or deliver a message, you can be notified via email sent to the postmaster address. You can also send exception condition warnings (low disk space, poor server response) to postmaster.

Monitoring and Maintaining the Log Files

Messaging Server creates a separate set of log files for each of the major protocols or services it supports including SMTP, IMAP, POP, and HTTP. These are located in *DataRoot*/**log**. You should monitor the log files on a routine basis--especially if you are having problems with the server.

Be aware that logging can impact server performance. The more verbose the logging you specify, the more disk space your log files will occupy for a given amount of time. You should define effective but realistic log rotation, expiration, and backup policies for your server. See "Managing Logging" for information about defining logging policies for your server.

Setting Up the msprobe Utility

The **msprobe** utility automatically performs monitoring and restart functions. See "Monitoring Using msprobe and watcher Functions" for more information.

Utilities and Tools for Monitoring

The following tools are available for monitoring:

- immonitor-access
- imcheck
- Log Files
- imsimta counters
- imsimta qm counters
- MTA Monitoring Using SNMP
- Monitoring Using msprobe and watcher Functions
- Monitoring Using msstatbot Tool

immonitor-access

"immonitor-access" monitors the status of the following Messaging Server components/processes: Mail Delivery (SMTP server), Message Access and Store (POP and IMAP servers), Directory Service (LDAP server) and HTTP server. This utility measures the response times of the various services and the total round trip time taken to send and retrieve a message. The Directory Service is monitored by looking up a specified user in the directory and measuring the response time. Mail Delivery is

monitored by sending a message (SMTP) and the Message Access and Store is monitored by retrieving it. Monitoring the HTTP server is limited to finding out whether or not it is up and running.

See "immonitor-access" for complete instructions.

imcheck

Use "imcheck" to monitor database statistics including logs and transactions.

> **Note:** The **imcheck -s** command, which prints database statistics, is only valid for classic message store.

counterutil

This utility provides statistics acquired from different system counters. See "Gathering Message Store Counter Statistics by Using counterutil" for more information.

Log Files

Messaging server logs event records for SMTP, IMAP, POP, and HTTP. The policies for creating and managing the Messaging Server log files are customizable.

Since logging can affect the server performance, logging should be considered very carefully before the burden is put on the server. Refer to "Managing Logging" for more information.

imsimta counters

The MTA accumulates message traffic counters based upon the Mail Monitoring MIB, RFC 1566 for each of its active channels. The channel counters are intended to help indicate the trend and health of your email system. Channel counters are not designed to provide an accurate accounting of message traffic. See the discussion about MTA logging in "Managing Logging" for precise accounting.

The MTA channel counters are implemented using the lightest weight mechanisms available so that they cause as little impact as possible on actual operation. Channel counters do not try harder: if an attempt to map the section fails, no information is recorded. If one of the locks in the section cannot be obtained almost immediately, no information is recorded. When a system is shut down, the information contained in the in-memory section is lost forever.

The **imsimta counters -show** command provides MTA channel message statistics (see below). These counters need to be examined over time noting the minimum values seen. The minimums may actually be negative for some channels. A negative value means that there were messages queued for a channel at the time that its counters were zeroed (for example, the cluster-wide database of counters created). When those messages were dequeued, the associated counters for the channel were decremented and therefore leading to a negative minimum. For such a counter, the correct "absolute" value is the current value less the minimum value that counter has ever held since being initialized.

```
Channel Messages Recipients Blocks
------- -------- ---------- -------
tcp_local
Received 29379 79714 982252 (1)
Stored 61 113 -2004 (2)
Delivered 29369 79723 983903 (29369 first time) (3)
Submitted 13698 13699 18261 (4)
Attempted 0 0 0 (5)
```

```
Rejected 1 10 0 (6)
Failed 104 104 4681 (7)
Queue time/count 16425/29440 = 0.56 (8)
Queue first time/count 16425/29440 = 0.56 (9)
Total In Assocs 297637
Total Out Assocs 28306
```

1)Received is the number of messages enqueued to the channel named **tcp_local**. That is, the messages enqueued (E records in the **mail.log*** file) to the **tcp_local** channel by any other channel.

2)Stored is the number of messages stored in the channel queue to be delivered.

3)Delivered is the number of messages which have been processed (dequeued) by the channel **tcp_local**. (That is, D records in the **mail.log*** file.) A dequeue operation may either correspond to a successful delivery (that is, an enqueue to another channel), or to a dequeue due to the message being returned to the sender. This will generally correspond to the number **Received** minus the number **Stored**.

The MTA also keeps track of how many of the messages were dequeued upon first attempt; this number is shown in parentheses.

4)Submitted is the number of messages enqueued (E records in the **mail.log** file) by the channel **tcp_local** to any other channel.

5)Attempted is the number of messages which have experienced temporary problems in dequeuing, that is, Q or Z records in the **mail.log*** file.

6)Rejected is the number of attempted enqueues which have been rejected, that is, **J** records in the **mail.log*** file.

7)Failed is the number of attempted dequeues which have failed, that is, **R** records in the **mail.log*** file.

8)Queue time/count is the average time-spent-in-queue for the delivered messages. This includes both the messages delivered upon the first attempt, see (9), and the messages that required additional delivery attempts (hence typically spent noticeable time waiting fallow in the queue).

9)Queue first time/count is the average time-spent-in-queue for the messages delivered upon the first attempt.

Note that the number of messages submitted can be greater than the number delivered. This is often the case, since each message the channel dequeues (delivers) will result in at least one new message enqueued (submitted) but possibly more than one. For example, if a message has two recipients reached via different channels, then two enqueues will be required. Or if a message bounces, a copy will go back to the sender and another copy may be sent to the postmaster. Usually that will be two submissions (unless both are reached through the same channel).

More generally, the connection between **Submitted** and **Delivered** varies according to type of channel. For example, in the conversion channel, a message would be enqueued by some other arbitrary channel, and then the conversion channel would process that message and enqueue it to a third channel and mark the message as dequeued from its own queue. Each individual message takes a path:

```
elsewhere -> conversion E record Received
conversion -> elsewhere E record Submitted
conversion D record Delivered
```

However, for a channel such as **tcp_local** which is not a "pass through," but rather has two separate pieces (slave and master), there is no connection between **Submitted** and

Delivered. The **Submitted** counter has to do with the SMTP server portion of the **tcp_local** channel, whereas the **Delivered** counter has to do with the SMTP client portion of the **tcp_local** channel. Those are two completely separate programs, and the messages travelling through them may be completely separate.

Messages submitted to the SMTP server:

tcp_local -> elsewhere E record Submitted

Messages sent out to other SMTP hosts via the SMTP client:

```
elsewhere -> tcp_local E record Received
tcp_local D record Delivered
```

Channel dequeues (delivers) will result in at least one new message enqueued (submitted) but possibly more than one. For example, if a message has two recipients reached via different channels, then two enqueues will be required. Or if a message bounces, a copy will go back to the sender and another copy may be sent to the postmaster. Usually that will be reached through the same channel.

imsimta counters Implementation

For performance reasons, a node running the MTA keeps a cache of channel counters in memory using a shared memory section. As processes on the node enqueue and dequeue messages, they update the counters in this in-memory cache. If the in-memory section does not exist when a channel runs, the section will be created automatically. (The **imta start** command also creates the in-memory section, if it does not exist.)

The command **imta counters -clear** or the **imta qm** command **counters clear** may be used to reset the counters to zero.

imsimta qm counters

The **imsimta qm counters** utility displays MTA channel queue message counters. You must be root or **mailsrv** to run this utility. The output fields are the same as those described in insmita counters. See *Messaging Server Reference* for more information.

Example:

```
imsimta counters -create
imsimta qm counters show
Channel Messages Recipients Blocks
---------------------- ---------- ---------- ----------
tcp_intranet
Received 13077 13859 264616
Stored 92 91 -362
Delivered 12985 13768 264978
Submitted 2594 2594 3641
...
```

Every time you restart the MTA, you must run: **imsimta counters -create**

imsimta qm summarize

The **imsimta qm summarize** utility displays a summary of the number of messages and their status in the MTA channel queues.

For more details of the various switches available, see the **summarize** sub-command in imsimta qm and the **imsimta qm help summarize** command.

qm summarize modes

Like many of the **qm** sub-commands, **summarize** has two modes of operation: The **-directory_tree** mode examines the message files in the MTA queue directories on disk. The **-database** mode queries the **job_controller** process's in-memory database structures. The directory mode creates a heavier load on the IO system and may not reflect what **job_controller** is actually working on, but it can be useful to know if there is a difference between the two. The job controller makes the decisions about which messages are tried next, so the database mode will be more useful.

```
imsimta qm
qm.maint> sum -directory_tree
Channel Messages Size (Mb)
------------------------------- -------- ---------
conversion 0 0.0
hold 0 0.0
ims-ms (stopped) 2 0.0
process 0 0.0
reprocess 0 0.0
tcp_intranet (stopped) 0 0.0
tcp_local (stopped) 2 0.0
------------------------------- -------- ---------
Totals 4 0.0
```

Notice that the **-database** mode breaks down the number messages into three catagories. **Active** messages are currently being tried by a worker process. **Pending** messages are ready to be tried by a worker as soon as thread/slot is available. **Delayed** messages have been tried before and are waiting for a specified time to be tried again as per the **backoff** option for that channel.

```
qm.maint> sum -database
Total Total
Channel Messages = Active + Pending + Delayed Size (Mb)
------------------------------- -------- -------- -------- -------- ---------
conversion 0 0 0 0 0.0
hold 0 0 0 0 0.0
ims-ms (stopped) 2 0 2 0 0.0
l 0 0 0 0 0.0
process 0 0 0 0 0.0
reprocess 0 0 0 0 0.0
tcp_intranet (stopped) 0 0 0 0 0.0
tcp_local (stopped) 2 0 2 0 0.0
------------------------------- -------- -------- -------- -------- ---------
Totals 4 0 4 0 0.0
```

Note: In these examples, some channels had been stopped using the **imsimta qm stop** *channel* command to provide some data to look at.

Held messages

A **.HELD** message file is a message which has encountered a loop or otherwise been *sidelined* and requires administrative intervention for some reason. You can see such messages using the **-held** switch. Note that job_controller will have no knowledge of held messages, therefore the **-database** and **-held** switches are mutually exclusive. See "Diagnosing and Cleaning up .HELD Messages" for more information about **.HELD** messages .

```
qm.maint> sum -held -database
%QM-E-CMDERR, Conflicting parameters and/or qualifiers: (DATABASE AND HELD)

qm.maint> sum -held
```

```
Held Held
Channel Messages Size (Mb) Oldest Queued Messages Size (Mb) Oldest Held
------------------------------- -------- --------- ----------------- --------
--------- -----------------
conversion 0 0.0 0 0.0
hold 0 0.0 1 0.0 23 Apr, 21:35:16
ims-ms (stopped) 2 0.0 6 Apr, 13:24:00 0 0.0
process 0 0.0 0 0.0
reprocess 0 0.0 0 0.0
tcp_intranet (stopped) 0 0.0 0 0.0
tcp_local (stopped) 2 0.0 5 May, 10:16:08 0 0.0
------------------------------- -------- --------- ----------------- --------
--------- -----------------
Totals 4 0.0 6 Apr, 13:24:00 1 0.0 23 Apr, 21:35:16
```

Displaying Summary by Destination Host

The **-hosts** switch displays a breakdown of the messages in the queue by destination host for channels where that is meaningful. This information is stored in the **job_controller** process in-memory queue cache database. Therefore **-hosts** implies **-database**.

```
qm.maint> sum -hosts
Total Total
Channel Host Messages = Active + Pending + Delayed Size (Mb)
------------------------------- -------- -------- -------- -------- ---------
conversion 0 0 0 0 0.0
hold 0 0 0 0 0.0
ims-ms (stopped) 2 0 2 0 0.0
l 0 0 0 0 0.0
process 0 0 0 0 0.0
reprocess 0 0 0 0 0.0
tcp_intranet (stopped) 0 0 0 0 0.0
tcp_local (stopped) 2 0 2 0 0.0
aol.com 1 0 1 0 0.0
sun.com 1 0 1 0 0.0
------------------------------- -------- -------- -------- -------- ---------
Totals 4 0 4 0 0.0
```

imsimta qm jobs

After starting the tcp_local channel:

```
tcp_local 1 1 0 0 0.0
aol.com 1 1 0 0 0.0
```

And to see what processes are working on what jobs:

```
qm.maint> jobs tcp_local
tcp_local channel:

Pending: 0 jobs
Active: 1 jobs, 1 messages (0.00 Mb), 1 recipients
Current jobs have delivered 1 messages, requeued 0 messages

Active jobs and messages:

22157: 1 messages (0.00 Mb), 1 recipients
1 messages processed and 0 requeued

Active hosts:
```

```
aol.com

Active messages:

ZZg0u410_P_~1.01 (1.0 Kb)
```

MTA Monitoring Using SNMP

Messaging Server supports system monitoring through the Simple Network Management Protocol (SNMP). Using an SNMP client (sometimes called a *network manager*) such as Sun Net Manager or HP OpenView (not provided with this product), you can monitor certain parts of the Messaging Server. Refer to "SNMP Support" for details.

Monitoring Using msprobe and watcher Functions

Messaging Server provides two processes, **watcher** and **msprobe** to monitor various system services. **watcher** watches for server crashes and restarts them as necessary. **msprobe** monitors server hangs (unresponsiveness). Specifically **msprobe** monitors the following:

- *Server Response Time*. **msprobe** connects to the enabled servers using their protocol commands and measures their response times. If the response time exceeds the alarm warning threshold, an alarm message is sent (see "Alarm Messages") to a server, or the server response time exceeds a specified timeout period, the server is restarted. Server response times are recorded in a counter database and is logged to the default log file. **counterutil** can be used to display the server response time statistics ("counterutil").

The following servers are monitored by **msprobe**: **imap**, **pop**, **http**, **cert**, **job_controller**, **smtp**, **lmtp**, **mmp** and **ens**. When **smtp** or **lmtp** are not responding, the dispatcher is restarted. **ens** cannot be automatically restarted.

- *Disk usage*. **msprobe** checks the disk availability and usage for every message store partition. Specifically it checks the message store **mboxlist** database directory and the MTA queue directory. If disk usage exceeds a configured threshold, an alarm message is sent. The disk sizes and usages are recorded in a counter database and is logged to the default log file. Administrators can use the **counterutil** utility (see "counterutil") to display the disk usage statistics.
- *Message Store mboxlist Database Log File Accumulation*. Log file accumulation is an indication of an **mboxlist** database error. **msprobe** counts the number of active log files and if the number of active log files is larger than the threshold, **msprobe** logs a critical error message to the **default** log file to inform the admin to restart the server. If the autorestart is enabled (**local.autorestart** to **yes**), the store daemon is restarted.

watcher and **msprobe** are controlled by the **msconfig** options shown in Table 18–1. See "Automatic Restart of Failed or Unresponsive Services" for more information.

Table 18–1 msprobe and watcher msconfig Options

Options	Description
base.autorestart.enable	Enable automatic server restart. Automatically restarts failed or hung services. Default: 1
base.autorestart.timeout	Failure retry time-out. If a server fails more than twice in this designated amount of time, then the system stops trying to restart the server. The value (set in seconds) should be set to a period value longer than the **msprobe** interval (**schedule.task:msprobe**). Default: 600 seconds
msprobe.probe:*service***.timeout**	Timeout for a specific server before restart. *service* can be imap, pop, http, cert, job_controller, smtp, lmtp, mmp or ens. Default: use **msprobe.timeout**
msprobe.probe:*service***.warningthreshold**	Number of seconds of a specific server's non-response before a warning message is logged to **default** log file. *service* can be imap, pop, http, cert, job_controller, smtp, lmtp, mmp or ens. Default: Use **msprobe.warningthreshold**
msprobe.warningthreshold	Number of seconds of server non-response before a warning message is logged to **default** log file. Default: 25 secs
msprobe.queuedir	MTA queue directory to check if queue size exceeded threshold defined by **alarm.system:diskavail.thresholddirection**. Default: none
msprobe.timeout	Period of server non-response before restarting that server. See "Expire and Purge Log and Scheduling Options" **schedule.task:msprobe.crontab**. Default: 30 seconds
schedule.task:msprobe.crontab	**msprobe** run schedule. A **crontab** style schedule string (see **schedule.task:expire.enable** in). Note that by default, this is automatically set. See "Pre-defined Automatic Tasks". To disable: set **schedule.task:msprobe.enable** to **0**.
watcher.enable	Enable watcher which monitors service failures. (IMAP, POP, HTTP, job controller, dispatcher, message store (**stored**), **imsched**, and MMP. (LMTP/SMTP servers are monitored by the dispatcher and LMTP/SMTP clients are monitored by the job_controller.) Logs error messages to the default log file for specific failures. Default: 1

Alarm Messages

msprobe can issue alarms in the form of email messages to the postmaster (see "To Monitor imapd, popd and httpd") warning of a specified condition. A sample email alarm sent when a certain threshold is exceeded is shown below:

```
Subject: ALARM: server response time in seconds of "ldap_example.com_389" is 10
Date: Tue, 17 Jul 2001 16:37:08 -0700 (PDT)
From: postmaster@example.com
To: postmaster@example.com
Server instance: /opt/sun/comms/messaging64
Alarmid: serverresponse
Instance: ldap_example_europa.com_389
Description: server response time in seconds
Current measured value (17/Jul/2001:16:37:08 -0700): 10
Lowest recorded value: 0
Highest recorded value: 10
Monitoring interval: 600 seconds
Alarm condition is when over threshold of 10
Number of times over threshold: 1
```

You can specify how often **msprobe** monitors disk and server performance, and under what circumstances it sends alarms. This is done by using the **msconfig** command to set the alarm options. Table 18–2 shows some useful alarm options along with their default setting. See *Messaging Server Reference* for all options.

Table 18–2 Useful Alarm Message msconfig Options

Option	Description (Default in Parenthesis)
alarm.noticehost	(localhost) Machine to which you send warning messages.
alarm.noticeport	(587) The SMTP port to which to connect when sending alarm message.
alarm.noticercpt	(Postmaster@localhost) Whom to send alarm notice.
alarm.noticesender	(Postmaster@localhost) Address of sender the alarm.
alarm.system:diskavail.description	(Percentage mail partition diskspace available.) Text for description field for disk availability alarm.
alarm.system:diskavail.statinterval	(3600) Interval in seconds between disk availability checks. Set to 0 to disable checking of disk usage.
alarm.system:diskavail.threshold	(10) Percentage of disk space availability below which an alarm is sent.
alarm.system:diskavail.thresholddirection	(-1) Specifies whether the alarm is issued when disk space availability goes below threshold (-1) or above it (1).
alarm.system:diskavail.warninginterval	(24). Interval in hours between subsequent repetition of disk availability alarms.
alarm.system:serverresponse.description	(Server response time in seconds.) Text for description field for servers response alarm.
alarm.system:serverresponse.statinterval	(600) Interval in seconds between server response checks. Set to 0 to disable checking of server response.
alarm.system:serverresponse.threshold	(10) If server response time in seconds exceeds this value, alarm issued.
alarm.system:serverresponse.thresholddirection	(1) Specifies whether alarm is issued when server response time is greater that (1) or less than (-1) the threshold.
alarm.system:serverresponse.warninginterval	(24) Interval in hours between subsequent repetition of server response alarm.

Monitoring Using msstatbot Tool

Message stores uses **msstatbot** tool to perform basic administrative tasks, and monitor cluster health.

Beyond message stores, Elasticsearch engine and MTA also uses **msstatbot** monitoring tool to visualize and track their running states and health.

For message store, the tool supports administrative and monitoring functions as follows:

- Administrative Functions: Administrative function includes backing up and restoring the data. A backup is a snapshot of all on-disk data files (SSTable files) stored in the data directory. You can set a retention policy that defines how to handle the snapshot files for older backup data. The default policy is to retain On Server backup files for 30 days.
- Monitoring Functions: Monitoring functions includes monitoring clusters and diagnosing problems in the cluster and nodes. Table 18–3 lists the **nodetool** commands to collect statistics and status:

Table 18–3 nodetool Commands

Nodetool command	Description
Status	cluster information (state, load, IDs, ...)
tablestats	statistics on tables Support json format (-F json, --format json)
tpstats	usage statistics of thread pools Support json format (-F json, --format json)
gcstats	JVM GC Statistics
netstats	Network information on provided host (connecting node by default)

MTA stats can also be collected using **:mtastats**.

For the **nodetool** commands details, see the Cassandra documentation at: *http://cassandra.apache.org/doc/latest/tools/nodetool*.

Stats Available from the msstatbot Tool

Following is the stats available from the **msstatbot** tool:

- netstats - cassandra node
- tpstats - cassandra node
- tablestats -cassandra node
- gcstats - cassandra node
- status - cassandra node
- mtastats - mta node

Installing the msstatbot Tool

msstatbot tool is distributed as python 2.7 package for Messaging Server statistics monitoring daemon (**msstatd**).You can install the **msstatbot** tool using the following rpm command:

```
rpm -i msstatbot-1.0-1.noarch.rpm
```

The **msstatbot** gets installed in the location **/opt/sun/comms/messaging64/lib/python2.7/site-packages/**. This location is non-relocatable.

In Cassandra node, the **msstatbot** also gets installed in the location **/opt/sun/comm/messaging64/lib/python 2.7/site-packages/**.

msstatd server provides APIs for the clients to configure, start, stop, and check running health of Oracle messaging server services.

Configuration

The **msstatbot** tool supports three types of configuration:

- when it is installed on message server node, the configurations has to set. Example:

  ```
  <serverroot>/bin/msconfig set role.msstatbot.port 8889
  ```

It loads Messaging Server's unified configuration.

Make sure that **msconfig** should have following parameters configured:

- elasticsearch.hostlist
- elasticsearch.port
  - ./msconfig show elasticsearch
  - role.elasticsearch.hostlist = <ip list>
  - role.elasticsearch.port = <port #>
  - role.store.searchengine elastic
- role.dispatcher.service:SMTP.tcp_ports: SMTP server tcp port
  - ./msconfig show service:SMTP.tcp_ports
  - role.dispatcher.service:SMTP.tcp_ports = 25
- msstatbot.port: default to 8889
- msstatbot.enabledstats: enabled stats to monitor, which should be set to **mtastat** on MTA node
  - ./msconfig show msstatbot
  - role.msstatbot.port = 8190
  - role.msstatbot.enabledstats = mtastats:20

Example:

```
role.store.dbtype = cassandra
role.store.searchengine = elastic
role.store.casconnectpoints = 10.196.12.157
role.store.casmetarf = 1
role.store.casmsgrf = 1
role.msstatbot.port = 8889
role.msstatbot.enabledstats = mtastats:20
```

- when it is installed on cassandra node, it loads json formatted configuration:

```
{
"es_hosts": "replace_with_elasticsearch_host_name",
 "es_port": "replace_with_elasticsearch_port",
"storetype": "cassandra",
"port": 8889,
"enablestats": "gcstats:15",
"nodetoolpath": "path/to/nodetool",
"pidfile": "cassbot",
"logfile": "path/to/logfile",
"loglevel": "INFO",
"maxbytes": 20971520,
"backupcount":10
```

- when it is installed on Cassandra node and Messaging Server, make sure that **msconfig** should have following parameters configured:
  - elasticsearch.hostlist:
  - elasticsearch.port
    - ./msconfig show elasticsearch
    - role.elasticsearch.hostlist = <ip list>

        * role.elasticsearch.port = <port #>
        * role.store.searchengine elastic
    - role.dispatcher.service:SMTP.tcp_ports: SMTP server tcp port
        * ./msconfig show service:SMTP.tcp_ports
        * role.dispatcher.service:SMTP.tcp_ports = 25
    - msstatbot.port: default to 8889
    - msstatbot.enabledstats: enabled stats to monitor, which should be stat to mtastats on MTA node
        * ./msconfig show msstatbot
        * role.msstatbot.port = 8190
        * role.msstatbot.enabledstats = mtastats:20
        * role.msstatbot.nodetoolpath = <node tool path>

The signficance of ':<number>'with the stats, is the frequency (in secs) at which the stats are collected from the cassandra/MTA.

Notes

- nodetool has to be in the path, or specified with nodetoolpath in the configuration
- replace the hosts and ports for Elasticsearch
- if pidfile is set, daemon will use it as pidfile, otherwise uses ./msstatpid. As for Messaging Server config, it will to set to path/to/ms/data/proc/msstatpid
- logfile, loglevel, maxbytes and backupcount are for logging configuration.
    - logfile: set the path to log file;
    - loglevel: set the logging level of [CRITICAL, ERROR, WARNING, INFO, DEBUG], otherwise default to INFO;
    - maxbyptes: the max log file size, rotating the log file if exceeding the size;
    - backupcount: total number log files kept in the log folder.

Assumptions

- nodetool path setting should meet one of following three conditions:
    - nodetool should be on the path.
    - when cassandra is installed, by default, it has nodetool path in that particular location. nodetoolpath is given in the configuration.
    - nodetool is installed under **/var/opt/cassandra/dse-*/bin** <default Cassandra installation location.
- Elasticsearch dependency
    - Elasticsearch is installed and accessible to all nodes running this program.
    - Elasticsearch python client lib is installed with python version used to run the program.
    - Elasticsearch hosts and ports should be configured.

Starting and Stopping Statistics Monitoring

You can start the **msstatd** server, run **msstatd** with **start** command, and configuration file as follows:

```
python msstatd.py -c start -f msstatd.conf
```

You can stop the **msstatd** server, run **msstatd** with **stop** command, and configuration file as follows:

```
python msstatd.py -c stop -f msstatd.conf
```

When **msstatd** tool is installed with messaging server, it will load the configuration from **msconfig xml**. In this case, we assume that **msstatd** tool is installed under **/opt/sun/comms/messaging64/lib/python2.7/site-packages/src/**, and it is run with root or mailsrv privilege.

```
python msstatd.py -c start
python msstatd.py -c stop
```

msstatd Syntax

```
msstatd.py [-h] [-c {start,stop}] [-p PORT] [-i IP] [-f CONFIG]
```

Table 18–4 describes the **msstatd** options.

Table 18–4 msstatd Options

Option	Description
-h, **--help**	Displays the help.
-c {start,stop}, **--command {start,stop}**	start | stop the msstatd server.
-p PORT , **--port PORT**	Listening port for msstatd server. PORT: the port the daemon listens on, if missing, the default is **8889**.
-i IP, **--ip IP**	msstatd server IP. IP: the node IP address, if missing, the default is hostname of the node.
-f CONFIG, **--config CONFIG**	msstatd server configuration file. CONFIG: the json formatted configuration, if missing, the daemon will check If it is a messaging server node, it will load unified configuration; If it is a cassandra node (nodetool is installed and in the path), it will load default configuration.

To start/stop/restart stat collection after **msstatd** server starts:

```
curl -X POST <host>:<port>/stat/ -H "Content-Type: application/json" -d
'{"start":"netstats:30"}'
curl -X POST <host>:<port>/stat/ -H "Content-Type: application/json" -d
'{"stop":"gcstats"}'
curl -X POST <host>:<port>/stat/ -H "Content-Type: application/json" -d
'{"restart":"netstats:25"}'
```

Querying the Node Statistics

The query of statistics is with Elasticsearch. But, **msstatd** provides RESTful APIs to query data too.

- With browser:

  ```
  http://<msstatd host>:<msstatd port>/stat/netstats?count=1&node=<cassandra
  node>
  ```

- With CURL:

  ```
  curl -X GET
  <host>:<port>/stat/<tpstats|tablestats>?node=?&ks=?&table=?&count=?
  ```

  where,

  node is Cassandra node, *ks* is keyspace, *table* is table name, and *count* is count of results to return.

Following is the responses to query **tablestat**:

```
Tabletstats json format :  {"hits": {"hits": [{"sort": [1551180643392], "_type":
"cassstats", "_source": {"bloom_filter_space_used_f": 0.0,
"number_of_partitions_estimate_f": 6,
"bloom_filter_off_heap_memory_used_f": 0.0, "space_used_live_f": 0,
"table_s": "ms_msg.message",
"compression_metadata_off_heap_memory_used_f": 0,
"average_live_cells_per_slice_last_five_minutes_f": 1.0,
"memtable_off_heap_memory_used_f": 0, "percent_repaired_f": 100.0,
"sstable_compression_ratio_f": -1.0, "local_write_latency_ms_f": 0.028,
"ts": 1551180643392, "maximum_tombstones_per_slice_last_five_minutes_f":
1.0, "proc": "cassandra", "memtable_switch_count_f": 0, "node":
"kkm00cxy", "local_read_count_f": 15, "pending_flushes_f": 0,
"local_write_count_f": 7, "off_heap_memory_used_total_f": 0,
"average_tombstones_per_slice_last_five_minutes_f": 1.0,
"space_used_total_f": 0.0, "memtable_data_size_f": 27096.0,
"compacted_partition_minimum_bytes_f": 0,
"compacted_partition_maximum_bytes_f": 0.0,
"maximum_live_cells_per_slice_last_five_minutes_f": 1.0,
"bloom_filter_false_ratio_f": 0.0, "compacted_partition_mean_bytes_f":
0, "dropped_mutations_f": 0.0, "index_summary_off_heap_memory_used_f":
0.0, "bloom_filter_false_positives_f": 0.0, "local_read_latency_ms_f":
0.18, "memtable_cell_count_f": 7, "space_used_by_snapshots_total_f": 0},
"_score": null, "_index": "ms_tablestats_2019_02_25", "_id":
"tablestats-16929923c40"}], "total": 21, "max_score": null}, "_shards":
{"successful": 5, "failed": 0, "skipped": 0, "total": 5}, "took": 5,
"timed_out": false}
```

Log Files

Two log files, **system.log** and **msstatd.log**, are generated in the location **/python2.7/site-packages/src/**.

Uninstalling the msstatbot Tool

You can install the **msstatbot** tool using the following rpm command:

```
rpm -e msstatbot
```

This command uninstalls the packages from **/python 2.7/site-packages/**.

19

Monitoring the MTA

This chapter describes how to monitor the Oracle Communications Messaging Server Message Transfer Agent (MTA).

Monitoring the Size of the Message Queues

Excessive message queue growth may indicate that messages are not being delivered, are being delayed in their delivery, or are coming in faster than the system can deliver them. Reasons for this situation include a denial of service attack caused by huge numbers of messages flooding your system, or the Job Controller not running.

See "Channel Message Queues", "Messages Are Not Dequeued", and "MTA Messages Are Not Delivered" for more information on message queues.

Symptoms of Message Queue Problems

- Disk space usage grows.
- User not receiving messages in a reasonable time.
- Message queue sizes are abnormally high.

To Monitor the Size of the Message Queues

Probably the best way to monitor the message queues is to use imsmita counters and **imsimta qm** summarize.

You can also monitor the number of files in the queue directories (*DataRoot*/**queue/)**. The number of files will be site-specific, and you'll need to build a baseline history to find out what is "too many." This can be done by recording the size of the queue files over a two week period to get an approximate average.

Checking for Held messages

If the MTA detects a message is looping, it will be **sidelined** by renaming the queue message file to **.HELD**. For more discussion of how messages can become **.HELD** and what to do about them, refer to "Diagnosing and Cleaning up .HELD Messages". To see whether there are any held messages, use the **imsimta qm summarize -held** command described in *Messaging Server Reference*.

Monitoring Rate of Delivery Failure

A delivery failure is a failed attempt to deliver a message to an external site. A large increase in rate of delivery failure can be a sign of a network problem such as a dead DNS server or a remote server timing out on responding to connections.

Symptoms of Rate of Delivery

There are no outward symptoms. Lots of **Q** records will appear in to **mail.log_current**.

To Monitor the Rate of Delivery Failure

Delivery failures are recorded in the MTA logs with the logging entry code **Q**. Look at the record in the file *DataRoot***/log/mail.log_current**. Example:

mail.log:06-Oct-2003 00:24:03.66 501d.0b.9 ims-ms Q 5 durai.balusamy@Sun.COM rfc822;durai.balusamy@Sun.COM durai@ims-ms-daemon <00ce01c38bda$c7e2b240$6501a8c0@guindy>Mailbox is busy

Monitoring Inbound SMTP Connections

An unusual increase in the number of inbound SMTP connections from a given IP address may indicate:

- An external user is trying to relay mail.
- An external user is trying to do a service denial attack.

Symptoms of Unauthorized SMTP Connections

- **External user relaying mail** : No outward symptoms.
- **Service denial attack**: External attempt to overload the SMTP servers with message requests.

To Monitor Inbound SMTP Connections

- **External user relaying mail**: Look in *DataRoot***/log/mail.log_current** for records with the logging entry code **J** (rejected relays). To turn on logging of remote IP addresses run the following command: **msconfig set log_connection 1**

  Note that there is a slight performance trade-off when this feature is enabled.

- **Service denial attack**: To find out who and how many users are connecting to the SMTP servers, you can run the command **netstat** and check for connections at the SMTP port (default: 25). Example:

```
Local address Remote address State
192.18.79.44.25 192.18.78.44.56035 32768 0 32768 0 CLOSE_WAIT
192.18.79.44.25 192.18.136.54.57390 8760 0 24820 0 ESTABLISHED
192.18.79.44.25 192.18.26.165.48508 33580 0 24820 0 TIME_WAIT
```

Note that you will first need to determine the appropriate number of SMTP connections and their states (**ESTABLISHED**, **CLOSE_WAIT**, etc.) for your system to determine if a particular reading is out of the ordinary. If you find many connections staying in the **SYN_RECEIVED** state this might be caused by a broken network or a denial of service attack. In addition, the lifetime of an SMTP server process is limited. This is controlled by the MTA Dispatcher configuration option **MAX_LIFE_TIME**. The default is 86,400 seconds (one day). Similarly, **MAX_LIFE_CONNS** specifies the

maximum number of connections a server process can handle in its lifetime. If you find a particular SMTP server that has around for a long time you may want to investigate.

Monitoring the Dispatcher and Job Controller Processes

The Dispatcher and Job Controller Processes must be operating for MTA to work. You should have one process of each kind.

Symptoms of Dispatcher and Job Controller Processes Down

If the Dispatcher is down or does not have enough resources, SMTP connections are refused. If the Job Controller is down, queue size will grow.

To Monitor Dispatcher and Job Controller Processes

Check to see that the processes called **dispatcher** and **job_controller** exist. See "Check that the Job Controller and Dispatcher Are Running" for more information.

20

SNMP Support

This chapter describes how to enable Simple Network Management Protocol (SNMP) support for system monitoring of Oracle Communications Messaging Server. It also gives an overview of the type of information provided by SNMP. Note that it does not describe how to view this information from an SNMP client. Refer to your SNMP client documentation for details on how to use it to view SNMP-based information.

Using an SNMP client (sometimes called a *network manager*) such as Sun Net Manager or HP OpenView (not provided with the this product), you can monitor certain parts of Messaging Server. See "Monitoring Messaging Server" for more information on monitoring the Messaging Server.

This information also describes some of the data available from the Messaging Server SNMP implementation, but complete MIB details are available from RFC 2788 (http://www.faqs.org/rfcs/rfc2788.html) and RFC 2789 (http://www.faqs.org/rfcs/rfc2788.html).

SNMP Implementation

Messaging Server implements two standardized MIBs, the Network Services Monitoring MIB (RFC 2788) and the Mail Monitoring MIB (RFC 2789). The Network Services Monitoring MIB provides for the monitoring of network services such as POP, IMAP, HTTP, and SMTP servers. The Mail Monitoring MIB provides for the monitoring of MTAs. The Mail Monitoring MIB allows for monitoring both the active and historical state of each MTA channel. The active information focuses on currently queued messages and open network connections (for example, counts of queued messages, source IP addresses of open network connections), while the historical information provides cumulative totals (for example, total messages processed, total inbound connections).

Note: For a complete listing of Messaging Server SNMP monitoring information, refer to RFC 2788 and RFC 2789.

SNMP is supported on platforms running Oracle Solaris and Linux. Messaging Server on the Oracle Solaris 9 Operating System uses Solstice Enterprise Agents (SEA). Starting with the Oracle Solaris 10 Operating System, Messaging Server supports the open source Net-SNMP monitoring framework, relegating the Oracle Solaris 9 Solstice Enterprise Agents (SEA) technology to legacy (end of support life) status. Additionally, Net-SNMP is widely used on Linux platforms. Messaging Server uses its Net-SNMP-based SNMP subagent on Oracle Solaris 10 and later as well as Linux platforms.

With the adoption of the Net-SNMP framework, Messaging Server's SNMP subagent provides new functionality:

- Support for SNMP versions 2c and 3. This support is provided by the Net-SNMP framework. The former SNMP technology, Solstice Enterprise Agents, only provided support for SNMP version 1. Enhanced security features and access controls are the primary benefit of these two versions of SNMP.
- The subagent may be configured to run as a "standalone" SNMP agent. This provides sites with additional means of isolating their various SNMP agents running on the same system.
- Multiple "instances" of Messaging Server running on the same system may concurrently be monitored. This support is provided through either the second item in this list, or through the use of SNMP version 3 "context names". This allows for SNMP monitoring of Messaging Server in failover clusters.

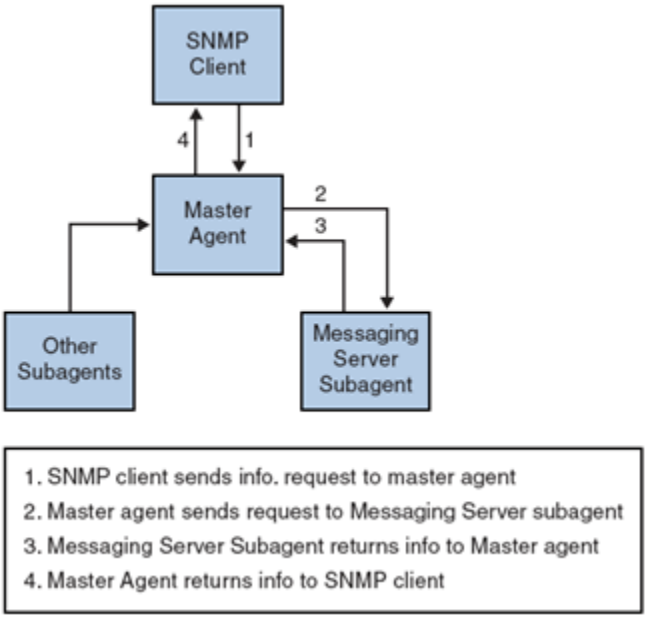
Limitations of the Messaging Server SNMP support are as follows:

- Only one instance of Messaging Server per host computer can be monitored via SNMP on Oracle Solaris 9.
- The SNMP support is for monitoring only. No SNMP management is supported.
- No SNMP traps are implemented. (RFC 2788 provides similar functionality without using traps.)

SNMP Operation in Messaging Server

The Messaging Server SNMP process is an SNMP subagent which, upon startup, registers itself with the platform's native SNMP master agent. SNMP requests from clients go to the master agent. The master agent then forwards any requests destined for the Messaging Server to the Messaging Server subagent process. The Messaging Server subagent process then processes the request and relays the response back to the client via the master agent. Figure 20–1 shows this process.

Figure 20–1 SNMP Information Flow



Configuring SNMP Support for Oracle Solaris 10

Note: SNMP is not supported on Solaris 11. For a workaround, contact Oracle Support.

By default, SNMP monitoring is disabled within Messaging Server. This default is chosen in an attempt to minimize the number of services presented by a default Messaging Server configuration. Do not interpret this default as meaning that there is a performance penalty incurred by using SNMP monitoring. Indeed, Messaging Server's SNMP support consumes very little resources and is intended to have minimal impact upon Messaging Server. The upshot of all of this is, of course, that one time configuration steps are required before using Messaging Server's SNMP support. Additionally, the default configuration of the platform's Net-SNMP master agent, **snmpd**, typically needs to be changed to run subagents such as Messaging Server's. This change is the topic of the next section.

Net-SNMP Configuration

Messaging Server's Net-SNMP based SNMP subagent uses the AgentX protocol to communicate with the platform's SNMP master agent (RFC 2741). The Net-SNMP master agent, **snmpd**, must be configured to permit the use of the AgentX protocol. To do this, ensure that the platform's **snmpd.conf** file contains the following line:

```
master agentx
```

If that line is not present, then add it and then restart the **snmpd** daemon. Sending a SIGHUP signal to the daemon is not sufficient. Once the **snmpd** daemon has been restarted, look for the UNIX domain socket which **snmpd** creates for AgentX communications. On Oracle Solaris and Linux systems, this socket by default appears as the special file **/var/agentx/master**. However, its location and name may be changed in the **snmpd.con** file.

The Oracle Solaris 10 **snmpd** configuration is as follows:

```
% cp /etc/sma/snmp/snmpd.conf /etc/sma/snmp/snmpd.conf.save
% cat >> /etc/sma/snmp/snmpd.conf
# Messaging Server's subagent requires the AgentX protocol
master agentx
^D
% cat >> /etc/sma/snmp/snmpd.conf
% ls -al /var/agentx/
srwxrwxrwx 1 root root 0 Aug 9 13:58 /var/agentx/master
```

Additionally, on Red Hat Enterprise Linux AS 3 systems, the default **snmpd.conf** file restricts the information which may be viewed by the "public" SNMP community. It is therefore necessary to either remove that restriction or to extend it to include the MIBs served out by Messaging Server's subagent. For initial testing, perform the later. This is accomplished by including the OID subtrees mib-2.27 and mib-2.28 in a view named "systemview" as shown in the following example. For actual deployment, each site must take their overall security policy into consideration. Note that the information provided by the SNMP subagent is "read only".

```
% cp /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.save
% cat >>/etc/snmp/snmpd.conf
# Messaging Server's subagent requires the AgentX protocol
master agentx
# Messaging Server's subagent exports mib-2.27 and .28
```

```
# Add the mib-2.27 and .28 OID subtrees to the systemview
view systemview included .1.3.6.1.2.1.27
view systemview included .1.3.6.1.2.1.28
^D
% /bin/service snmpd restart
% ls -al /var/agentx/master
srwxr-xr-x 1 root root 0 Aug 8 21:20 /var/agentx/master
```

If you are using SNMP v3 context names to distinguish between the MIBs of different instances of Messaging Server concurrently running on the same host computer, then you also need to configure at least one SNMP v3 username and password for use with your SNMP v3 queries.

Messaging Server Subagent Configuration

For basic operation of Messaging Server's SNMP subagent, you need only enable it and issue a one time manual start command. Henceforth, whenever Messaging Server is started or stopped, the subagent will likewise be started or stopped. The necessary commands to effect this configuration on both Oracle Solaris and Linux systems are as follows:

```
msconfig set snmp.enable 1
start-msg snmp
```

Once SNMP is running, you can test the subagent from the command line with the **snmpwalk** command. See the screen shots below for an example appropriate to Oracle Solaris and Linux. Note that the files **rfc2248.txt** and **rfc2249.txt** are copies of the Network Services and MTA MIBs. On Oracle Solaris systems, these files may also be found in the **/etc/sma/snmp/mibs/** directory under the names **NETWORK-SERVICES-MIB.txt** and **MTA-MIB.txt**. It is not necessary provide these files to the **snmpwalk** tool. However, doing so permits **snmpwalk** to print names for each of the MIB variables rather than their numeric object identifiers (OIDs).

Basic testing on Oracle Solaris:

```
% d=/opt/sun/comms/messaging64/examples/mibs /usr/sfw/bin/snmpwalk -v 1 -c public
\
-m +$D/rfc2248.txt:$D/rfc2249.txt 127.0.0.1 mib-2.27

NETWORK-SERVICES-MIB::applName.1 = STRING: /opt/sun/comms/messaging64 MTA on
mail.example.com
...
% D=/opt/sun/comms/messaging64/examples/mibs /usr/sfw/bin/snmpwalk -v 1 -c public
\
-m +$D/rfc2248.txt:$D/rfc2249.txt 127.0.0.1 mib-2.28

MTA-MIB::mtaReceivedMessages.1 = Counter32: 1452
MTA-MIB::mtaStoredMessages.1 = Gauge32: 21
...
```

Basic testing on Linux:

```
% export d=/opt/sun/messaging/examples/mibs
% /usr/bin/snmpwalk -v 1 -c public \
-m +$D/rfc2248.txt:$D/rfc2249.txt 127.0.0.1 mib-2.27
NETWORK-SERVICES-MIB::applName.1 = STRING: /opt/sun/messaging MTA on
mail.example.com
...
% /usr/bin/snmpwalk -v 1 -c public \
-m +$D/rfc2248.txt:$D/rfc2249.txt 127.0.0.1 mib-2.28
```

```
MTA-MIB::mtaReceivedMessages.1 = Counter32: 21278
MTA-MIB::mtaStoredMessages.1 = Gauge32: 7
...
```

Running as a Standalone SNMP Agent

Before configuring Messaging Server's SNMP subagent to run as a standalone SNMP agent, you must first decide which Ethernet interface and UDP port to use to listen for SNMP requests. By default, it listens on all available Ethernet interfaces by using UDP port 161. In most cases, you should change the port number so as to not interfere with the platform's SNMP master agent, **snmpd**. In some circumstances such as HA failover, you should change the Ethernet interface from all available interfaces - **INADDR_ANY** - to a specific interface identified by its IP address. These two concepts, Ethernet interface and UDP port, are controlled by the **snmp.listenaddr** and **snmp.port** options.

Once you have made choices for the Ethernet interface and UPD port, set the value of the **snmp.standalone** option to one and restart the subagent. Once restarted, it operates as an SNMP agent independent of **snmpd** and any subagents.

For example, to run as a standalone agent listening on UDP port 9161 of the Ethernet interface with IP address 10.53.1.37, issue the commands shown below.

Configuring to run as a standalone agent:

```
% ./msconfig set snmp.port 9161
% ./msconfig set snmp.listenaddr 10.53.1.37
% ./msconfig set snmp.standalone 1
% ./stop-msg snmp
% ./start-msg snmp
% ./snmpwalk -v 1 -c public 10.53.1.37:9161 .
SNMPv2-SMI::mib-2.27.1.1.2.1 = STRING: "/opt/sun/comms/messaging64 MTA on
mail.example.com"
...
```

Monitoring Multiple Instances of Messaging Server

Two techniques for monitoring multiple instances of Messaging Server running on the same host computer are herein discussed. The first technique, running the subagent in standalone mode, is well suited to high-availability failover (HA) configurations in which the individual instances of Messaging Server may dynamically move between host computers. The second technique, the use of SNMP v3 context names, has some limited benefit in situations where multiple instances of Messaging Server are confined to a single system and it is desirable to limit the number of IP addresses polled by SNMP monitoring software (for example, when licensing of the monitoring software has a per IP address cost component). This latter technique may also be used in HA failover settings but would require polling just as many IP addresses as the standalone mode technique.

Using Standalone Agents for High-availability Failover

In a high-availability failover setting where SNMP monitoring of Messaging Server is desired, it is recommended that you run Messaging Server's SNMP subagent as a standalone agent as described in "Running as a Standalone SNMP Agent". When the subagents are run in standalone mode, each HA instance of Messaging Server should have its **snmp.listenaddr** option set to the value of that instance's failover IP address. To simplify management, each instance should use the same UDP port, but that port should be distinct from those used by the **snmpd** daemons running on each of the

physical cluster hosts. Typically those daemons will be using UDP port 161 so explicitly specify a different port number with the **snmp.port** option.

When Messaging Server's SNMP support is configured as recommended here, a monitoring station can monitor each instance of Messaging Server through its failover IP address or hostname regardless of which physical cluster host the instance is running on. Moreover, you are assured that Messaging Server's standalone SNMP agents do not conflict with one another as each listens only on its own virtual Ethernet interface identified by that instance's unique failover IP address. (These virtual Ethernet interfaces are automatically created by the HA failover framework.) Owing to the careful selection of a UDP port, the agents do not conflict with the **snmpd** daemons running on systems within the cluster.

Distinguishing Multiple Instances Through SNMP v3 Context Names

While there is no downside to using Messaging Server's SNMP support in standalone mode as described in "Running as a Standalone SNMP Agent", it is recognized that some sites may prefer to use a more traditional subagent mode while still maintaining the capability of monitoring multiple instances of Messaging Server running concurrently on the same system. For instance, an SNMP monitoring system whose licensing model limits the number of IP addresses which may be polled. To achieve this goal, continue to run Messaging Server's SNMP subagent with **snmp.standalone** set to zero. Additionally, configure each instance of Messaging Server to use a distinct SNMP v3 context name by specifying a non-zero value for the **snmp.enablecontextname** option. If a context name different than the value of **base.defaultdomain** is desired, then set the desired name with the **snmp.contextname** option. Once each instance of Messaging Server's SNMP subagent is restarted, they can then be monitored with SNMP v3 queries that include the proper context names. The MIBs of two instances of Messaging Server running on the same system are distinguished by the instance's SNMP v3 context name and so no MIB object identifier (OID) conflicts will arise.

Messaging Server's Net-SNMP-based SNMP Subagent Options

Table 20–1, " SNMP Subagent Options" describes the options that apply only to Messaging Server's Net-SNMP based SNMP subagent. That subagent is used on Oracle Solaris platforms running Oracle Solaris 10 and later as well as Linux platforms. The options described in this section do not apply to the legacy SNMP subagent supplied for Oracle Solaris platforms running Oracle Solaris 9 and earlier operating systems.

Table 20–1 SNMP Subagent Options

Option (Default)	Description
snmp.enable (0)	The Messaging Server SNMP subagent only runs when this option is given a value of **1**, in which case Messaging Server automatically stops and starts the subagent as part of its normal startup and shutdown procedures. By default this option is set to zero, which disables operation of the subagent. Before enabling the subagent, ensure that the platform's master agent has been properly configured as described in "Running as a Standalone SNMP Agent".
snmp.standalone (0)	Messaging Server's SNMP support normally runs as a SNMP subagent, receiving SNMP requests through the platform's SNMP master agent, **snmpd**. This operational mode is the default and is selected by giving this option a value of 0. However, as described in "Running as a Standalone SNMP Agent", the subagent may run in a "standalone" mode whereby it operates as a SNMP agent independent of **snmpd**. When run in standalone mode, the subagent, now an SNMP agent, listens directly for SNMP requests on the Ethernet interface and UDP port specified by, respectively, the **snmp.listenaddr** and **snmp.port** options. To run in this standalone mode, specify a value of 1 for this option. Running in standalone mode does not interfere with other SNMP master or subagents running on the system.
snmp.listenaddr (INADDR_ANY)	Hostname or IP address of the Ethernet interface to listen for SNMP requests on when running in standalone mode. By default, all available interfaces are listened on. This corresponds to specifying the value **INADDR_ANY**. A specific interface may be selected by specifying either the IP address or hostname associated with that interface. The interface may be either a physical interface or a virtual interface. This option is ignored when **snmp.standalone** is set to 0.
snmp.cachettl (30)	Time to live (TTL) in seconds for cached monitoring data. This option controls how long the subagent will report the same monitoring data before refreshing that data with new information obtained from Messaging Server. With the exception of message loop information, data is cached for no longer than 30 seconds by default. Loop information, as determined by scanning for **.HELD** files, is updated only once every 10 minutes. That because of the resource cost of scanning all the on-disk message queues. Note that the subagent does not continually update its monitoring data: it is only updated upon receipt of an SNMP request and the cached data has expired (that is, outlived its TTL). If the TTL is set to 30 seconds and SNMP requests are made only every five minutes, then each SNMP request causes the subagent to obtain fresh data from Messaging Server. That is, data from Messaging Server is obtained only once every five minutes. If, on the other hand, SNMP requests are made every 10 seconds, then the subagent responds to some of those requests with cached data as old as 29 seconds. Messaging Server is polled only once every 30 seconds.
snmp.servertimeout (5)	The subagent determines the operational status of each monitored service by actually opening TCP connections to each service and undergoing a protocol exchange. This timeout value, measured in seconds, controls how long the subagent waits for a response to each step in the protocol exchange. By default, a timeout value of five seconds is used.

Table 20–1 (Cont.) SNMP Subagent Options

Option (Default)	Description
snmp.directoryscan (1)	Use this option to control whether or not the subagent performs scans of the on-disk message queues for **.HELD** message files and the oldest message files. That information corresponds to the **mtaGroupLoopsDetected**, **mtaGroupOldestMessageStored**, and **mtaGroupOldestMessageId** MIB variables. When this option has the value **1**, then a cache of this information is maintained and updated as needed. Sites with thousands of queued messages, that are not interested in these particular MIB variables should consider setting this option's value to 0.
snmp.enablecontextname (0)	The subagent has the ability to register its MIBs under an SNMP v3 *context name*. When this is done, the MIBs may only be requested by a SNMP v3 client that specifies the context name in its SNMP request. Use of context names allows multiple, independent subagents to register Network Services and MTA MIBs under the same OID tree (that is, under the same SNMP master agent). See "Monitoring Multiple Instances of Messaging Server" for further information. To enable the use of SNMP v3 context names, specify a value of 1 for this option. When that is done, the subagent defaults to using the value of the **base.defaultdomain** option for its context name. To use a different value for the context name, use the **snmp.contextname** option.
snmp.contextname (**base.defaultdomain**)	When the use of SNMP v3 context names has been enabled with **snmp.enablecontextname**, this option can be used to explicitly set the context name used by the subagent for its MIBs. The values supplied for this option are string values and must be appropriate for use as a SNMP v3 context name. This option is ignored when **snmp.enablecontextname** has the value 0.

Monitoring from an SNMP Client

The base OIDs for RFC 2788 (`http://www.faqs.org/rfcs/rfc2788.html`) and RFC 2789 (`http://www.faqs.org/rfcs/rfc2789.html`) are:

- mib-2.27 = 1.3.6.1.2.1.27
- mib-2.28 = 1.3.6.1.2.1.28

Point your SNMP client at those two OIDs and access as the "public" SNMP community.

To load copies of the MIBs into your SNMP client, ASCII copies of the MIBs are located in the *MessagingServer_home*/**lib/config-templates** directory under the file names **rfc2788.mib** and **rfc2789.mib**. For directions on loading those MIBs into your SNMP client software, consult the SNMP client software documentation. The **SnmpAdminString** data type used in those MIBs may not be recognized by some older SNMP clients. In that case, use the equivalent files **rfc2248.mib** and **rfc2249.mib** also found in the same directory.

SNMP Information from the Messaging Server

This section summarizes the Messaging Server information provided via SNMP.

For detailed information refer to the individual MIB tables in RFC 2788 (`http://www.faqs.org/rfcs/rfc2788.html`) and RFC 2789 (`http://www.faqs.org/rfcs/rfc2789.html`). Note that the RFC/MIB terminology refers to the messaging services (MTA, HTTP, and so on) as *applications* (**appl**), Messaging Server network connections as *associations* (**assoc**), and MTA channels as *MTAgroups* (**mtaGroups**).

On platforms where more than one instance of Messaging Server may be concurrently monitored, there may then be multiple sets of MTAs and servers in the **applTable**, and multiple MTAs in the other tables.

Note: The cumulative values reported in the MIBs (for example, total messages delivered, total IMAP connections, and so on) are reset to zero after a reboot.

Each site has different thresholds and significant monitoring values. A good SNMP client allows you to do trend analysis and then send alerts when sudden deviations from historical trends occur.

applTable

The **applTable** provides server information. It is a one-dimensional table with one row for the MTA and an additional row for each of the following servers, if enabled: WebMail HTTP, IMAP, POP, SMTP, and SMTP Submit. This table provides version information, uptime, current operational status (up, down, congested), number of current connections, total accumulated connections, and other related data.

Here is an example of data from **applTable** (mib-2.27.1.1).

```
applTable:

applName.1 = mailsrv-1 MTA on mailsrv-1.west.example.org (1)
applVersion.1 = 5.1
applUptime.1 = 7322 (2)
applOperStatus.1 = up (3)
applLastChange.1 = 7422 (2)
applInboundAssociations.1 = (5)
applOutboundAssociations.1 = (2)
applAccumulatedInboundAssociations.1 = 873
applAccumulatedOutboundAssociations.1 = 234
applLastInboundActivity.1 = 1054822 (2)
applLastOutboundActivity.1 = 1054222 (2)
applRejectedInboundAssociations.1 = 0 (4)
applFailedOutboundAssociations.1 = 17
applDescription.1 = Sun Java System Messaging Server 6.1
applName.2 1 = mailsrv-1 HTTP WebMail svr. mailsrv-1.example.org (1)
...
applName.3 = mailsrv-1 IMAP server on mailsrv-1.west.example.org
...
applName.4 = mailsrv-1 POP server on mailsrv-1.west.example.org
...
applName.5 = mailsrv-1 SMTP server on mailsrv-1.west.example.org
...
applName.6 = mailsrv-1 SMTP Submit server on mailsrv-1.west.example.org
...
```

Notes:

1. The application (**.appl***) suffixes (**.1**, **.2**, and so on) are the row numbers, **applIndex**. **applIndex** has the value 1 for the MTA, value 2 for the HTTP server, and so on. Thus, in this example, the first row of the table provides data on the MTA, the second on the POP server, and so on. The name after the equal sign is the name of the Messaging Server instance being monitored. In this example, the instance name is **mailsrv-1**.
2. These are SNMP TimeStamp values and are the value of **sysUpTime** at the time of the event. **sysUpTime**, in turn, is the count of hundredths of seconds since the SNMP master agent was started.

3. The operational status of the HTTP, IMAP, POP, SMTP, and SMTP Submit servers is determined by actually connecting to them by their configured TCP ports and performing a simple operation using the appropriate protocol (for example, a HEAD request and response for HTTP, a **HELO** command and response for SMTP, and so on). From this connection attempt, the status-**up** (1), **down** (2), or **congested** (4)-of each server is determined. Note that these probes appear as normal inbound connections to the servers and contribute to the value of the **applAccumulatedInboundAssociations** MIB variable for each server. For the MTA, the operational status is taken to be that of the Job Controller. If the MTA is shown to be up, then the Job Controller is up. If the MTA is shown to be down, then the Job Controller is down. This MTA operational status is independent of the status of the MTA's Service Dispatcher. The operational status for the MTA only takes on the value of up or down. Although the Job Controller does have a concept of "congested," it is not indicated in the MTA status.
4. For the HTTP, IMAP, and POP servers the **applRejectedInboundAssociations** MIB variable indicates the number of failed login attempts and not the number of rejected inbound connection attempts.

applTable Usage

Monitoring server status (**applOperStatus**) for each of the listed applications is key to monitoring each server.

If it has been a long time since the MTA last inbound activity as indicated by **applLastInboundActivity**, then something may be broken preventing connections. If **applOperStatus=2** (down), then the monitored service is down. If **applOperStatus=1** (up), then the problem may be elsewhere.

assocTable

This table provides network connection information to the MTA. It is a two-dimensional table providing information about each active network connection. Connection information is not provided for other servers. Here is an example of data from applTable (mib-2.27.2.1).

```
assocTable:

assocRemoteApplication.1.1 = 129.146.198.167 (1)
assocApplicationProtocol.1.1 = applTCPProtoID.25 (2)
assocApplicationType.1.1 = peerinitiator(3) (3)
assocDuration.1.1 = 400 (4)
...
```

Notes: In the **.x.y** suffix (1.1), **x** is the application index, **applIndex**, and indicates which application in the **applTable** is being reported on. In this case, the MTA. The **y** serves to enumerate each of the connections for the application being reported on.

1. The source IP address of the remote SMTP client.
2. This is an OID indicating the protocol being used over the network connection. **aplTCPProtoID** indicates the TCP protocol. The **.n** suffix indicates the TCP port in use and **.25** indicates SMTP which is the protocol spoken over TCP port 25.
3. It is not possible to know if the remote SMTP client is a user agent (UA) or another MTA. As such, the subagent always reports **peer-initiator**; **ua-initiator** is never reported.
4. This is an SNMP **TimeInterval** and has units of hundredths of seconds. In this example, the connection has been open for 4 seconds.

assocTable Usage

This table is used to diagnose active problems. For example, if you suddenly have 200,000 inbound connections, this table can let you know where they are coming from.

mtaTable

This is a one-dimensional table with one row for each MTA in the **applTable**. Each row gives totals across all channels (referred to as groups) in that MTA for select variables from the **mtaGroupTable**. Here is an example of data from **applTable** (mib-2.28.1.1).

```
mtaTable:

mtaReceivedMessages.1 = 172778
mtaStoredMessages.1 = 19
mtaTransmittedMessages.1 = 172815
mtaReceivedVolume.1 = 3817744
mtaStoredVolume.1 = 34
mtaTransmittedVolume.1 = 3791155
mtaReceivedRecipients.1 = 190055
mtaStoredRecipients.1 = 21
mtaTransmittedRecipients.1 = 3791134
mtaSuccessfulConvertedMessages.1 = 0 (1)
mtaFailedConvertedMessages.1 = 0
mtaLoopsDetected.1 = 0 (2)
```

Notes: The **.*x*** suffix (.1) provides the row number for this application in the **applTable**. In this example, **.1** indicates this data is for the first application in the **applTable**. Thus, this is data on the MTA.

1. Only takes on non-zero values for the conversion channel.
2. Counts the number of **.HELD** message files currently stored in the MTA's message queues.

mtaTable Usage

If mtaLoopsDetected is not zero, then there is a looping mail problem. Locate and diagnose the .HELD files in the MTA queue to resolve the problem.

If the system does virus scanning with a conversion channel and rejects infected messages, then **mtaSuccessfulConvertedMessages** gives a count of infected messages in addition to other conversion failures.

mtaGroupTable

This two-dimensional table provides channel information for each MTA in the **applTable**. This information includes such data as counts of stored (that is, queued) and delivered mail messages. Monitoring the count of stored messages, **mtaGroupStoredMessages**, for each channel is critical. When the value becomes abnormally large, mail is backing up in your queues.

Here is an example of data from **mtaGroupTable** (mib-2.28.2.1).

```
mtaGroupTable:

mtaGroupName.1.1 = tcp_intranet 1
...
mtaGroupName.1.2 = ims-ms
...
mtaGroupName.1.3 = tcp_local
mtaGroupDescription.1.3 = mailsrv-1 MTA tcp_local channel
```

```
mtaGroupReceivedMessages.1.3 = 12154
mtaGroupRejectedMessages.1.3 = 0
mtaGroupStoredMessages.1.3 = 2
mtaGroupTransmittedMessages.1.3 = 12148
mtaGroupReceivedVolume.1.3 = 622135
mtaGroupStoredVolume.1.3 = 7
mtaGroupTransmittedVolume.1.3 = 619853
mtaGroupReceivedRecipients.1.3 = 33087
mtaGroupStoredRecipients.1.3 = 2
mtaGroupTransmittedRecipients.1.3 = 32817
mtaGroupOldestMessageStored.1.3 = 1103
mtaGroupInboundAssociations.1.3 = 5
mtaGroupOutboundAssociations.1.3 = 2
mtaGroupAccumulatedInboundAssociations.1.3 = 150262
mtaGroupAccumulatedOutboundAssociations.1.3 = 10970
mtaGroupLastInboundActivity.1.3 = 1054822
mtaGroupLastOutboundActivity.1.3 = 1054222
mtaGroupRejectedInboundAssociations.1.3 = 0
mtaGroupFailedOutboundAssociations.1.3 = 0
mtaGroupInboundRejectionReason.1.3 =
mtaGroupOutboundConnectFailureReason.1.3 =
mtaGroupScheduledRetry.1.3 = 0
mtaGroupMailProtocol.1.3 = applTCPProtoID.25
mtaGroupSuccessfulConvertedMessages.1.3 = 03 2
mtaGroupFailedConvertedMessages.1.3 = 0
mtaGroupCreationTime.1.3 = 0
mtaGroupHierarchy.1.3 = 0
mtaGroupOldestMessageId.1.3 = <01IFBV8AT8HYB4T6UA@red.iplanet.com>
mtaGroupLoopsDetected.1.3 = 0 3
mtaGroupLastOutboundAssociationAttempt.1.3 = 1054222
```

Notes: In the **.x.y** suffix (example: **1.1, 1.2. 1.3**), **x** is the application index, **applIndex**, and indicates which application in the **applTable** is being reported on. In this case, the MTA. The **y** serves to enumerate each of the channels in the MTA. This enumeration index, **mtaGroupIndex**, is also used in the **mtaGroupAssociationTable** and **mtaGroupErrorTable** tables.

1. The name of the channel being reported on. In this case, the **tcp_intranet** channel.
2. Only takes on non-zero values for the conversion channel.
3. Counts the number of **.HELD** message files currently stored in this channel's message queue.

mtaGroupTable Usage

Trend analysis on **Rejected** and **Failed** might be useful in determining potential channel problems.

A sudden jump in the ratio of **mtaGroupStoredVolume** to **mtaGroupStoredMessages** could mean that a large junk mail is bouncing around the queues.

A large jump in **mtaGroupStoredMessages** could indicate unsolicited bulk email is being sent or that delivery is failing for some reason.

If the value of **mtaGroupOldestMessageStored** is greater than the value used for the undeliverable message notification times (notices channel option) this may indicate a message which cannot be processed even by bounce processing. Note that bounces are done nightly so you want to use **mtaGroupOldestMessageStored** > (maximum age + 24 hours) as the test.

If **mtaGroupLoopsDetected** is greater than 0, a mail loop has been detected.

mtaGroupAssociationTable

This is a three-dimensional table whose entries are indices into the **assocTable**. For each MTA in the **applTable**, there is a two-dimensional sub-table. This two-dimensional sub-table has a row for each channel in the corresponding MTA. For each channel, there is an entry for each active network connection which that channel has currently underway. The value of the entry is the index into the **assocTable** (as indexed by the entry's value and the **applIndex** index of the MTA being looked at). This indicated entry in the **assocTable** is a network connection held by the channel.

In simple terms, the **mtaGroupAssociationTable** table correlates the network connections shown in the **assocTable** with the responsible channels in the **mtaGroupTable**.

Here is an example of data from **mtaGroupAssociationTable** (mib-2.28.3.1).

```
mtaGroupAssociationTable:

mtaGroupAssociationIndex.1.3.1 = 1 1
mtaGroupAssociationIndex.1.3.2 = 2
mtaGroupAssociationIndex.1.3.3 = 3
mtaGroupAssociationIndex.1.3.4 = 4
mtaGroupAssociationIndex.1.3.5 = 5
mtaGroupAssociationIndex.1.3.6 = 6
mtaGroupAssociationIndex.1.3.7 = 7
```

Notes: In the **.x.y.z** suffix, **x** is the application index, **applIndex**, and indicates which application in the **applTable** is being reported on. In this case, the MTA. The **y** indicates which channel of the **mtaGroupTable** is being reported on. In this example, 3 indicates the **tcp_local channel**. The **z** serves to enumerate the associations open to or from the channel.

- The value here is an index into the **assocTable**. Specifically, **x** and this value become, respectively, the values of the **applIndex** and **assocIndex** indices into the **assocTable**. Or, put differently, this is saying that (ignoring the **applIndex**) the first row of the **assocTable** describes a network connection controlled by the **tcp_local** channel.

mtaGroupErrorTable

This is another three-dimensional table which gives the counts of temporary and permanent errors encountered by each channel of each MTA while attempting delivery of messages. Entries with index values of 4000000 are temporary errors while those with indices of 5000000 are permanent errors. Temporary errors result in the message being re-queued for later delivery attempts. Permanent errors result in either the message being rejected or otherwise returned as undeliverable.

Here is an example of data from **mtaGroupErrorTable** (mib-2.28.5.1).

```
mtaGroupErrorTable:

mtaGroupInboundErrorCount.1.1.4000000 1 = 0
mtaGroupInboundErrorCount.1.1.5000000 = 0
mtaGroupInternalErrorCount.1.1.4000000 = 0
mtaGroupInternalErrorCount.1.1.5000000 = 0
mtaGroupOutboundErrorCount.1.1.4000000 = 0
mtaGroupOutboundErrorCount.1.1.5000000 = 0
mtaGroupInboundErrorCount.1.2.4000000 1 = 0
...
mtaGroupInboundErrorCount.1.3.4000000 1 = 0
...
```

Notes:

- In the **.x.y.z** suffix, **x** is the application index, **applIndex**, and indicates which application in the **applTable** is being reported on. In this case, the MTA. The **y** indicates which channel of the **mtaGroupTable** is being reported on. In this example, 1 specifies the **tcp_intranet** channel, 2 the **ims-ms** channel, and 3 the **tcp_local** channel. Finally, the **z** is either 4000000 or 5000000 and indicates, respectively, counts of temporary and permanent errors encountered while attempting message deliveries for that channel.

mtaGroupErrorTable Usage

A large jump in error count may likely indicate an abnormal delivery problem. For instance, a large jump for a **tcp_** channel may indicate a DNS or network problem. A large jump for the **ims-ms** channel may indicate a delivery problem to the message store (for example, a partition is full, **stored** problem, and so on).

21

Short Message Service (SMS)

This chapter describes how to implement Short Message Service (SMS) in Unified Configuration for Oracle Communications Messaging Server.

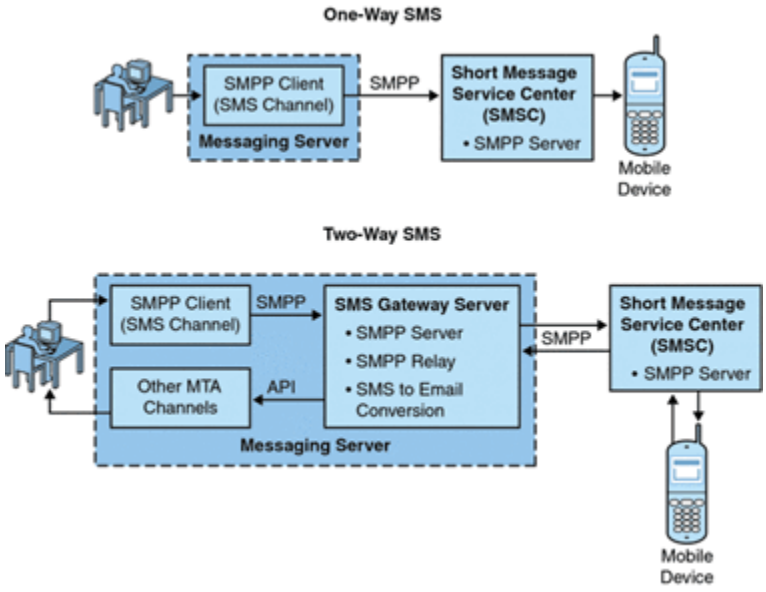
Introduction

Messaging Server implements email-to-mobile and mobile-to-email messaging using a Short Message Service (SMS). SMS can be configured to be either one-way (email-to-mobile only) or two-way (both email-to-mobile and mobile-to-email). To enable one-way service only, you must add and configure the SMS channel. To enable two-way service, you must add and configure the SMS channel, and in addition, configure the SMS Gateway Server.

For both one- and two-way SMS, the generated SMS messages are submitted to a Short Message Service Center (SMSC) using the Short Message Peer to Peer (SMPP) protocol. Specifically, the SMSC must provide a V3.4 or later SMPP server that supports TCP/IP.

Figure 21–1 illustrates the logical flow of messages for both one-way and two-way SMS.

Figure 21–1 Logical Flow For One-Way and Two-Way SMS



One-Way SMS

To enable one-way service, the Messaging Server implements an SMPP client (the MTA SMS channel) that communicates with remote SMSCs. The SMS channel converts enqueued email messages to SMS messages as described in "The Email to SMS Conversion Process" of multipart MIME messages as well as character set translation issues.

Operating in this capacity, the SMS channel functions as an (SMPP) External Short Message Entity (ESME).

Two-Way SMS

Two-Way SMS enables the mail server not only to send email to remote devices, but allows for receiving replies from the remote devices and for remote device email origination.

Enabling two-way SMS service requires both the MTA SMS channel (SMPP client), as explained in the previous topic, and the SMS Gateway Server. Messaging Server installs an SMS Gateway Server as part of its general installation process, which you must then configure. The SMS Gateway Server performs two functions:

- SMPP relay The SMS Gateway Server acts as a transparent SMPP client between the MTA SMS channel and SMSCs. However, in addition, while acting as a relay, the SMS Gateway Server generates unique SMS source addresses for relayed messages, and saves the message IDs returned by the remote SMSCs for later correlation with SMS notification messages.
- SMPP server The SMS Gateway Server acts as an SMPP server to receive mobile originated SMS messages, replies to prior email messages, and SMS notifications. The SMS Gateway Server extracts destination email addresses from the SMS messages using profiles that define the conversion process. Profiles also describe how to handle notification messages returned by remote SMSCs in response to previously sent email-to-mobile messages.

Note: Messaging Server does not support the two-way SMS on the Windows platform.

Requirements

This manual assumes that you have read Logica CMG's SMPP specification, and the SMPP documentation for your SMSC.

In order to implement SMS, the SMSC must support SMPP V3.4, or later, over TCP/IP and there must be TCP/IP connectivity between the host running Messaging Server and the SMSC.

See "SMS Gateway Server Storage Requirements" for storage planning information for the SMS Gateway Server.

SMS Channel Theory of Operation

The SMS channel is a multi-threaded channel which converts queued email messages to SMS messages and then submits them for delivery to an SMSC.

Directing Email to the Channel

When the SMS channel is configured as per "SMS Channel Configuration" purposes of discussion, let us assume that the host name **sms.example.org** is a host name associated with the channel. In that case, email is directed to the channel with an address of the form:

local-part@sms.example.org

in which **local-part** is either the SMS destination address (for example, a wireless phone number, pager ID, etc.) or an attribute-value pair list in the format:

/attribute1=value1/attribute2=value2/.../@sms.example.org

The recognized attribute names and their usages are given in Table 21–1. These attributes allow for per-recipient control over some channel options.

Table 21–1 SMS Attributes

Attribute	Attribute Value and Usage
ID	SMS destination address (for example, wireless phone number, pager ID, etc.) to direct the SMS message to. This attribute and associated value must be present.
FROM	SMS source address. Ignored when option **USE_HEADER_FROM=0**.
FROM_NPI	Use the specified **NPI** value. Ignored when option **USE_HEADER_FROM=0**.
FROM_TON	Use the specified **TON** value. Ignored when option **USE_HEADER_FROM=0**.
MAXLEN	The maximum, total bytes (that is, eight bit bytes) to place into the generated SMS message or messages for this recipient. The lower value of either **MAXLEN** and the value specified by the "MAX_MESSAGE_SIZE" channel option is used.
MAXPAGES	The maximum number of SMS messages to split the email message into for this recipient. The lower value of either **MAXPAGES** and the value specified by the "MAX_PAGES_PER_MESSAGE" channel option is used.
NPI	Specify a Numeric Plan Indicator (NPI) value for the destination SMS address specified with the **ID** attribute. See the description of the "DEFAULT_DESTINATION_NPI" channel option for information on the accepted values for this attribute. When this attribute is used, its value overrides the value given by the **DEFAULT_DESTINATION_NPI** channel option.
PAGELEN	Maximum number of bytes to place into a single SMS message for this recipient. The minimum of this value and that specified with the "MAX_PAGE_SIZE" channel option is used.
TO	Synonym for ID.
TO_NPI	Synonym for NPI.
TO_TON	Synonym for TON.
TON	Specify a Type of Number (TON) value for the destination SMS address given with the **ID** attribute. See the description of the "DEFAULT_DESTINATION_TON" channel option for information on the accepted values for this attribute. When this attribute is used, its value overrides the value given by the **DEFAULT_DESTINATION_TON** channel option.

Some example addresses:

```
123456@sms.example.org
/id=123456/@sms.example.org
/id=123456/maxlen=100/@sms.example.org
/id=123456/maxpages=1/@sms.example.org
```

See "Site-defined Address Validity Checks and Translations" for information on performing translations, validity checks, and other operations on the SMS destination address portion of the email address.

The Email to SMS Conversion Process

In order for email to be sent to a remote site, email must be converted to SMS messages that can be understood by the remote SMSCs. This section describes the process of converting an email message queued to the SMS channel to one or more SMS messages. As described below, options allow control over the maximum number of SMS messages generated, the maximum total length of those SMS messages, and the maximum size of any single SMS message. Only text parts (that is, MIME text content types) from the email message are used and the maximum number of parts converted may also be controlled.

Character sets used in the email message's header lines and text parts are all converted to Unicode and then converted to an appropriate SMS character set.

When there is no **SMS_TEXT** mapping table (see "Site-defined Text Conversions") an email message queued to the SMS channel receives the processing illustrated in Figure 21–2 and Figure 21–3.

Figure 21–2 Channel Email Processing

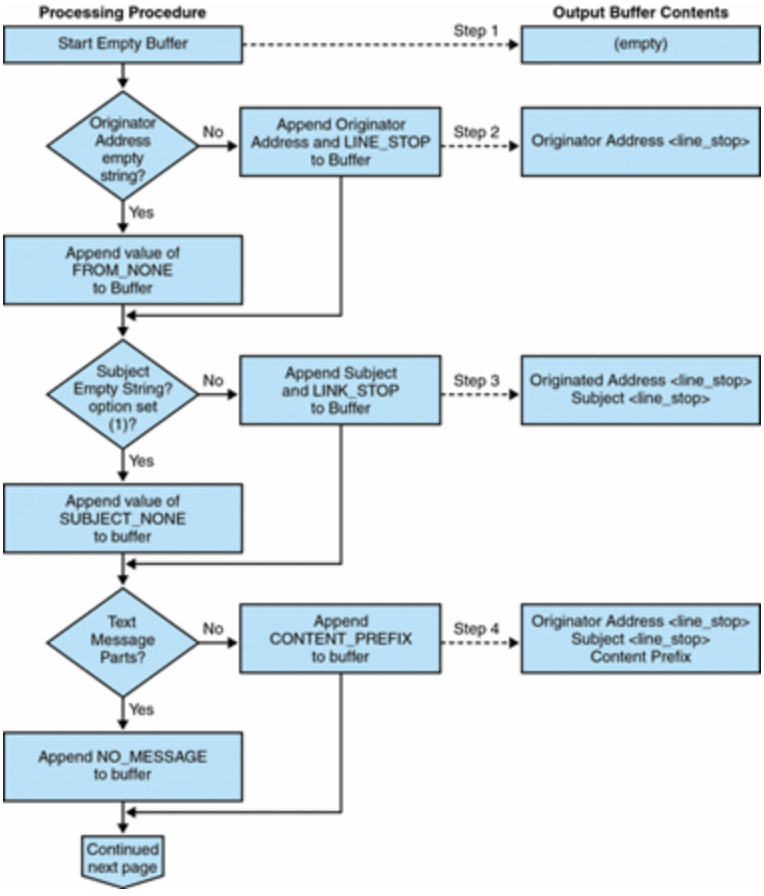
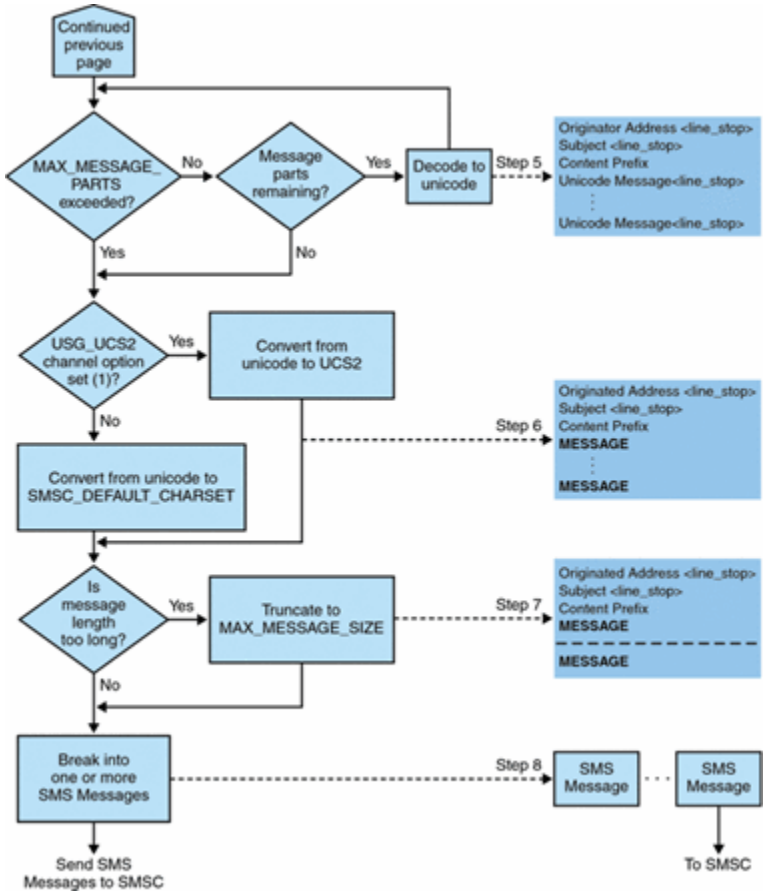


Figure 21–3 SMS Channel Email Processing (continued)



The following steps correspond to the numbered boxes in Figure 21–2:

1. An empty output buffer is started. The character set used for the buffer is Unicode.
2. The email message's originator address is taken from one of the following five sources, shown in decreasing order of preference:

   ```
   1. Resent-from:
   2. From:
   3. Resent-sender:
   4. Sender:
   5. Envelope From:
   ```

   If the originator address is an empty string, then the value of the "FROM_NONE" channel option is instead appended to the buffer. If, however, the originator address is a non-empty string, then the result of processing the "FROM_FORMAT" channel option, and the value of the **LINE_STOP** channel option are appended to the output buffer.
3. If a **Subject:** header line is not present or is empty, then the value of the "SUBJECT_NONE" option is appended to the output buffer. Otherwise, the result of processing the "SUBJECT_FORMAT" option, and the value of the "LINE_STOP" channel option are appended to the output buffer.
4. If there are no text message parts, then the value of the "NO_MESSAGE" channel option is appended to the output buffer. If there are text message parts, then the value of the "CONTENT_PREFIX" channel option is appended to the output buffer. Non-text message parts are discarded.

5. For each text part, while the **MAX_MESSAGE_PARTS** limit has not been reached, the text part is decoded to Unicode and appended to the buffer, along with the value of the **LINE_STOP** channel option.
6. The resulting output buffer is then converted from Unicode to either the SMSC's default character set or UCS2 (UTF-16). The SMSC's default character set is specified with the "SMSC_DEFAULT_CHARSET" option.
7. After being converted, it is then truncated to not exceed "MAX_MESSAGE_SIZE" bytes.
8. The converted string from "The Email to SMS Conversion Process" is then broken into one or more SMS messages, no single SMS message longer than **MAX_PAGE_SIZE** bytes. At most, "MAX_PAGES_PER_MESSAGE" SMS messages will be generated.

Note: As an email message may have multiple recipients, Step6 through Step 8 may need to be done for each recipient address which makes use of the **MAXLEN**, **MAXPAGES**, or **PAGELEN** attributes described in "Directing Email to the Channel".

Sample Email Message Processing

For example, with the channel's default settings, the email message:

```
From: John Doe
To: 1234567@sms.example.org
Subject: Today's meeting
Date: Fri, 26 March 2001 08:17
```

The staff meeting is at 14:30 today in the big conference room.

Would be converted to the SMS message:

jdoe@example.org (Today's meeting) The staff meeting is at 14:30 today in the big conference room.

A different set of option settings, that follows:

```
CONTENT_PREFIX=Msg:
FROM_FORMAT=From:${pa}
SUBJECT_FORMAT=Subj:$s
```

would instead produce:

From:John Doe Subj:Today's meeting Msg:The staff meeting is at 14:30 today in the big conference room.

The SMS Message Submission Process

Once an email message has been converted to one or more SMS messages, with possibly different sets for each recipient, the SMS messages are then submitted to the destination SMSC. The submissions are effected using SMPP V3.4 over TCP/IP. The hostname (**SMPP_SERVER**) of the SMPP server is taken to be the official host name associated with the SMS channel; the TCP port (**SMPP_PORT**) to use is specified with the **port** channel option.

When there are messages to process, the channel is started. The channel binds to the SMPP server as a transmitter, presenting the credentials specified with the **ESME_** channel options described in "SMPP Options". Table 21–2 lists the fields set in a

BIND_TRANSMITTER PDU (Protocol Data Unit), and gives their values:

Table 21–2 Fields in Generated in a BIND_TRANSMITTER PDU

Field	Channel Options	Value
system_id	ESME_SYSTEM_ID	Default value is an empty string.
password	ESME_PASSWORD	Default value is an empty string.
system_type	ESME_SYSTEM_TYPE	Default value is an empty string.
interface_version	n/a	0x34 indicating SMPP V3.4.
addr_ton	ESME_ADDRESS_TON	Default value is 0x00 indicating an unknown TON.
addr_npi	ESME_ADDRESS_NPI	Default value is 0x00 indicating an unknown NPI.
addr_range	ESME_IP_ADDRESS	Default value is an empty string.

Note that the channel is multithreaded. Depending on how much mail there is to send, the channel may have multiple dequeue threads running. (There can even be multiple channel processes running.) Each thread does a **BIND_TRANSMITTER** and then on that TCP/IP connection, sends all of the SMS messages it has to send, and then sends an **UNBIND,** and then closes the connection. No attempt is made to hold a connection open for a period of idle time for potential reuse. If the remote SMPP server sends back a throttle error, then an **UNBIND** is issued, the TCP/IP connection is closed, and a new connection and **BIND** established. It behaves similarly if the remote SMPP server sends an **UNBIND** before it is finished sending its SMS messages.

The SMS messages are then submitted using SMPP **SUBMIT_SM** PDUs. If a permanent error is returned (for example, **ESME_RINVDSTADR**), then the email message is returned as undeliverable. If a temporary error is returned, then the email message is re-enqueued for a later delivery attempt. To clarify, a permanent error is one for which the condition is likely to exist indefinitely and for which repeated delivery attempts will have no positive effect, such as invalid SMS destination addresses. Whereas, a temporary error is one for which the condition is likely to not exist in the near future, such as a server down or server congested condition.

If the **USE_HEADER_FROM** option has the value 1, then the source address for the submitted SMS message is set. The value used is derived from the originating email message and is chosen to be the most likely (email) address to which any replies should be directed. Accordingly, the source address taken from one of the following seven sources, shown in decreasing order of preference:

```
1. Resent-reply-to:
2. Resent-from:
3. Reply-to:
4. From:
5. Resent-sender:
6. Sender:
7. Envelope From:
```

Note that the **Resent-reply-to:** and **Reply-to:** header lines are only considered if the "USE_HEADER_REPLY_TO" option has the value 1. The default value is the value 0. As such, only items 4, 6, and 7 are considered by the default configuration. Finally, since the source address in an SMS message is limited to 20 bytes, the source address chosen will be truncated if it exceeds that limit.

Table 21–3 shows the mandatory fields set in a **SUBMIT_SM** PDU:

Table 21–3 Mandatory Fields in Generated SUBMIT_SM PDUs

Field	Value
service_type	"DEFAULT_SERVICE_TYPE" channel option; default value is an empty string.
source_addr_ton	"DEFAULT_SOURCE_TON" channel option; if **USE_HEADER_FROM=1**, then this field is usually forced to the value 0x05 indicating an alphanumeric TON; otherwise, the default value is 0x01 indicating an international TON.
source_addr_npi	"DEFAULT_SOURCE_NPI" channel option; default value is 0x00.
source_addr	"DEFAULT_SOURCE_ADDRESS" channel option if **USE_HEADER_FROM=0**; otherwise, an alphanumeric string representing the originator of the email message.
dest_addr_ton	TON addressing attribute or "DEFAULT_DESTINATION_TON" channel option; default value is 0x01 indicating an international TON.
dest_addr_npi	NPI addressing attribute or "DEFAULT_SOURCE_NPI" channel option; default value is 0x00 indicating an unknown NPI.
dest_addr	Destination SMS address derived from the local part of the email envelope To: address; see "Directing Email to the Channel".
esm_class	For one-way SMS, set to 0x03, indicating store and forward mode, default SMSC message type, and do not set reply path. For a two-way MSM message, set to 0x83.
protocol_id	0x00; unused for CDMA and TDMA; for GSM, 0x00 indicates no Internet, but SME-to-SME protocol.
priority_flag	0x00 for GSM & CDMA and 0x01 for TDMA, all indicating normal priority; See the description of the "DEFAULT_PRIORITY" channel option.
schedule_delivery_time	Empty string indicating immediate delivery.
validity_period	"DEFAULT_VALIDITY_PERIOD" channel option; default value is an empty string indicating that the SMSC's default should be used.
registered_delivery	0x00 indicating no registered delivery.
replace_if_present_flag	0x00 indicating that any previous SMS messages should not be replaced.
data_coding	0x00 for the SMSC's default character set; 0x08 for the UCS2 character set.
sm_default_msg_id	0x00 indicating not to use a pre-defined message.
sm_length	Length and content of the SMS message; see "The Email to SMS Conversion Process".
short_message	Length and content of the SMS message; see "The Email to SMS Conversion Process".

Table 21–4 shows the optional fields in a **SUBMIT_SM** PDU:

Table 21–4 Optional Fields in Generated SUBMIT_SM PDUs.

Field	Value
privacy	See the description of the "DEFAULT_PRIVACY" channel option; default is to not provide this field unless the email message has a **Sensitivity:** header line

Table 21–4 (Cont.) Optional Fields in Generated SUBMIT_SM PDUs.

Field	Value
sar_refnum	See the description of the "USE_SAR" channel option; default is to not provide these fields
sar_total	See **sar_refnum** above.
sar_seqnum	See **sar_refnum** above.

The channel remains bound to the SMPP server until either it has no more SMS messages to submit (the message queue is empty), or "MAX_PAGES_PER_BIND" has been exceeded. In the latter case, a new connection is made and bind operation performed if there remain further SMS messages to send.

Note that the SMS channel is multithreaded. Each processing thread in the channel maintains its own TCP connection with the SMPP server. For example, if there are three processing threads each with SMS messages to submit, then the channel will have three open TCP connections to the SMPP server. Each connection will bind to the SMPP server as a transmitter. Moreover, any given processing thread will only have one outstanding SMS submission at a time. That is, a given thread will submit an SMS message, then wait for the submission response (that is, **SUBMIT_SM_RESP** PDU) before submitting another SMS message.

Site-defined Address Validity Checks and Translations

Sites may want to apply validity checks or translations to SMS destination addresses encoded in the recipient email addresses described in "Directing Email to the Channel".

- Strip non-numeric characters (for example, translating 800.555.1212 to 8005551212)
- Prepend a prefix (for example, translating 8005551212 to +18005551212)
- Validate for correctness (for example, 123 is too short)

The first two tasks can be done specifically with the "DESTINATION_ADDRESS_NUMERIC" and "DESTINATION_ADDRESS_PREFIX" channel options. In general, all three of these tasks, and others can be implemented using mapping tables: either mapping table callouts in the rewrite rules or by means of a **FORWARD** mapping table. Using a mapping table callout in the rewrite rules will afford the most flexibility, including the ability to reject the address with a site-defined error response. The remainder of this section will focus on just such an approach - using a mapping table callout from the rewrite rules.

Let us suppose that destination addresses need to be numeric only, be 10 or 11 digits long, and be prefixed with the string "+1". This can be accomplished with the following rewrite rules:

```
sms.example.org ${X-REWRITE-SMS-ADDRESS,$U}@sms.example.org
sms.example.org $?Invalid SMS address
```

The first rewrite rule above calls out to the site-define mapping table named **X-REWRITE-SMS-ADDRESS**. That mapping table is passed the local part of the email address for inspection. If the mapping process decides that the local part is acceptable, then the address is accepted and rewritten to the SMS channel. If the mapping process does not accept the local part, then the next rewrite rule is applied. Since it is a $? rewrite rule, the address is rejected with the error text "Invalid SMS address".

The **X-REWRITE-SMS-ADDRESS** mapping table is shown below. It performs the necessary validation steps for local parts in either attribute-value pair list format or just a raw SMS destination address.

```
X-VALIDATE-SMS-ADDRESS

! Iteratively strip any non-numeric characters
$_*$[$ -/:-~]%* $0$2$R
! Accept the address if it is of the form 1nnnnnnnnnn or nnnnnnnnnn
! In accepting it, ensure that we output +1nnnnnnnnnn
1%%%%%%%%% +1$0$1$2$3$4$5$6$7$8$9$Y
%%%%%%%%% +1$0$1$2$3$4$5$6$7$8$9$Y
! We didn&rsquo;t accept it and consequently it&rsquo;s invalid
* $N

X-REWRITE-SMS-ADDRESS
*/id=$_*/* $C$0/id=$|X-VALIDATE-SMS-ADDRESS;$1|/$2$Y$E
*/id=$_*/* $N
* $C$|X-VALIDATE-SMS-ADDRESS;$0|$Y$E
* $N
```

With the above set up, be sure that "DESTINATION_ADDRESS_NUMERIC" option has the value 0 (the default). Otherwise, the "+" will be stripped from the SMS destination address.

Site-defined Text Conversions

Sites may customize Steps 1 - 6 described in "The Email to SMS Conversion Process" using a mapping table.

The name of the mapping table should be *SMS_Channel_TEXT* where *SMS_Channel* is the name of the SMS channel; for example, **SMS_TEXT** if the channel is named **sms** or **SMS_MWAY_TEXT** if the channel is named **sms_mway**.

Two types of entries may be made in this mapping table. However, before explaining the format of those entries, let it be made clear that an understanding of how to use mappings is essential in order to understand how to construct and use these entries. An example mapping table is given after the description of these two types of entries.

Now, the two types of entries are:

- Message Header Entries
- Message Body Entries

Message Header Entries

These entries specify which message header lines should be included in an SMS message and how they should be abbreviated or otherwise converted. Only if a header line is successfully mapped to a string of non-zero length by one of these entries will it be included in the SMS message being generated. Each entry has the format

H | *patternreplacement-text*

If a message header line matches the pattern then it will be replaced with the replacement text replacement-text using the mapping's pattern matching and string substitution facilities. The final result of mapping the header line will then be included in the SMS message provided that the metacharacter $Y was specified in the replacement text. If a header line does not match any pattern string, if it maps to a string of length zero, or if the $Y metacharacter is not specified in the replacement text, then the header line will be omitted from the SMS message. The two entries:

```
H|From:* F:$0$Y
H|Subject:* S:$0$Y
```

cause the **From:** and **Subject:** header lines to be included in SMS messages with **From:** and **Subject:** abbreviated as **F:** and **S:**. The entries:

```
H|Date:* H|D:$0$R$Y
H|D:*,*%19%%*:*:* H|D:$0$ $5:$6$R$Y
```

cause the **Date:** header line to be accepted and mapped such that, for instance, the header line

Date: Wed, 16 Dec 1992 16:13:27 -0700 (PDT)

will be converted to

D: Wed 16:13

Very complicated, iterative mappings may be built. Sites wanting to set up custom filters first need to understand how mappings work. The **H|** in the right-hand-side of the entry may be omitted, if desired. The **H|** is allowed in that side so as to cut down on the number of table entries required by sets of iterative mappings.

Message Body Entries

Body mappings are not supported.

Example SMS Mapping Table

See "Example SMS_TEXT Mapping Table" for an example **SMS_TEXT** mapping table. The numbers inside parentheses at the end of each line correspond to the item numbers in the section titled "Explanatory Text" that follows this table.

Example SMS_TEXT Mapping Table

```
SMS_TEXT

H|From:* H|F:$0$R$Y (1)
H|Subject:* H|S:$0$R$Y (1)
H|F:*&lt;*>* H|F:$1$R$Y ()
H|F:*(*)* H|F:$0$2$R$Y (2)
H|F:*"*"* H|F:$0$2$R$Y (3)
H|F:*@* H|F:$0$R$Y (4)
H|%:$ * H|$0:$1$R$Y (5)
H|%:*$ H|$0:$1$R$Y (5)
H|%:*$ $ * H|$0:$1$ $2$R$Y (6)
B|*--* B|$0-$1$R (7)
B|*..* B|$0.$1$R (7)
B|*!!* B|$0!$1$R (7)
B|*??* B|$0?$1$R (7)
B|*$ $ * B|$0$ $1$R (6)
B|$ * B|$0$R (5)
B|*$ B|$0$R (5)
```

Explanatory Text

The entries in the example **SMS_TEXT** mapping table above are explained below:

In the example above, the metacharacter $R is used to implement and control iterative application of the mappings. By iterating on these mappings, powerful filtering is achieved. For instance, the simple mappings to remove a single leading or trailing space (6) or reduce two spaces to a single space (7) become, when taken as a whole, a

filter which strips all leading and trailing spaces and reduces all consecutive multiple spaces to a single space. Such filtering helps reduce the size of each SMS message.

1. These two entries cause **From:** and **Subject:** header lines to be included in an SMS message. **From:** and **Subject:** are abbreviated as, respectively, **F:** and **S:**. Some of the other entries may have further effects on **From:** and **Subject:** header lines.

   This entry will reduce a **From:** header line containing a <...> pattern to only the text within the angle brackets. For example:

   F: "John C. Doe" <jdoe@example.org> (Hello) will be replaced with:**F: jdoe@example.org**

2. This entry will remove, inclusively, everything inside of a (...) pattern in a **From:** header line. For example:

   F: "John C. Doe" <jdoe@example.org> (Hello) will be replaced with:**F: "John C. Doe" <jdoe@example.org>**

3. This entry will remove, inclusively, everything inside of a "..." pattern in a **From:** header line. For example:

   F: "John C. Doe" <jdoe@example.org> (Hello) will be replaced with:**F: <jdoe@example.org> (Hello)**

4. This entry will remove, inclusively, everything to the right of an at-sign, @, in a **From:** header line. For example:

   F: "John C. Doe" <jdoe@example.org> (Hello) will be replaced with:**F: "John C. Doe" <jdoe@**

5. These four entries remove leading and trailing spaces from lines in the message header and body.

6. These two entries reduce two spaces to a single space in lines of the message header and body.

7. These four entries reduce double dashes, periods, exclamation and question marks to single occurrences of the matching character. Again, this helps save bytes in an SMS message.

The order of the entries is very important. For instance, with the given ordering, the body of the message **From:** header line:

From: "John C. Doe" (Hello)

will be reduced to:

jdoe

The steps taken to arrive at this are as follows:

1. We begin with the From: header line:

   From: "John C. Doe" (Hello)

   The pattern in the first mapping entry matches this and produces the result:

   F: "John C. Doe" (Hello)

   The $R metacharacter in the result string causes the result string to be remapped.

2. The mapping is applied to the result string of the last step. This produces:

   F: jdoe@example.org

   The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

3. Next, the mapping is applied producing:

   F: jdoe The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

4. Next, the mapping is applied producing:

   F:jdoe The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

5. Since no other entries match, the final result string:

   F:jdoe is incorporated into the SMS message.

Note: The **imsimta test -mapping** utility may be used to test a mapping table. For instance,

```
imsimta test -mapping -noimage_file -mapping_file=test.txt
Enter table name: SMS_TEXT
Input string: H|From: "John C. Doe" (Hello)
Output string: H|F:jdoe
Output flags: [0,1,2,89]
Input string: ^D
```

For further details, see the imsimta test utility. See *Messaging Server Reference* for more information on utility.

SMS Channel Configuration

This section gives directions on how to set up the SMS channel for both one-way (email-to-mobile) and two-way (email-to-mobile and mobile-to-email) functionality. The SMS channel is set up the same for both one-way and two-way functionality, with the exceptions noted in "Configuring the SMS Channel for Two-Way SMS" the section.

Adding an SMS Channel

Two steps are required to add an SMS channel to a Messaging Server configuration:

1. Adding the Channel Definition and Rewrite Rules.
2. Setting SMS Channel Options.

While there are no channel options which must be set in all situations, it is likely that one or more of the following options may need to be set:

- ESME_PASSWORD
- ESME_SYSTEM_ID
- MAX_PAGE_SIZE
- DEFAULT_SOURCE_TON
- DEFAULT_DESTINATION_TON

As described, the SMPP server's hostname or IP address and TCP port must be set with channel options.

You may configure more than one SMS channel, giving different characteristics to different SMS channels. See "Adding Additional SMS Channels" for further information on the use of multiple SMS channels.

Note for the instructions that follow: if you change channel definitions or rewrite rules, you must recompile.

Note also that the time before a channel change takes effect can differ depending on what the change is. Many channel option changes take effect in all channels started since the change was made, which may seem almost instantaneous since the Job Controller is often starting new channels. Some of the changes don't take effect until you recompile and restart the SMTP server. These options are processed as a message is enqueued to the channel and not when the channel itself runs.

Adding the Channel Definition and Rewrite Rules

To add the channel definition and rewrite rules, do the following:

To Add Channel Definition and Rewrite Rules

1. Before adding an SMS channel to the MTA's configuration, pick a name for the channel. The name of the channel may be either **sms** or **sms_x** where **x** is any case-insensitive string whose length is between one and thirty-six bytes. For example, **sms_mway**.
2. To add the channel definition, run **msconfig edit channels**. At the bottom of the channel definitions, add a blank line followed by the two lines:

   ```
   _channel-name_ port _p_ threaddepth _t_ \ backoff "pt2m" "pt5m" "pt10m"
   "pt30m" notices 1
   _smpp-host-name_
   ```

   where *channel-name* is the name you chose for the channel, *p* is the TCP port the SMPP server listens on, *t* is the maximum simultaneous number of SMPP server connections per delivery process, and *smpp-host-name* is the host name of the system running the SMPP server. For example, you might specify a channel definition as follows:

   ```
   sms_mway port 55555 threaddepth 20 \
   backoff "pt2m" "pt5m" "pt10m" "pt30m" notices 1
   smpp.example.org
   ```

   See "Controlling the Number of Simultaneous Connections" for instructions on how to calculate **threaddepth**.

   See "Adjusting the Frequency of Delivery Retries" for a discussion of the **backoff** and **notices** channel options.

   If you want to specify an IP address rather than a host name, for **smpp-host-name**, specify a domain literal. For example, if the IP address is 127.0.0.1, then specify [127.0.0.1] for **smpp-host-name**. Alternatively, consider using the "SMPP_SERVER" channel option.

   Note: The use of the **master** channel option is ignored if present.

3. Once the channel definition has been added, run **msconfig edit rewrite** and add a rewrite rule in this format:

   smpp-host-name$u@smpp-host-name

   For example,**smpp.example.org $u@smpp.example.org**
4. Save the changes.
5. Recompile the configuration with the **imsimta cnbuild** command.

6. Restart the SMTP server with the **imsimta restart dispatcher** command.
7. With the above configuration, email messages are directed to the channel by addressing them to **id@smpp-host-name** (for example, **123456@smpp.example.org**).

   See "The Email to SMS Conversion Process" for further information on addressing.
8. Optionally, if you want to hide the SMPP server's host name from users or associate other host names with the same channel, then add additional rewrite rules. For instance, to associate **host-name-1** and **host-name-2** with the channel, add the following to rewrite rules:

   ```
   host-name-1 $U%host-name-1@smpp-host-name
   host-name-2 $U%host-name-2@smpp-host-name
   ```

   For example, if the SMPP server's host name is **smpp.example.org** but you want users to address email to **id@sms.example.com**, then add the rewrite rule:**sms.example.com $U%sms.example.com@smpp.example.org**

   Note that the "SMPP_SERVER" and "SMPP_PORT" channel options will override the channel's official host name and **port** channel option settings. When the **SMPP_PORT** option is used, it is not necessary to also use the **port** option. The advantage of using these two options is that they can be put into effect and subsequently changed without needing to recompile the configuration. An additional use of the **SMPP_SERVER** option is described in the "Adding Additional SMS Channels" section.

Controlling the Number of Simultaneous Connections

The **threaddepth** channel option controls the number of messages to assign to each delivery thread within a delivery process. To calculate the total number of concurrent connections allowed, multiply the values of the two following options: **SMPP_MAX_CONNECTIONS** and **job_limit** (**SMPP_MAX_CONNECTIONS** * **job_limit**). The "SMPP_MAX_CONNECTIONS" option controls the maximum number of delivery threads in a delivery process. And, the **job_limit** option, for the Job Controller processing pool in which the channel is run, controls the maximum number of simultaneous delivery processes.

To limit the total number of concurrent connections, you must adjust appropriately either or both of these options. For instance, if the remote SMPP server allows only a single connection, then both **SMPP_MAX_CONNECTIONS** and **job_limit** must be set to **1**. When adjusting the values, it's preferable to allow **job_limit** to exceed **1**.

Setting SMS Channel Options

In general, a channel options contain site-specific options required for the operation of the channel. Channel options are not required for SMS. If you determine that some are necessary for your installation, set them using **msconfig**. For example:

```
msconfig
msconfig> set channel:sms_mway.options.profile GSM
msconfig# set channel:sms_mway.options.smsc_default_charset iso-8859-1
msconfig# set channel:sms_mway.options.use_ucs2 1
msconfig# write
```

See "Available Options" for a list of available SMS channel options and a description of each.

Available Options

The SMS channel contains several options which divide into six broad categories:

- *Email to SMS conversion*: Options which control the email to SMS conversion process.
- *SMS Gateway Server Option:* Gateway profile option.
- *SMS fields*: Options which control SMS-specific fields in generated SMS messages.
- *SMPP protocol*: Options associated with the use of the SMPP protocol over TCP/IP.
- *Localization*: Options which allow for localization of text fields inserted into SMS messages.
- *Miscellaneous*: Debug and logging options.

These options are summarized in Table 21–5, and are described more fully in the sections that follow.

Table 21–5 SMS Channel Options

Option	Description	Default
Email to SMS Conversion	-	-
GATEWAY_NOTIFICATIONS	Specify whether or not to convert email notification messages to SMS messages.	**0**
MAX_MESSAGE_PARTS	Max. number of message parts to extract from an email message.	**2**
MAX_MESSAGE_SIZE	Maximum number of bytes to extract from an email message.	**960**
MAX_PAGE_SIZE	Maximum number of bytes to put into a single SMS message.	**160**
MAX_PAGES_PER_MESSAGE	Max. number of SMS messages to break an email message into.	**6**
ROUTE_TO	Route SMS messages to the specified IP host name.	NA
SMSC_DEFAULT_CHARSET	The default character set used by the SMSC.	**US-ASCII**
USE_HEADER_FROM	Set the SMS source address.	**0**
USE_HEADER_PRIORITY	Control the use of priority information from the email message's header.	**1**
USE_HEADER_REPLY_TO	Control the use of **Reply-to:** header lines when generating SMS source addresses.	**0**
USE_HEADER_SENSITIVITY	Control the use of privacy information from the email message's header.	**1**
USE_UCS2	Use the UCS2 character set in SMS messages when applicable.	**1**
SMS Gateway Server Option	-	-
GATEWAY_PROFILE	Match the gateway profile name configured in the SMS Gateway Server's configuration file, **sms_gateway.cnf**.	NA
SMS Fields Options	-	-
DEFAULT_DESTINATION_NPI	Default NPI for SMS destination addresses.	**0x00**
DEFAULT_DESTINATION_TON	Default TON for SMS destination addresses.	**0x01**

Table 21–5 (Cont.) SMS Channel Options

Option	Description	Default
DEFAULT_PRIORITY	Default priority setting for SMS messages.	**0=GSM, CDMA1=TDMA**
DEFAULT_PRIVACY	Default privacy value flag for SMS messages.	**-1**
DEFAULT_SERVICE_TYPE	SMS application service associated with submitted SMS messages.	NA
DEFAULT_SOURCE_ADDRESS	Default SMS source address.	**0**
DEFAULT_SOURCE_NPI	Default NPI for SMS source addresses.	**0x00**
DEFAULT_SOURCE_TON	Default TON for SMS source addresses.	**0x01**
DEFAULT_VALIDITY_PERIOD	Default validity period for SMS messages.	NA
DESTINATION_ADDRESS_NUMERIC	Reduce the destination SMS address to only the characters 0 - 9.	**0**
DESTINATION_ADDRESS_PREFIX	Text string to prefix destination SMS addresses with.	N/A
PROFILE	SMS profile to use.	**GSM**
USE_SAR	Sequence multiple SMS messages using the SMS *sar_* fields.	**0**
SMPP Protocol Options	-	-
ESME_ADDRESS_NPI	ESME NPI to specify when binding to the SMPP server.	**0x00**
ESME_ADDRESS_TON	ESME TON to specify when binding to the SMPP server.	**0x00**
ESME_IP_ADDRESS	IP address of the host running Messaging Server.	NA
ESME_PASSWORD	Password to present when binding to the SMPP server.	NA
ESME_SYSTEM_ID	System identification to present to the SMSC when binding.	NA
ESME_SYSTEM_TYPE	System type to present to the SMSC when binding.	NA
MAX_PAGES_PER_BIND	Maximum number of SMS messages to submit during a single session with an SMPP server.	**1024**
REVERSE_ORDER	Transmission sequence of multi-part SMS messages.	**0**
SMPP_MAX_CONNECTIONS	Maximum number of simultaneous SMPP server connections.	**20**
SMPP_PORT	For one-way SMS, TCP port the SMPP server listens on. For two-way SMS, same TCP port used for the **LISTEN_PORT** for the SMPP relay.	NA
SMPP_SERVER	For one-way SMS, host name of the SMPP server to connect to.For two-way SMS, set to point to the host name or IP address of the SMS Gateway server. If using the SMPP relay's **LISTEN_INTERFACE_ADDRESS** option, then be sure to use the host name or IP address associated with the specified network interface address.	NA
TIMEOUT	Timeout for completion of reads and writes with the SMPP server.	**30**
Localization Options	-	-
CONTENT_PREFIX	Text to introduce the content of the email message.	**Msg**:
DSN_DELAYED_FORMAT	Formatting string for delivery delay notifications.	an empty string

Table 21–5 (Cont.) SMS Channel Options

Option	Description	Default
DSN_FAILED_FORMAT	Formatting string for delivery failure notifications.	see description
DSN_RELAYED_FORMAT	Formatting string for relay notifications.	see description
DSN_SUCCESS_FORMAT	Formatting string to successful delivery notifications.	see description
FROM_FORMAT	Text to display indicating the originator of the email message.	**$a**
FROM_NONE	Text to display when there is no originator.	NA
LANGUAGE	(i-default) Language group to select text fields from.	**i-default**
LINE_STOP	Text to place at the end of each line extracted from the email message.	space character
NO_MESSAGE	Text to indicate that the message had no content.	**]no message]**
SUBJECT_FORMAT	Text to display indicating the subject of the email message.	**$s**
SUBJECT_NONE	Text to display when there is no subject for the email message.	NA
Miscellaneous Options	-	-
DEBUG	Enable verbose debug output.	**6**
LISTEN_CONNECTION_MAX	Maximum number of concurrent, inbound TCP connections to allow across all SMPP relay and server instantiations.	**10,000**
LOG_PAGE_COUNT	Controls the value recorded in the **mail.log** file's message size field to be page count instead of blocks.	**0**

Email to SMS Conversion Options

The following options control the conversion of email messages to SMS messages. The value range for the options are in parenthesis. In general, a given email message may be converted into one or more SMS messages. See "The Email to SMS Conversion Process" for more information.

GATEWAY_NOTIFICATIONS

(*0 or 1*) Specifies whether or not to convert email notifications to SMS notifications. Email notification messages must conform to RFCs 1892, 1893, 1894. The default value is *0*.

When **GATEWAY_NOTIFICATIONS=0**, such notifications are discarded and are not converted to SMS notifications.

To enable the notifications to be converted to SMS notifications, set **GATEWAY_NOTIFICATIONS=1**. When the option set to *1*, the localization options (**DSN_*_FORMAT**) control which notification types (success, failure, delay, relayed) are converted into SMS messages and sent through the gateway. (If the notification type has a value of an empty string, then that type notification is not converted into SMS messages.)

MAX_MESSAGE_PARTS

(*integer*) When converting a multi-part email message to an SMS message, only the first **MAX_MESSAGE_PARTS** number of text parts will be converted. The remaining parts are discarded. By default, **MAX_MESSAGE_PARTS** is 2. To allow an unlimited number of message parts, specify a value of -1. When a value of 0 is specified, then no

message content will be placed into the SMS message. This has the effect of using only header lines from the email message (for example, **Subject:**) to generate the SMS message.

Note that an email message containing both text and an attachment will typically consist of two parts. Note further that only plain text message parts are converted. All other MIME content types are discarded.

MAX_MESSAGE_SIZE

(*integer, >= 10*) With this option, an upper limit may be placed on the total number of bytes placed into the SMS messages generated from an email message. Specifically, a maximum of **MAX_MESSAGE_SIZE** bytes will be used for the one or more generated SMS messages. Any additional bytes are discarded.

By default, an upper limit of 960 bytes is imposed. This corresponds to **MAX_MESSAGE_SIZE**=960. To allow any number of bytes, specify a value of zero.

The count of bytes used is made after converting the email message from Unicode to either the SMSC's default character set or UCS2. This means, in the case of UCS2, that a **MAX_MESSAGE_SIZE** of 960 bytes will yield, at most, 480 characters since each UCS2 character is at least two bytes long.

Note that the **MAX_MESSAGE_SIZE** and "MAX_PAGES_PER_MESSAGE" options both serve the same purpose: to limit the overall size of the resulting SMS messages. Indeed, "MAX_PAGE_SIZE"=960 and "MAX_PAGE_SIZE"=160 implies **MAX_PAGES_PER_MESSAGE=6**. So why are there two different options? So as to allow control of the overall size or number of pages without having to consider the maximal size of a single SMS message, **MAX_PAGE_SIZE**. While this may not be important in the channel options themselves, it is important when directing email to the channel described in "Directing Email to the Channel" and addressing attributes described in Table 21–1, " SMS Attributes".

Finally, note that the smaller of the two limits of **MAX_MESSAGE_SIZE** and **MAX_PAGE_SIZE * MAX_PAGES_PER_MESSAGE** is used.

MAX_PAGE_SIZE

(*integer, >= 10*) The maximum number of bytes to allow in a single SMS message is controlled with the **MAX_PAGE_SIZE** option. By default, a value of 160 bytes is used. This corresponds to **MAX_PAGE_SIZE=160**.

MAX_PAGES_PER_MESSAGE

(*integer, 1 - 255*) The maximum number of SMS messages to generate for a given email message is controlled with this option. In effect, this option truncates the email message, only converting to SMS messages that part of the email message which fits into **MAX_PAGES_PER_MESSAGE** SMS messages. See the description of the "MAX_PAGE_SIZE" option for further discussion.

By default, **MAX_PAGES_PER_MESSAGE** is set to the larger of **1** or "MAX_MESSAGE_SIZE" divided by "MAX_PAGE_SIZE".

ROUTE_TO

(*string*, *IP host name*, *1-64 bytes*) All SMS messages targeted to the profile will be rerouted to the specified IP host name using an email address of the form:

```
SMS-destination-address@route-to
```

where **SMS-destination-address** is the SMS message's destination address and the **route-to** is the IP host name specified with this option. The entire content of the SMS

message is sent as the content of the resulting email message. The **PARSE_RE_*** options are ignored.

> **Note:** Use of **PARSE_RE_*** and **ROUTE_TO** options are mutually exclusive. Use of both in the same gateway profile is a configuration error.

SMSC_DEFAULT_CHARSET

(*string*) With this option, the SMSC's default character set may be specified. Use the character set names given in the file

installation-directory/**lib/charsets.txt**

When this option is not specified, then US-ASCII is assumed. Note that the mnemonic names used in **charsets.txt** are defined in **charnames.txt** in the same directory.

When processing an email message, the header lines and text message parts are first decoded and then converted to Unicode. Next, the data is then converted to either the SMSC's default character set or UCS2, depending on the value of the "USE_UCS2" option and whether or not the SMS message contains at least one glyph not found in the default SMSC character set. Note that the UCS2 character set is a 16-bit encoding of Unicode and is often referred to as UTF-16.

USE_HEADER_FROM

(*integer, 0-2*) Set this option to allow the **From:** address to be passed to the SMSC. The value indicates where the **From:** address is taken from and what format it will have. Table 21–6 shows the allowable values and their meaning.

Table 21–6 USE_HEADER_FROM Values

Value	Description
0	SMS source address never set from the **From:** address. Use attribute-value pair found
1	SMS source address set to **from-local@from-domain**, where the From: address is: **@from-route:from-local@from-domain**
2	SMS source address set to *from-local*, where the From: address is: **@from-route:from-local@from-domain**

USE_HEADER_PRIORITY

(*0 or 1*) This option controls handling of RFC 822 **Priority:** header lines. By default, information from the **Priority:** header line is used to set the resulting SMS message's priority flag, overriding the default SMS priority specified with the "DEFAULT_PRIORITY" option. This case corresponds to **USE_HEADER_PRIORITY=1**. To disable use of the RFC 822 **Priority:** header line, specify **USE_HEADER_PRIORITY=0**.

See the description of the "DEFAULT_PRIORITY" option for further information on the handling the SMS priority flag.

USE_HEADER_REPLY_TO

(*0 or 1*) When **USE_HEADER_FROM =1**, this option controls whether or not a **Reply-to:** or **Resent-reply-to:** header line is considered for use as the SMS source address. By default, **Reply-to:** and **Resent-reply-to:** header lines are ignored. This corresponds to an option value of 0. To enable consideration of these header lines, use an option value of 1.

Note that RFC 2822 has deprecated the use of **Reply-to:** and **Resent-reply-to:** header lines.

USE_HEADER_SENSITIVITY

(*0 or 1*) The **USE_HEADER_SENSITIVITY** option controls handling of RFC 822 **Sensitivity:** header lines. By default, information from the **Sensitivity:** header line is used to set the resulting SMS message's privacy flag, overriding the default SMS privacy specified with the "DEFAULT_PRIVACY" option. This case, which is the default, corresponds to **USE_HEADER_SENSITIVITY=1**. To disable use of RFC 822 Sensitivity: header lines, specify **USE_HEADER_SENSITIVITY=0**.

See the description of the "DEFAULT_PRIVACY" option for further information on the handling the SMS privacy flag.

USE_UCS2

(*0 or 1*) When appropriate, the channel will use the UCS2 character set in the SMS messages it generates. This is the default behavior and corresponds to **USE_UCS2=1**. To disable the use of the UCS2 character set, specify **USE_UCS2=0**. See the description of the "SMSC_DEFAULT_CHARSET" option for further information on character set issues. Table 21–7 shows the allowable values and their meaning

Table 21–7 Valid Values for USE_UCS2

USE_UCS2	Result
1 (default)	The SMSC default character set will be used whenever possible. When the originating email message contains glyphs not in the SMSC default character set, then the UCS2 character set will be used.
0	The SMSC default character set will always be used. Glyphs not available in that character set will be represented by mnemonics (for example, "AE" for AE-ligature).

SMS Gateway Server Option

The following option describes the SMS Gateway Server.

GATEWAY_PROFILE

The name of the gateway profile in the SMS Gateway Server configuration file, **sms_gateway.cnf**.

SMS Options

The following options allow for specification of SMS fields in generated SMS messages.

DEFAULT_DESTINATION_NPI

(*integer, 0 - 255*) By default, destination addresses will be assigned an NPI (Numeric Plan Indicator) value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. Typical NPI values include those found in Table 21–8 that follows:

Table 21–8 Numeric Plan Indicator Values

Value	Description
0	Unknown

Table 21–8 (Cont.) Numeric Plan Indicator Values

Value	Description
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.164 (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*) By default, destination addresses will be assigned a TON (Type of Number) designator value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. Typical TON values include those found in Table 21–9 that follows:

Table 21–9 Typical TON Values

Value	Description
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) SMS messages have a mandatory priority field. The interpretation of SMS priority values is shown in Table 21–10.

Table 21–10 SMS Priority Values Interpreted for Each SMS Profile Type

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

With this option, the default priority to assign to SMS messages may be specified. When not specified, a default priority of 0 is used for **PROFILE=GSM** and **CDMA**, and a priority of **1** for "PROFILE"=TDMA.

Note that if "USE_HEADER_PRIORITY"=1 and an email message has an RFC 822 **Priority:** header line, then the priority specified in that header line will instead be used to set the priority of the resulting SMS message. Specifically, if **USE_HEADER_PRIORITY=0**, then the SMS priority flag is always set in accord with the **DEFAULT_PRIORITY** option and the RFC 822 **Priority:** header line is always ignored. If **USE_HEADER_PRIORITY=1**, then the originating email message's RFC 822 **Priority:** header line is used to set the SMS message's priority flag. If that header line is not present, then the SMS priority flag is set using the **DEFAULT_PRIORITY** option.

The mapping used to translate RFC 822 **Priority:** header line values to SMS priority flags is shown in Table 21–11.

Table 21–11 Mapping for Translating Priority Header to SMS Priority Flags

RFC 822	SMS Priority Flag	-	-
Priority: value	GSM	TDMA	CDMA
Third	Non-priority (0)	Bulk (0)	Normal (0)
Second	Non-priority (0)	Bulk (0)	Normal (0)
Non-urgent	Non-priority (0)	Bulk (0)	Normal (0)
Normal	Non-priority (0)	Normal (1)	Normal (0)
Urgent	Priority (1)	Urgent (2)	Urgent (2)

DEFAULT_PRIVACY

(*integer, -1, 0 - 255*) Whether or not to set the privacy flag in an SMS message, and what value to use is controlled with the **DEFAULT_PRIVACY** and "USE_HEADER_SENSITIVITY" options. By default, a value of -1 is used for **DEFAULT_PRIVACY**. Table 21–12 shows the result of setting the **DEFAULT_PRIVACY** and "USE_HEADER_SENSITIVITY" options to various values.

Table 21–12 Result of Values for DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY

DEFAULT_PRIVACY	USE_HEADER_SENSITIVITY	Result
-1	0	The SMS privacy flag is never set in SMS messages.
n >= 0	0	The SMS privacy flag is always set to the value n. RFC 822 **Sensitivity:** header lines are always ignored.
-1 (default)	1 (default)	The SMS message's privacy flag is only set when the originating email message has an RFC 822 **Sensitivity:** header line. In that case, the SMS privacy flag is set to correspond to the **Sensitivity:** header line's value. This is the default.
n >= 0	1	The SMS message's privacy flag is set to correspond to the originating email message's RFC 822 **Sensitivity:** header line. If the email message does not have a **Sensitivity:** header line, then the value of the SMS privacy flag is set to n.

The SMS interpretation of privacy values is shown in Table 21–13.

Table 21–13 SMS Interpretation of Privacy Values

Value	Description
0	Unrestricted
1	Restricted
2	Confidential
3	Secret
>= 4	Undefined

The mapping used to translate RFC 822 **Sensitivity:** header line values to SMS privacy values is shown in Table 21–14.

Table 21–14 Mapping Translation of Sensitivity Headers to SMS Privacy Values

RFC 822	SMS Privacy Value
Personal	1 (Restricted)
Private	2 (Confidential)
Company confidential	3 (Secret)

DEFAULT_SERVICE_TYPE

(*string, 0 - 5 bytes*) Service type to associate with SMS messages generated by the channel. By default, no service type is specified (that is, a zero length string). Some common service types are: **CMT** (cellular messaging), **CPT** (cellular paging), **VMN** (voice mail notification), **VMA** (voice mail alerting), **WAP** (wireless application protocol), and **USSD** (unstructured supplementary data services).

DEFAULT_SOURCE_ADDRESS

(*string, 0 - 20 bytes*) Source address to use for SMS messages generated from email messages. Note that the value specified with this option is overridden by the email message's originator address when **USE_HEADER_FROM=1**. By default, the value is disabled, that is, has a value of **0**.

DEFAULT_SOURCE_NPI

(*integer, 0 - 255*) By default, source addresses will be assigned an NPI value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the "DEFAULT_SOURCE_NPI" option for a table of typical NPI values.

DEFAULT_SOURCE_TON

(*integer, 0 - 255*) By default, source addresses will be assigned a TON designator value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the "DEFAULT_DESTINATION_TON" option for a table of typical TON values.

DEFAULT_VALIDITY_PERIOD

(*string, 0 - 252 bytes*) By default, SMS messages are not given a relative validity period; instead, they use the SMSC's default value. Use this option to specify a different relative validity period. Values may be specified in units of seconds, minutes, hours, or days. Table 21–15 specifies the format and description of the various values for this option:

Table 21–15 DEFAULT_VALIDITY_PERIOD Format and Values

Format	Description
nnn	Implicit units of seconds; for example, 604800
*nnn*s	Units of seconds; for example, 604800s
*nnn*m	Units of minutes; for example, 10080m
*nnn*h	Units of hours; for example, 168h
*nnn*d	Units of days; for example, 7d

A specification of 0, 0s, 0m, 0h, or 0d may be used to select the SMSC's default validity period. That is, when a specification of 0, 0s, 0m, 0h, or 0d is used, an empty string is specified for the validity period in generated SMS messages.

Note that this option does not accept values in UTC format.

DESTINATION_ADDRESS_NUMERIC

(0 or 1) Use this option to strip all non-numeric characters from the SMS destination address extracted from the email envelope **To:** address. For instance, if the envelope **To:** address is:

"(800) 555-1212"@sms.example.org

then it will be reduced to:

8005551212@sms.example.org

To enable this stripping, specify a value of 1 for this option. By default, this stripping is disabled which corresponds to an option value of 0. Note that when enabled, the stripping is done before any destination address prefix is added via the "DESTINATION_ADDRESS_PREFIX" option.

DESTINATION_ADDRESS_PREFIX

(*string*) In some instances, it may be necessary to ensure that all SMS destination addresses are prefixed with a fixed text string; for example, "+". This option may be used to specify just such a prefix. The prefix will then be added to any SMS destination

address which lacks the specified prefix. To prevent being stripped by the "DESTINATION_ADDRESS_NUMERIC" option, this option is applied after the **DESTINATION_ADDRESS_NUMERIC** option.

PROFILE

(*string*) Specify the SMS profiling to be used with the SMSC. Possible values are **GSM**, **TDMA**, and **CDMA**. When not specified, GSM is assumed. This option is only used to select defaults for other channel options such as "DEFAULT_PRIORITY" and "DEFAULT_PRIVACY".

USE_SAR

(0 or 1) Sufficiently large email messages may need to be broken into multiple SMS messages. When this occurs, the individual SMS messages can optionally have sequencing information added using the SMS sar_ fields. This produces a "segmented" SMS message which can be re-assembled into a single SMS message by the receiving terminal. Specify **USE_SAR=1** to indicate that this sequencing information is to be added when applicable. The default is to not add sequencing information and corresponds to **USE_SAR=0**.

When **USE_SAR=1** is specified, the "REVERSE_ORDER" option is ignored.

SMPP Options

The following options allow for specification of SMPP protocol options. The options with names beginning with the string "**ESME_**" serve to identify the MTA when it acts as an External Short Message Entity (ESME); that is, when the MTA binds to an SMPP server in order to submit SMS messages to the server's associated SMSC.

ESME_ADDRESS_NPI

(integer, 0 - 255) By default, bind operations will specify an ESME NPI value of zero indicating an unknown NPI. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the "DEFAULT_DESTINATION_NPI" option for a table of typical NPI values.

ESME_ADDRESS_TON

(integer, 0 - 255) By default, bind operations will specify an ESME TON value of 0. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the "DEFAULT_DESTINATION_TON" option for a table of typical TON values.

ESME_IP_ADDRESS

(*string, 0 - 15 bytes*) When binding to the SMPP server, the *BIND* PDU indicates that the client's (that is, ESME's) address range is an IP address. This is done by specifying a TON of 0x00 and an NPI of 0x0d. The value of the address range field is then set to be the IP address of the host running the SMS channel. Specify the IP address in dotted decimal format; for example, 127.0.0.1.

ESME_PASSWORD

(*string, 0 - 8 bytes*) When binding to the SMPP server, a password may be required. If so, then specify that password with this option. By default, a zero-length password string is presented.

ESME_SYSTEM_ID

(*string, 0 - 15 bytes*) When binding to the SMPP server, a system ID for the MTA may be supplied. By default, no system ID is specified (that is, a zero-length string is used). To specify a system ID, use this option.

ESME_SYSTEM_TYPE

(*string, 0 - 12 bytes*) When binding to the SMPP server, a system type for the MTA may be supplied. By default, no system type is specified (that is, a zero-length string is used).

MAX_PAGES_PER_BIND

(*integer, >= 0*) Some SMPP servers may limit the maximum number of SMS messages submitted during a single, bound session. In recognition of this, this option allows specification of the maximum number of SMS messages to submit during a single session. Once that limit is reached, the channel will unbind, close the TCP/IP connection, re-connect, and then rebind.

By default, a value of 1024 is used for **MAX_PAGES_PER_BIND**. Note that the channel will also detect **ESME_RTHROTTLED** errors and adjust **MAX_PAGES_PER_BIND** during a single run of the channel accordingly.

REVERSE_ORDER

(*0 or 1*) When an email message generates more than one SMS message, those SMS messages can be submitted to the SMSC in sequential order (**REVERSE_ORDER=0**), or reverse sequential order (**REVERSE_ORDER=1**). Reverse sequential order is useful for situations where the receiving terminal displays the last received message first. In such a case, the last received message will be the first part of the email message rather than the last. By default, **REVERSE_ORDER=1** is used.

Note that this option is ignored when "USE_SAR"=1 is specified.

SMPP_MAX_CONNECTIONS

(*integer, 1 - 50*) This option controls the maximum number of simultaneous SMPP connections per process. As each connection has an associated thread, this option also places a limit on the maximum number of "worker" threads per process. By default, **SMPP_MAX_CONNECTIONS=20**.

SMPP_PORT

(*integer, 1 - 65535*) The TCP port which the SMPP server listens on may be specified with either this option or the *port* channel option. This port number must be specified through either of these two mechanisms. If it is specified with both mechanisms, then the setting made with the **SMPP_PORT** option takes precedence. Note that there is no default value for this option.

For two-way SMS, make sure its the same port as the **LISTEN_PORT** for the SMPP relay.

SMPP_SERVER

(*string, 1 - 252 bytes*) For one-way SMS, by default, the IP host name of the SMPP server to connect to is the official host name associated with the channel; that is, the host name shown on the second line of the channel's definition in MTA's configuration. This option may be used to specify a different host name or IP address which will override that specified in the channel definition. When specifying an IP address, use dotted decimal notation; for example, 127.0.0.1.

For two-way SMS, set to point to the host name or IP address of the SMS Gateway Server. If using the SMPP relay's **LISTEN_INTERFACE_ADDRESS** option, then be sure to use the host name or IP address associated with the specified network interface address.

TIMEOUT

(*integer, >= 2*) By default, a timeout of 30 seconds is used when waiting for data writes to the SMPP server to complete or for data to be received from the SMPP server. Use the **TIMEOUT** option to specify, in units of seconds, a different timeout value. The specified value should be at least 1second.

Localization Options

In constructing SMS messages, the SMS channel has several fixed text strings it puts into those messages. These strings, for example, introduce the email's **From:** address and **Subject:** header line. With the channel options described in this section, versions of these strings may be specified for different languages and a default language for the channel then specified. "Example Language Specification Options" shows the language part of the option file:

Example Language Specification Options

```
msconfig
msconfig> set channel:mway_sms.options.language default-language
msconfig# set channel:mway_sms.options.from_prefix From:
msconfig# set channel:mway_sms.options.subject_prefix Subj:
msconfig# set channel:mway_sms.options.content_prefix Msg:
msconfig# set channel:mway_sms.options.line_stop
set channel:mway_sms.options.no_message [no message]
msconfig# set channel:mway_sms.options.reply_prefix re:
msconfig# write
msconfig> msconfig> set channel:mway_sms.options.language en
msconfig# set channel:mway_sms.options.from_prefix From:
msconfig# set channel:mway_sms.options.subject_prefix Subj:
msconfig# set channel:mway_sms.options.content_prefix Msg:
msconfig# set channel:mway_sms.options.line_stop
msconfig# set channel:mway_sms.options.no_message [no message]
msconfig# set channel:mway_sms.options.reply_prefix Re:
msconfig# write

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.

CONTENT_PREFIX

(*string, 0 - 252 bytes*) Text string to place in the SMS message before the content of the email message itself. Default global value is the US-ASCII string "**Msg:**".

DSN_DELAYED_FORMAT

(string, 0-256 characters) Formatting string for delivery delay notifications. By default, an empty string is used for this option, thereby inhibiting the conversion to SMS of delay notifications. Note that "GATEWAY_NOTIFICATIONS" must be set to **1** for this option to be in effect. This option is ignored when **GATEWAY_NOTIFICATIONS=0**.

DSN_FAILED_FORMAT

(string, 0-256 characters) Formatting string for permanent delivery failure notifications. The default value of this option is the string:

```
Unable to deliver your message to $a; no further delivery attempts will be
made.
```

To inhibit conversion of failure notifications, specify an empty string for this option. Note that "GATEWAY_NOTIFICATIONS" must be set to **1** for this option to be in effect. This option is ignored when **GATEWAY_NOTIFICATIONS=0**.

DSN_RELAYED_FORMAT

(string, 0-256 characters) Formatting string for relay notifications. The default value is the string:

```
Your message to $a has been relayed to a messaging system which may not
provide a final delivery confirmation
```

To inhibit conversion of relay notifications, specify an empty string for this option. Note that "GATEWAY_NOTIFICATIONS" must be set to **1** for this option to be in effect. This option is ignored when **GATEWAY_NOTIFICATIONS=0**.

DSN_SUCCESS_FORMAT

(string, 0-256 characters) Formatting string for successful delivery notifications. The default value is the string:

```
Your message to $a has been delivered
```

To inhibit conversion of successful delivery notifications, specify an empty string for this option. Note that "GATEWAY_NOTIFICATIONS" must be set to **1** for this option to be in effect. This option is ignored when **GATEWAY_NOTIFICATIONS=0**.

FROM_FORMAT

(*string, 0 - 252 bytes*) Formatting template to format the originator information to insert into the SMS message. The default global value is the US-ASCII string "**$a**" which substitutes in the originator's email address. See "Formatting Templates" for more information.

FROM_NONE

(*string, 0 - 252 bytes*) Text string to place in the SMS message when there is no originator address to display. The default global value is an empty string.

Note that normally, this option will never be used as sites will typically reject email messages which lack any originator address.

LANGUAGE

(*string, 0 - 40 bytes*) The default language group to select text strings from. If not specified, then the language will be derived from the host's default locale specification. If the host's locale specification is not available or corresponds to "C", then i-default will be used. (i-default corresponds to "English text intended for an international audience.")

LINE_STOP

(*string, 0 - 252 bytes*) Text string to place in the SMS message between lines extracted from the email message. The default global value is the US-ASCII space character, " ".

NO_MESSAGE

(*string, 0 - 252 bytes*) Text string to place in the SMS message to indicate that the email message had no content. The default global value is the US-ASCII string "**[no message]"**.

SUBJECT_FORMAT

(*string, 0 - 252 bytes*) Formatting template to format the content of the **Subject:** header line for display in the SMS message. The global default value for this option is the US-ASCII string "**($s)**". See "Formatting Templates" for further details.

See the **SUBJECT_NONE** option for a description of the handling when there is no **Subject:** header line or the content of that header line is an empty string.

SUBJECT_NONE

(*string, 0 - 252 bytes*) Text string to display when the originating email message either has no **Subject:** header line, or the **Subject:** header line's value is an empty string. The default global value for this option is the empty string.

DEBUG

(integer, bitmask) Enable debug output. The default value is *6* which selects warning and error messages. Any non-zero value enables debug output for the channel itself, the same as specifying **master_debug** on the channel definition.Table 21–16 defines the bit values of the **DEBUG** bitmask.

Table 21–16 DEBUG Bitmask

Bit	Value	Description
0-31	*-1*	Extremely verbose output
0	*1*	Informational messages
1	*2*	Warning messages
3	*4*	Error messages
3	*8*	Subroutine call tracing
4	*16*	Hash table diagnostics
5	*32*	I/O diagnostics, receive

Table 21–16 (Cont.) DEBUG Bitmask

Bit	Value	Description
6	*64*	I/O diagnostics, transmit
7	*128*	SMS to email conversion diagnostics (mobile originate and SMS notification)
8	*256*	PDU diagnostics, header data
9	*512*	PDU diagnostics, body data
10	*1024*	PDU diagnostics, type-length-value data
11	2048	Option processing; sends all option settings to the log file.

Formatting Templates

The formatting templates specified with "FROM_FORMAT" and "SUBJECT_FORMAT" and all the **DSN_*** channel options are UTF-8 strings which may contain a combination of literal text and substitution sequences. Assuming the sample email address of

Jane Doe <user@example>

The recognized substitution sequences are shown in Table 21–17.

Table 21–17 Substitution Sequences

Sequence	Description
$a	Replace with the local and domain part of the originator's email address (for example, "**user@example**")
$d	Replace with the domain part of the originator's email address (for example, "**domain**")
$p	Replace with the phrase part, if any, of the originator's email address (for example, "**Jane Doe**")
$s	Replace with the content of the **Subject:** header line
$u	Replace with the local part of the originator's email address (for example, "**user**")
\x	Replace with the literal character "**x**"

For example, the formatting template

From: $a

produces the text string

From: user@example

The construct,

${*x*y:alternate text}

may be used to substitute in the text associated with the sequence **x**. If that text is the empty string, the text associated with the sequence **y** is instead used. And, if that text is the empty string, to then substitute in the alternate text. For example, consider the formatting template

From: ${pa:unknown sender}

For the originator email address

John Doe <jdoe@example.org>

which has a phrase part, the template produces:

From: John Doe

However, for the address

jdoe@example.org

which lacks a phrase, it produces

From: jdoe@example.org

And for an empty originator address, it produces

From: unknown sender

Adding Additional SMS Channels

You may configure the MTA to have more than one SMS channel. There are two typical reasons to do this:

1. To communicate with different SMPP servers. This is quite straightforward: just add an additional SMS channel to the configuration, being sure to (a) give it a different channel name, and (b) associate different host names with it. For example,

   ```
   sms_mway port 55555 threaddepth 20
   smpp.example.org

   sms_ace port 777 threaddepth 20
   sms.ace.net
   ```

   Note that no new rewrite rule is needed. If there is no directly matching rewrite rule, Messaging Sever looks for a channel with the associated host name. For example, if the server is presented with **user@host.domain**, it would look for a channel of the name "host.domain". If it finds such a channel, it routes the message there. Otherwise, it starts looking for a rewrite rule for the ".domain" and if none is there, then for the dot (".") rule.

2. To communicate with the same SMPP server but using different channel options. To communicate with the same SMPP server, using different channel options, specify the same SMPP server in the "SMPP_SERVER" channel option for each channel definition. Using this mechanism is necessary since two different channels cannot have the same official host name (that is, the host name listed in the second line of the channel definition). To allow them to communicate with the same SMPP server, define two separate channels, with each specifying the same SMPP server in their "SMPP_SERVER" in their channel options. For example, you could have the following channel definitions,

   ```
   sms_mway_1 port 55555 threaddepth 20
   SMS-DAEMON-1

   sms_mway_2 port 55555 threaddepth 20
   SMS-DAEMON-2
   ```

   and rewrite rules,

   ```
   sms-1.example.org $u%sms-1.example.org@SMS-DAEMON-1
   sms-2.example.org $U%sms-2.example.org@SMS-DAEMON-2
   ```

   Then, to have them both use the same SMPP server, each of these two channels would specify "SMPP_SERVER"**=smpp.example.org** in their channel options.

Adjusting the Frequency of Delivery Retries

When an SMS message cannot be delivered owing to temporary errors (for example, the SMPP server is not reachable), the email message is left in the delivery queue and retried again later. Unless configured otherwise, the Job Controller will not re-attempt delivery for an hour. For SMS messaging, that is likely too long to wait. As such, it is recommended that the backoff channel option be used with the SMS channel to specify a more aggressive schedule for delivery attempts. For example,

```
sms_mway port 55555 threaddepth 20 \
backoff "pt2m" "pt5m" "pt10m" "pt30m" notices 1
smpp.example.org
```

With the above settings, a redelivery attempt will be made at two minutes after the first attempt. If that then fails, then five minutes after the second attempt. Then ten minutes later and finally every thirty minutes. The **notices 1** channel option causes the message to be returned as undeliverable if it cannot be delivered after a day.

Sample One-Way Configuration (MobileWay)

The MTA SMS channel may be used with any SMPP V3.4 compatible SMPP server. For purposes of illustrating an example configuration, this section explains how to configure the SMS channel for use with a MobileWay SMPP server. MobileWay (http://www.mobilway.com) is a leading provider of global data and SMS connectivity. By routing your SMS traffic through MobileWay, you can reach SMS subscribers on most of the major SMS networks throughout the world.

When requesting an SMPP account with MobileWay, you may be asked to answer the following questions:

- IP address of your SMPP client: Supply the IP address of your Messaging Server system as seen by other domains on the Internet.
- Default validity period: This is the SMS validity period which MobileWay will use should a validity period not be specified in the SMS messages you submit. SMS messages which cannot be delivered before this validity period expires will be discarded. Supply a reasonable value (for example, 2 days, 7 days, etc.).
- Window size: This is the maximum number of SMS messages your SMPP client will submit before it will stop and wait for responses from the SMPP server before submitting any further SMS messages. You must supply a value of 1 message.
- Timezone: Specify the timezone in which your Messaging Server system operates. The timezone should be specified as an offset from GMT.
- Timeout: Not relevant to one-way SMS messaging.
- IP address and TCP port for outbind requests: Not relevant for one-way SMS messaging.

After supplying MobileWay with the answers to the above questions, they will provide you with an SMPP account and information necessary to communicate with their SMPP servers. This information includes

```
Account Address: a.b.c.d:p
Account Login: system-id
Account Passwd: secret
```

The Account Address field is the IP address, **a.b.c.d**, and TCP port number, P., of the MobileWay SMPP server you will be connecting to. Use these values for the "SMPP_SERVER" and "SMPP_PORT" channel options. The Account Login and Passwd are,

respectively, the values to use for the "ESME_SYSTEM_ID" and "ESME_PASSWORD" channel options. Using this information, your channel's options should be set as follows:

```
msconfig
msconfig> set channel:mway_sms.options.smpp_server a.b.c.d
msconfig# set channel:mway_sms.options.smpp_port p
msconfig# set channel:mway_sms.options.esme_system_id system-id
msconfig# set channel:mway_sms.options.esme_password secret
msconfig# write
```

Now, to interoperate with MobileWay you must make two additional option settings:

```
msconfig
msconfig> set channel:mway_sms.options.esme_address_ton 0x01
msconfig# set channel:mway_sms.options.default_destination_ton 0x01
msconfig# write
```

The rewrite rule can appear as:

sms.your-domain $u@sms.your-domain

And, the channel definition can appear as:

```
sms_mobileway
sms.your-domain
```

Once the channel options, rewrite rules, and channel definitions are in place, a test message may be sent. MobileWay requires International addressing of the form

+<country-code><subscriber-number>

For instance, to send a test message to the North American subscriber with the subscriber number (800) 555-1212, you would address your email message to

+18005551212@sms.your-domain

Debugging

To debug the channel, specify the **master_debug** channel option in the channel's definition. For example,

```
sms_mway port 55555 threaddepth 20 \
backoff "pt2m" "pt5m" "pt10m" "pt30m" notices 1 master_debug
```

With the **master_debug** channel option, basic diagnostic information about the channel's operation will be output to the channel's log file. For verbose diagnostic information about the SMPP transactions undertaken by the channel, also set the channel debug option to 1 by running:

msconfig set channel:mway_sms.options.debug -1

Configuring the SMS Channel for Two-Way SMS

See "SMS Channel Configuration" for general directions on configuring the SMS channel. Configure the SMS channel as though it will be talking directly to the remote SMSC, with the exceptions listed in Table 21–18.

Table 21–18 Two-Way Configuration Exceptions

Exception	Explanation
master channel option	Remove the **master** channel option, if present.It is no longer needed for SMS channel configuration.
SMPP_SERVER	Set to point to the host name of IP address of the SMS Gateway Server. If using the SMPP relay's **LISTEN_INTERFACE_ ADDRESS** option (see "Configuration Options"), then be sure to use the host name or IP address associated with the specified network interface address.
SMPP_PORT	Same TCP port as used for the **LISTEN_PORT** setting used to instantiate the SMPP relay (see "An SMPP Relay").
DEFAULT_SOURCE_ ADDRESS	Pick a value and then configure the remote SMSC to route this address back to the Gateway SMPP server. In the SMS channel's options, specify the chosen value with this option.
GATEWAY_PROFILE	Set to match the gateway profile name. See "A Gateway Profile".
USE_HEADER_FROM	Set to **0**.

All other channel configurations should be done as described in the SMS Channel documentation.

As mentioned in "Setting Up Bidirectional SMS Routing", the remote SMSC needs to be configured to route the SMS address, defined in the **DEFAULT_SOURCE_ ADDRESS** channel option, to the Gateway's SMPP server using the TCP port number specified with the **LISTEN_PORT** option. See "An SMPP Server" for how to specify the **LISTEN_PORT**.

Note that multiple SMS channels may use the same SMPP relay. Similarly, there need be only one SMPP server or gateway profile to handle SMS replies and notifications for multiple SMS channels. The ability to configure multiple relays, servers, and gateway profiles exists to effect different usage characteristics through configuration options.

SMS Gateway Server Theory of Operation

The SMS Gateway Server facilitates two-way SMS through mechanisms that allow mobile originated SMS messages to be matched to the correct email address.

Function of the SMS Gateway Server

The SMS Gateway Server simultaneously functions as both an SMPP relay and server. It may be configured to have multiple "instantiations" of each function. For instance, it may be configured to have three different SMPP relays, each listening on different TCP ports or network interfaces and relaying to different remote SMPP servers. Similarly, it may be configured to have four different SMPP servers, each listening on different combinations of TCP ports and network interfaces.

The SMS Gateway Server may be configured with zero or more gateway profiles for sending SMS messages to email. Each gateway profile describes which destination SMS addresses match the profile, how to extract the destination email addresses from SMS messages, and various characteristics of the SMS to email conversion process. Each SMS message presented to the SMS Gateway Server through either its SMPP relay or server are compared to each profile. If a match is found, then the message is routed to email.

Finally, the gateway profiles also describe how to handle notification messages returned by remote SMSCs in response to previous email-to-mobile messages.

Behavior of the SMPP Relay and Server

When acting as an SMPP relay, the SMS Gateway Server attempts to be as transparent as possible, relaying all requests from local SMPP clients on to a remote SMPP server and then relaying back the remote server's responses. There are two exceptions:

- When a local SMPP client submits a message whose SMS destination address matches one of the configured gateway profiles, the submitted SMS message is sent directly back to email; the SMS message is not relayed to a remote SMPP server.
- When a local or remote SMPP client submits a message whose SMS destination address matches a unique SMS source address previously generated by the SMPP relay, the SMS message is a reply to a previously relayed message. This reply is directed back to the originator of the original message.

Note that typically the SMS Gateway Server will be configured such that the unique SMS source addresses which it generates match one of the gateway profiles.

> **Note:** The SMS Gateway Server's SMPP relay is only intended for use with qualified, SMPP clients, that is, the Messaging Server's SMS channel. It is not intended for use with arbitrary SMPP clients.

When acting as an SMPP server, the SMS Gateway Server directs SMS messages to email for three circumstances:

- The SMS messages are mobile originated and match a gateway profile.
- The SMS messages are mobile originated and the SMS destination address matches a previously generated unique SMS source address.
- The SMS messages are SMS notifications which correspond to email-to-mobile messages previously relayed by the SMS Gateway Server's SMPP relay.

All other SMS messages are rejected by the SMPP server.

Remote SMPP to Gateway SMPP Communication

Remote SMPP clients communicate to the Gateway SMPP server with Protocol Data Units (PDUs). Remote SMPP clients emit request PDUs to which the Gateway SMPP server responds. The Gateway SMPP server operates synchronously. It completes the response to a request PDU before it processes the next request PDU from the connected remote SMPP client.

Table 21–19 lists the request PDUs the Gateway SMPP server handles, and specifies the Gateway SMPP server's response.

Table 21–19 SMPP Server Protocol Data Units

Request	SMPP Server Response
BIND_ TRANSMITTERBIND_ TRANSCEIVERUNBIND	Responded to with the appropriate response PDU. Authentication credentials are ignored.
OUTBIND	Gateway SMPP server sends back a **BIND_RECEIVER** PDU. Authentication credentials presented are ignored.

Table 21–19 (Cont.) SMPP Server Protocol Data Units

Request	SMPP Server Response
SUBMIT_SMDATA_SM	Attempts to match the destination SMS address with either a unique SMS source address or the **SELECT_RE** setting of a Gateway profile. If neither is matched, the PDU is rejected with an **ESME_RINVDSTADR** error.
DELIVER_SM	Attempts to find either the destination SMS address or the receipted message ID in the historical record. If neither is matched, returns the error **ESME_RINVMSGID**.
BIND_RECEIVER	Not supported. Returns a **GENERIC_NAK** PDU with an **ESME_RINVCMDID** error.
SUBMIT_MULTI	Not supported. Returns a **GENERIC_NAK** PDU with an ESME_RINVCMDID error.
REPLACE_SM	Not supported. Returns a **GENERIC_NAK** PDU with an ESME_RINVCMDID error.
CANCEL_SM	Not supported. Returns a **GENERIC_NAK** PDU with an ESME_RINVCMDID error.
QUERY_SM	Not supported. Returns a **GENERIC_NAK** PDU with an **ESME_RINVCMDID** error.
QUERY_LAST_MSGS	Not supported. Returns a **GENERIC_NAK** PDU with an **ESME_RINVCMDID** error.
QUERY_MSG_DETAILS	Not supported. Returns a **GENERIC_NAK** PDU with an **ESME_RINVCMDID** error.
ENQUIRE_LINK	Returns **ENQUIRE_LINK_RESP** PDU.
ALERT_NOTIFICATION	Accepted but ignored.

SMS Reply and Notification Handling

The SMS Gateway Server maintains a historical record of each SMS message relayed through its SMPP relays. The need to use historical data arises from the fact that when submitting an email message to SMS it is generally not possible to convert the email address of the message's originator to an SMS source address. Since any SMS replies and notifications will be directed to this SMS source address, a problem arises. This is resolved by using automatically generated, unique SMS source addresses in relayed messages. The remote SMSCs are then configured to route these SMS source addresses back to the Gateway SMPP server.

The historical data is represented as an in-memory hash table of message IDs and generated, unique SMS source addresses. This data along with the associated email origination data are also stored on disk. The disk based storage is a series of files, each file representing **HASH_FILE_ROLLOVER_PERIOD** seconds of transactions (the default is 30 minutes). Each file is retained for **RECORD_LIFETIME** seconds (the default is 3 days).

Each record has three components:

- Email origination data (such as, envelope From: and To: addresses). This data is supplied by the MTA SMS channel when it submits a message.
- The unique SMS source address generated by the SMPP relay and inserted into the relayed SMS message.
- The resulting receipted message ID returned by the remote SMSC's SMPP server when it accepts a submission.

Routing Process for SMS Replies

The Gateway SMPP relays and servers use historical records to handle SMS replies, notifications and mobile originated messages. When an SMS message is presented to the SMPP relay or server, the following routing process is followed:

1. The SMS destination address is compared against the historical record to see if there is a matching, unique SMS source address that the SMPP relay previously generated. If a match is found, see Step 6.
2. If there is no match, but the message is an SMS notification (SMPP **DELIVER_SM** PDU), then the receipted message ID, if present, is compared against the historical record. If a match is found, go to Step 8. The SMS Gateway Server actually allows these to be presented to either the SMPP relay or SMPP server.
3. If there is no match, then the destination SMS address is compared against the **SELECT_RE** option expressions for each configured gateway profile. If a match is found, then go to Step 9.
4. If there is no match and the SMS message was presented to the Gateway SMPP relay, then the message is relayed to the remote SMPP server.
5. If there is no match and the SMS message was presented to the Gateway SMPP server, then the message is determined to be an invalid message and an error response is returned in the SMPP response PDU. For email to SMS, a Non Delivery Notification (NDN) is eventually generated.
6. If a matching unique SMS source address was found, then the SMS message is further inspected to see if it is a reply or a notification message. To be a notification message it must be a **SUBMIT_SM** PDU with a receipted message ID. Otherwise, it is considered to be a reply.
7. If it is a reply then the SMS message is converted to an email message using the origination email information from the historical record.
8. If it is a notification, then the SMS message is converted to an email Delivery Status Notification (DSN) as per RFC 1892-1894. Note that the ESMTP **NOTIFY** flags (RFC 1891) of the original email message will be honored (For example, if the SMS message is a "success" DSN but the original email message requested only "failure" notifications, then the SMS notification will be discarded).
9. If the destination SMS address matches a **SELECT_RE** option in a configured gateway profile, then the SMS message is treated as a mobile originated message and converted back to email message as per the **PARSE_RE_n** rules for that gateway profile. If the conversion fails, then the SMS message is invalid and an error response is returned.

SMS Gateway Server Configuration

This section gives directions on how to set up the SMS Gateway Server for both email-to-mobile and mobile-to-email functionality.

Setting Up Bidirectional SMS Routing

The recommended way to set up bidirectional email and SMS routing between the MTA and SMSC is a three step process:

- Set the SMS Address Prefix-- Choose an SMS address prefix. Any prefix may be used, so long as it is ten characters or less.

- Set the Gateway Profile-- Reserve the prefix for use with the SMS Gateway Server (by setting the gateway profile).
- Configure the SMSC-- Configure the SMSC to route SMS destination addresses to the SMS Gateway SMPP server that start with the prefix. Mobile originated email will have only the prefix. Replies and notifications will have the prefix followed by exactly ten decimal digits.

Set the SMS Address Prefix

The source SMS addresses generated by the MTA SMS channel should be set to match the selected SMS address prefix. Do this by setting the following:

- MTA SMS channel options:

  USE_HEADER_FROM=0

  DEFAULT_SOURCE_ADDRESS=*prefix*

  The first setting causes the channel to not attempt to set the SMS source address from information contained in the email message. The second setting causes the SMS source address to be set (to the selected prefix) when it is not set from any other source.

- Recognize the prefix as an SMS destination address to accept and route to email. Do this by specifying the **SELECT_RE** gateway profile option as follows:

  SELECT_RE=*prefix*

Set the Gateway Profile

The SMS Gateway Server's gateway profile should then be set to make all relayed SMS source addresses unique. This is the default setting but may be explicitly set by specifying the gateway profile option **MAKE_SOURCE_ADDRESSES_UNIQUE=1**. This will result in relayed SMS source addresses of the form:

```
prefixnnnnnnnnnn
```

where **nnnnnnnnnn** will be a unique, ten digit decimal number.

Configure the SMSC

Finally, the SMSC should be configured to route all SMS destination addresses matching the prefix (either just the prefix, or the prefix plus a ten digit number) to the SMS Gateway Server's SMPP server. The regular expression for such a routing will be similar to:

```
prefix([0-9]{10,10}){0,1}
```

where **prefix** is the value of **DEFAULT_SOURCE_ADDRESS**, [0-9] specifies the allowed values for the ten digit number, {10, 10} specifies that there will be a minimum of ten digits and a maximum of ten digits, and {0, 1} specifies that there can be zero or one of the 10-digit numbers.

Enabling and Disabling the SMS Gateway Server

- To enable the SMS Gateway Server, the option **sms_gateway.enable** must be set to the value *1*. Use the following **msconfig** command to set it:

  msconfig set sms_gateway.enable 1

- To disable the gateway server, set **sms_gateway.enable** to the value **0**, using the following command:

  msconfig sms_gateway.enable 0

Starting and Stopping the SMS Gateway Server

After the SMS Gateway Server is enabled, it may be started and stopped with the following commands:

start-msg sms

and

stop-msg sms

SMS Gateway Server Configuration File

In order to operate, the SMS Gateway Server requires a configuration file. The configuration file is a Unicode text file encoded using UTF-8. The file can be an ASCII text file. The name of the file must be:

```
installation-directory/config/sms_gateway.cnf
```

Each option setting in the file has the following format:

```
option-name=option-value
```

Options that are part of an option group appear in the following format:

```
[group-type=group-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

Configuring Email-To-Mobile on the Gateway Server

To implement the email-to-mobile part of two-way SMS, you must configure the following:

- A Gateway Profile
- An SMPP Relay
- An SMPP Server

A Gateway Profile

To configure an email-to-mobile gateway profile, follow these steps:

To Configure an Email-to-mobile Gateway Profile

1. Add a gateway profile to the SMS Gateway Server configuration file. To add an option group, use the following format:

   ```
   [GATEWAY_PROFILE=profile_name]
   option-name-1=option-value-1
   option-name-2=option-value-2a
   ...
   option-name-n=option-value-n
   ```

The length of the gateway profile name, **profile_name** in the preceding format, must not exceed 11 bytes. The name must be the same as the name for the **GATEWAY_PROFILE** channel option in the SMS channel option file. The name is case insensitive. See "Available Options" for a list of the valid channel options.

2. Set the gateway profile options (for example, **SMSC_DEFAULT_CHARSET**) to match characteristics of the remote SMSC.
3. Set the other gateway profile options to match the SMS channel's email characteristics.

   See "Gateway Profile Options" for a complete description of gateway profile options.
4. Set the **CHANNEL** option.

   Set its value to the name of the MTA SMS channel. When a notification is sent to email through the gateway, the resulting email message will be enqueued to the MTA using this channel name.

An SMPP Relay

To configure an SMPP Relay, complete the following steps:

To Configure an SMPP Relay

1. Add an SMPP relay instantiation (option group) to the SMS Gateway Server's configuration file. To add an option group, use the following format:

   ```
   [SMPP_RELAY=relay_name]
   option-name-1=option-value-1
   option-name-2=option-value-2
   ...
   option-name-n=option-value-n
   ```

   Any name may be used for the relay's name. All that matters is that the name not be used for any other SMPP relay instantiation within the same configuration file.
2. Set the **LISTEN_PORT** option.

   The value used for the SMS channel's **SMPP_PORT** option must match that used for the relay's **LISTEN_PORT** option. For the **LISTEN_PORT**, select a TCP port number which is not used by any other SMPP relay or server instantiation nor by any other server running on the same computer.
3. Set the **SERVER_HOST** option.

   The relay's **SERVER_HOST** option should give the host name for the remote SMSC's SMPP server. An IP address may be used in place of a host name.
4. Set the **SERVER_PORT** option.

   The relay's **SERVER_PORT** option should give the TCP port for the remote SMSC's SMPP server. See "SMPP Relay Options" for a complete description of all SMPP relay options.

An SMPP Server

To configure an SMPP server, complete the following steps:

To Configure an SMPP Server

1. Add an SMPP server instantiation (option group) to the SMS Gateway Server's configuration file. To add an option group, use the following format:

```
[SMPP_SERVER=server_name]
option-name-1=option-value-1
option-name-2=option-value-2...
option-name-n=option-value-n
```

Any name may be used for the server's name. All that matters is that the name not be used for any other SMPP server instantiation within the same configuration file.

2. Set *LISTEN_PORT* option. Select a TCP port number which is not being used by any other server or relay instantiation. Additionally, the port number must not be being used by any other server on the same computer. The remote SMSC needs to be configured to route notifications via SMPP to the SMS Gateway Server system using this TCP port. See "SMPP Server Options" for a complete description of all SMPP server options

Configuring Mobile-to-Email Operation

To configure mobile-to-email functionality, two configuration steps must be performed:

- Configure a Mobile-to-Email Gateway Profile
- Configure a Mobile-Email SMPP Server

Note that multiple gateway profiles may use the same SMPP server instantiation. Indeed, the same SMPP server instantiation may be used for both email-to-mobile and mobile-to-email applications.

Configure a Mobile-to-Email Gateway Profile

For mobile origination, a gateway profile provides two key pieces of information: how to identify SMS messages intended for that profile and how to convert those messages to email messages. Note that this profile can be the same one used for email-to-mobile with the addition of the *SELECT_RE* option.

To configure the gateway profile, follow these steps:

To Configure the Gateway Proflie

1. Add a gateway profile (option group) to the SMS Gateway Server's configuration file.

   To add an option group, use the following format:

   ```
   [GATEWAY_PROFILE=profile_name]
   option-name-1=option-value-1
   option-name-2=option-value-2
   ...
   option-name-n=option-value-n
   ```

   Any name of 11 characters or less may be used for the profile's name. All that matters is that it is not already used for another gateway profile within the same configuration file.

2. Set the **SELECT_RE** option must be specified for each gateway profile.

   The value of this option is an ASCII regular expression with which to compare SMS destination addresses. If an SMS destination address matches the regular expression, then the SMS message is sent through the gateway to email using the characteristics described by the matching profile. It is important to note that it is possible to configure multiple gateway profiles which have overlapping sets of SMS addresses (for example, a profile which matches the address 000 and another

which matches any other three-digit address). However, so doing should be avoided as an SMS message will be passed off to only one gateway profile: the first one which matches. And, moreover, the order in which they are compared is undefined.

3. Set the **CHANNEL** option.

   Its value should be the name of the MTA's SMS channel. See "Gateway Profile Options" for a complete description of all mobile origination options.

Configure a Mobile-Email SMPP Server

Adding an SMPP server is the same as for the email-to-mobile SMPP server (see "An SMPP Server").

The remote SMSC needs to be configured to route SMS traffic to the gateway SMPP server. To do this, the SMS destination address used by the SMSC to route mobile-to-email traffic should be the value set for the gateway profile option **SELECT_RE**.

For example, if the SMS address 000 is to be used for mobile-to-email traffic, then the SMSC needs to be configured to route traffic for the SMS destination address 000 to the gateway SMPP server. The gateway profile should use the option setting **SELECT_RE**=000.

Configuration Options

The SMS Gateway Server configuration file options are detailed in this section. The tables that follow list all the available configuration options with a brief description of each. There is a table each for global options, SMPP relay options, SMPP server options, and SMS Gateway Server profile options.

In the subsections that follow, complete descriptions are given for all the available configuration options. The subsections are:

- Global Options

  Global options must be placed at the top of the configuration file, before any option groups. The remaining options must appear within option groups.

- SMPP Relay Options
- SMPP Server Options
- Gateway Profile Options

Global Options

The SMS Gateway Server presently has three categories of global options:

- Thread Tuning Options
- Historical Data Tuning
- Miscellaneous

All global options must be specified at the top of the configuration file, before any option groups are specified. Table 21–20 lists all global configuration options.

Table 21–20 Global Options

Option	Default	Description
DEBUG	**6**	Selects the types of diagnostic output generated.
HISTORY_FILE_DIRECTORY	no default value	Absolute directory path for files of historical data.
HISTORY_FILE_MODE	**0770**	Permissions for files of historical data.
HISTORY_FILE_ROLLOVER_PERIOD	**30 mins**	Maximum length of time to write to the same file of historical data.
LISTEN_CONNECTION_MAX	**10,000**	Maximum number of concurrent inbound connections across all SMPP relay and server instantiations.
RECORD_LIFETIME	**3 days**	Lifetime of a record in the historical data archive.
THREAD_COUNT_INITIAL	**10 threads**	Initial number of worker threads.
THREAD_COUNT_MAXIMUM	**50 threads**	Maximum number of worker threads.
THREAD_STACK_SIZE	*64 Kb*	Stack size for each worker thread.

Thread Tuning Options

Each inbound TCP connection represents an SMPP session. The processing for a session is handled by a worker thread from a pool of threads. When the session processing needs to wait for completion of an I/O request, the worker thread parks the session and is given other work to perform. When the I/O request completes, the session is resumed by an available worker thread from the pool.

The following options allow for tuning of this pool of worker thread processes:

- THREAD_COUNT_INITIAL
- THREAD_COUNT_MAXIMUM
- THREAD_STACK_SIZE

THREAD_COUNT_INITIAL

(integer, > 0_)_ Number of threads to initially create for the pool of worker threads. This count does not include the dedicated threads used to manage the in-memory historical data (2 threads) nor the dedicated threads used to listen for incoming TCP connections (one thread per TCP port/interface address pair the SMS Gateway Server listens on). The default value is for **THREAD_COUNT_INITIAL** is 10 threads.

THREAD_COUNT_MAXIMUM

*(integer,***>= THREAD_COUNT_INITIAL***)* Maximum number of threads to allow for the pool of worker threads. The default value is 50 threads.

THREAD_STACK_SIZE

(integer, > 0_)_ Stack size in bytes for each worker thread in the pool of worker threads. The default value is 65,536 bytes (64 Kb).

Historical Data Tuning

When an SMS message is relayed, the message ID generated by the receiving, remote SMPP server is saved in an in-memory hash table. Along with this message ID, information about the original email message is also saved. Should that message ID subsequently be referenced by an SMS notification, this information may be retrieved.

The retrieved information can then be used to send the SMS notification to the appropriate email recipient.

The in-memory hash table is backed to disk by a dedicated thread. The resulting disk files are referred to as "history files". These history files serve two purposes: to save, in nonvolatile form, the data necessary to restore the in-memory hash table after a restart of the SMS Gateway Server, and to conserve virtual memory by storing potentially lengthy data on disk. Each history file is only written to for **HASH_FILE_ROLLOVER_PERIOD** seconds after which time it is closed and a new history file created. When a history file exceeds an age of **RECORD_LIFETIME** seconds, it is deleted from disk.

The following options allow for tuning historical files:

- HISTORY_FILE_DIRECTORY
- HISTORY_FILE_MODE
- HISTORY_FILE_ROLLOVER_PERIOD
- RECORD_LIFETIME

HISTORY_FILE_DIRECTORY

(string, absolute directory path) Absolute path to the directory to which to write the history files. The directory path will be created if it does not exist. The default value for this option is:

*DataRoot***/sms_gateway_cache/**

The directory used should be on a reasonably fast disk system and have more than sufficient free space for the anticipated storage; see "SMS Gateway Server Storage Requirements" to change this option to a more appropriate value.

HISTORY_FILE_MODE

(integer, octal value) File permissions to associated with the history files. By default, a value of 0770 (octal) is used.

HISTORY_FILE_ROLLOVER_PERIOD

(integer, seconds) The current history file is closed and a new one created every **HASH_FILE_ROLLOVER_PERIOD** seconds. By default, a value of 1800 seconds (30 minutes) is used.

RECORD_LIFETIME

(integer, seconds > 0_)_ Lifetime in seconds of a historical record. Records older than this lifetime will be purged from memory; history files older than this lifetime will be deleted from disk. By default, a value of 259,200 seconds (3 days) is used. Records stored in memory are purged in sweeps by a thread dedicated to managing the in-memory data. These sweeps occur every **HASH_FILE_ROLLOVER_PERIOD** seconds. Files on disk are purged when it becomes necessary to open a new history file.

Miscellaneous

This section describes the miscellaneous options.

DEBUG

(*integer, bitmask*) Enable debug output. The default value is **6** which selects warning and error messages.

Table 21–21 defines the bit values of the DEBUG bitmask.

Table 21–21 DEBUG Bitmask

Bit	Value	Descriptions
0-31	-1	Extremely verbose output.
0	1	Informational messages.
1	2	Warning messages.
3	4	Error messages.
3	8	Subroutine call tracing.
4	16	Hash table diagnostics.
5	32	I/O diagnostics, receive.
6	64	I/O diagnostics, transmit.
7	128	SMS to email conversion diagnostics (mobile originate and SMS notification).
8	256	PDU diagnostics, header data.
9	512	PDU diagnostics, body data.
10	1024	PDU diagnostics, type-length-value data.
11	2048	Option processing; sends all option settings to the log file.

LISTEN_CONNECTION_MAX

(*integer*, >= 0) The maximum number of concurrent, inbound TCP connections to allow across all SMPP relay and server instantiations. A value of 0 (zero) indicates that there is no global limit on the number of connections. There may, however, be per relay or server limits imposed by a given relay or server instantiation. Default: 10,000

SMPP Relay Options

The SMS Gateway Server can have multiple instantiations of its SMPP relay, each with different characteristics chief of which will be the TCP port and interface listened on. Put differently, for each network interface and TCP port pair the SMPP relay listens on, distinct characteristics may be ascribed. These characteristics are specified using the options described in this section.

Each instantiation should be placed within an option group of the form:

```
[SMPP_RELAY=relay-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

The string **relay-name** merely serves to differentiate this instantiation from other instantiations.

Table 21–22 lists the SMPP relay configuration options.

Table 21–22 SMPP Relay Options

Options	Default	Description
LISTEN_BACKLOG	**255**	Connection backlog for inbound SMPP client connections.
LISTEN_CONNECTION_MAX	no default value	Maximum number of concurrent inbound connections.
LISTEN_INTERFACE_ADDRESS	no default value	Network interface for inbound SMPP client connections.
LISTEN_PORT	no default value	TCP port for inbound SMPP client connections.
LISTEN_RECEIVE_TIMEOUT	**600 s**	Read timeout for inbound connections from SMPP clients.
LISTEN_TRANSMIT_TIMEOUT	**120 s**	Write timeout for inbound connections from SMPP clients.
MAKE_SOURCE_ADDRESSES_UNIQUE	**1**	Make relayed SMS source addresses unique and able to be replied to.
SERVER_HOST	no default value	Host name or IP address of the SMPP server to relay to.
SERVER_PORT	no default value	TCP port of the SMPP server to relay to.
SERVER_RECEIVE_TIMEOUT	**600 s**	Read timeout for outbound SMPP server connections.
SERVER_TRANSMIT_TIMEOUT	**120 s**	Write timeout for outbound SMPP server connections.

LISTEN_BACKLOG

(*integer*, **in [0,255]**) Connection backlog allowed by the TCP stack for inbound SMPP client connections. The default value is 255.

LISTEN_CONNECTION_MAX

(*integer, >= 0*) The maximum number of concurrent, inbound TCP connections to allow for this SMPP relay instantiation. Note that this value will be ignored if it exceeds the global *LISTEN_CONNECTION_MAX* setting.

LISTEN_INTERFACE_ADDRESS

(*string*,**"INADDR_ANY"***or dotted decimal IP address*) The IP address of the network interface to listen to for inbound SMPP client connections. May be either the string "**INADDR_ANY**" (all available interfaces) or an IP address in dotted decimal form. (For example, 193.168.100.1) The default value is "**INADDR_ANY**". Clustered HA configurations will need to set this value to correspond to the HA logical IP address.

LISTEN_PORT

(integer, TCP port number) TCP port to bind to for accepting inbound SMPP client connections. Specification of this option is mandatory; there is no default value for this option. Note also that there is no Internet Assigned Numbers Authority (IANA) assignment for this service.

LISTEN_RECEIVE_TIMEOUT

(integer, seconds > 0) Timeout to allow when waiting to read data from an SMPP client. The default value is 600 seconds (10 minutes).

LISTEN_TRANSMIT_TIMEOUT

(integer, seconds > 0) Timeout to allow when sending data to an SMPP client. The default value is 120 seconds (2 minutes).

MAKE_SOURCE_ADDRESSES_UNIQUE

(0 or 1) By default, the SMPP relay will append to each SMS source address a unique, ten digit string. The resulting SMS source address is then saved along with the other historical data. The result is a unique SMS address which may then be replied to by SMS users. The SMPP server will detect this address when used as an SMS destination address and will then send the SMS message to the correct email originator.

To disable this generating of unique SMS source addresses (for one-way SMS), specify a value of 0 (zero) for this option.

SERVER_HOST

(string, TCP hostname or dotted decimal IP address) SMPP server to relay SMPP client traffic to. Either a hostname or IP address may be specified. Specification of this option is mandatory; there is no default value for this option.

SERVER_PORT

(integer, TCP port number) TCP port for the remote SMPP server to which to relay. Specification of this option is mandatory; there is no default value for this option. There is no IANA assignment for this service; do not confuse with the IANA assignment for SNPP.

SERVER_RECEIVE_TIMEOUT

(integer, seconds > 0) Timeout to allow when waiting to read data from the SMPP server. The default value is 600 seconds (10 minutes).

SERVER_TRANSMIT_TIMEOUT

(integer, seconds > 0) Timeout to allow when sending data to the SMPP server. The default value is 120 seconds (2 minutes).

SMPP Server Options

The SMS Gateway Server can have multiple instantiations of its SMPP server, each with different characteristics chief of which will be the TCP port and interface listened on. Put differently, for each network interface and TCP port pair the SMPP server listens on, distinct characteristics may be ascribed. These characteristics are specified using the options described in this section.

Each instantiation should be placed within an option group of the form:

```
[SMPP_SERVER=server-name]
option-value-1=option-value-1
option-value-2=option-value-2
...
option-name-n=option-value-n
```

The string **server-name** merely serves to differentiate the instantiation from other instantiations.

Table 21–23 lists the SMPP server configuration options.

Table 21–23 SMPP Server Options

Options	Default	Description
LISTEN_BACKLOG	**255**	Connection backlog for inbound SMPP server connections
LISTEN_CONNECTION_MAX	no default value	Maximum number of concurrent inbound connections.
LISTEN_INTERFACE_ADDRESS	no default value	Network interface for inbound SMPP server connections.
LISTEN_PORT	no default value	TCP port for inbound SMPP server connections.
LISTEN_RECEIVE_TIMEOUT	**600 s**	Read timeout for inbound SMPP server connections.
LISTEN_TRANSMIT_TIMEOUT	**120 s**	Write timeout for inbound SMPP server connections.

LISTEN_BACKLOG

(*integer in***[0,255])** Connection backlog allowed by the TCP stack for inbound SMPP client connections. The default value is 255.

LISTEN_CONNECTION_MAX

(integer >= 0) The maximum number of concurrent, inbound TCP connections to allow for this SMPP server instantiation. Note that this value will be ignored if it exceeds the global **LISTEN_CONNECTION_MAX** setting.

LISTEN_INTERFACE_ADDRESS

*(string,***"INADDR_ANY"***or dotted decimal IP address)* The IP address of the network interface to listen to for inbound SMPP client connections on. May be either the string "**INADDR_ANY**" (all available interfaces) or an IP address in dotted decimal form. (For example, 193.168.100.1.) The default value is "**INADDR_ANY**".

LISTEN_PORT

(integer, TCP port number) TCP port to bind to for accepting inbound SMPP client connections. Specification of this option is mandatory; there is no default value for this option. Note that there is no IANA assignment for this service.

LISTEN_RECEIVE_TIMEOUT

(integer, seconds > 0) Timeout to allow when waiting to read data from an SMPP client. The default value is 600 seconds (10 minutes).

LISTEN_TRANSMIT_TIMEOUT

(integer, seconds > 0) Timeout to allow when sending data to an SMPP client. The default value is 120 seconds (2 minutes).

Gateway Profile Options

There may be zero or more gateway profiles. In the SMS Gateway Sever's configuration file, each gateway profile is declared within an option group in the following format:

```
[GATEWAY_PROFILE=profile-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

The string **profile-name** merely serves to differentiate the profile from other origination profiles.

Table 21–24 lists the SMS Gateway Server profile options.

Table 21–24 SMS Gateway Server Profile Options

Options	Default	Description
CHANNEL	**sms**	Channel to enqueue message as.
EMAIL_BODY_CHARSET	**US-ASCII**	Character set for email message bodies.
EMAIL_HEADER_CHARSET	**US-ASCII**	Character set for email message headers.
FROM_DOMAIN	no default value	Domain name for routing email back to SMS.
PARSE_RE_0, PARSE_RE_1, ..., PARSE_RE_9	no default value	Regular expressions for parsing SMS message text.
PROFILE	**GSM**	SMS profile to operate under: GSM, TDMA, or CDMA.
SELECT_RE	no default value	Regular expression for selecting the plugin.
SMSC_DEFAULT_CHARSET	**US-ASCII**	SMSC's default character set.
USE_SMS_PRIORITY	**0**	Gateway SMS priority flags to email.
USE_SMS PRIVACY	**0**	Gateway SMS privacy indicators to email.

CHANNEL

(string, 1-40 characters) Name of the MTA channel used to enqueue email messages. If not specified, then "sms" is assumed. The specified channel must be defined in the MTA's configuration.

EMAIL_BODY_CHARSET

(string, character set name) The character set to translate SMS text to prior to insertion into an email message's body. If necessary, the translated text will be MIME encoded. The default value is US-ASCII. If the SMS message contains glyphs not available in the charset, they will be converted to mnemonic characters, which may or may not be meaningful to the recipient.

A list of the character sets known to the MTA may be found in the following file:

*installation-directory***/lib/charsets.txt**

EMAIL_HEADER_CHARSET

(string, character set name) The character set to translate SMS text to prior to insertion into an RFC 822 **Subject: header** line. If necessary, the translated string will be MIME encoded. The default value is US-ASCII. If the SMS message contains glyphs not available in the charset, they will be converted to mnemonic characters, which may or may not be meaningful to the recipient

FROM_DOMAIN

(string, IP host name, 1-64 characters) Domain name to append to SMS source addresses when constructing envelope *From:* addresses for email messages. The name specified should be the correct name for routing email back to SMS. (For example, the host name associated with the MTA SMS channel.) If not specified, then the official host name of the channel specified with the **CHANNEL** option will be used.

PARSE_RE_0, PARSE_RE_1, ..., PARSE_RE_9

(string, UTF-8 regular expression) For mobile origination of email, the gateway profile needs to extract a destination email address from the text of the SMS message. This is done by means of one or more POSIX-compliant regular expressions (REs). The text of the SMS message will be evaluated by each regular expression until either a match producing a destination email address is found or the list of regular expressions exhausted.

> **Note:** Use of **PARSE_RE_*** and **ROUTE_TO** options are mutually exclusive. Use of both in the same gateway profile is a configuration error.

Each regular expression must be POSIX compliant and encoded in the UTF-8 character set. The regular expressions must output as string 0 the destination address. They may optionally output text to use in a **Subject:** header line as string 1, and text to use in the message body as string 2. Any text not "consumed" by the regular expression will also be used in the message body, following any text output as string 2.

The regular expressions will be tried in the order PARSE_RE_0, PARSE_RE_1, ..., up to PARSE_RE_9. If no regular expressions are specified, then the following default regular expression is used:

```
[ \t]*([^\( ]*)[ \t]*(?:\(([^\)]*)\))?[ \t]*(.*)
```

This default regular expression breaks into the following components:

```
[ \t]*
```

Ignore leading white space characters (*SPACE* and *TAB*).

```
([^\( ]*)
```

Destination email address. This is the first reported string.

```
[ \t]*
```

Ignore white space characters.

```
(?:\(([^\)]*)\))?
```

Optional subject text enclosed in parentheses. This is the second reported string. The leading **?:** causes the outer parentheses to not report a string. They are being used merely for grouping their contents together into a single RE for the trailing **?**. The trailing **?** causes this RE component to match only zero or one time and is equivalent to the expression:

```
{0,1}
```

[\t]*

Ignore white space characters.

(.)*

Remaining text to message body. This is the third reported string.

For example, with the above regular expression, the sample SMS message:

dan@example.com(Testing)This is a test

yields the email message:

```
To: dan@example.com
Subject: Testing
```

This is a test

As a second example, the SMS message:

sue@example.com This is another test

would yield:

To: sue@example.com

This is another test

Note that the SMS message, prior to evaluation with these regular expressions, will be translated to the UTF-16 encoding of Unicode. The translated text is then evaluated with the regular expressions which were previously converted from UTF-8 to UTF-16. The results of the evaluation are then translated to US-ASCII for the destination email address, **EMAIL_HEADER_CHARSET** for the **Subject:** text, if any, and **EMAIL_BODY_CHARSET** for the message body, if any.

PROFILE

(string, "GSM", "TDMA", or "CDMA") SMS profile to assume. Presently this information is only used to map SMS priority flags to RFC 822 *Priority:* header lines. Consequently, this option has no effect when **USE_SMS_PRIORITY=0** which is the default setting for that option.

SELECT_RE

(string, US-ASCII regular expression) A US-ASCII POSIX-compliant regular expression to compare against each SMS message's SMS destination address. If an SMS message's destination address matches this RE, then the SMS message will be sent through the gateway to email in accord with this gateway profile.

Note that since an SMS message's destination address is specified in the US-ASCII character set, this regular expression must also be expressed in US-ASCII.

SMSC_DEFAULT_CHARSET

(string, character set name) The name of the default character set used by the remote SMSC. The two common choices for this option are US-ASCII and UTF-16-BE (USC2). If not specified, US-ASCII is assumed.

USE_SMS_PRIORITY

(integer, 0 or 1) By default (with **USE_SMS_PRIORITY=0**), priority flags in SMS messages are ignored and not sent with the email messages. To have the priority flags passed with the email, specify **USE_SMS_PRIORITY=1**. When passed with the email, the mapping from SMS to email is as shown in Table 21–25:

Table 21–25 Priority Flag Mapping from SMS to Email

SMS Profile	SMS Priority Flag	Email Priority: Header Line
GSM	**0** (Non-priority)*1*, **2**, **3** (Priority)	No header line (implies **Normal**)**Urgent**.
TDMA	**0** (Bulk)**1** (Normal)**2** (Urgent)**3** (Very Urgent)	**Nonurgent No header line** (implies **Normal**) **UrgentUrgent**.
CDMA	**0** (Normal)**1** (Interactive)**2** (Urgent)**3** (Emergency)	No header line (implies **Normal**)**UrgentUrgentUrgent**.

Note that the email **Priority:** header line values are **Nonurgent**, **Normal**, and **Urgent**.

USE_SMS PRIVACY

(integer, 0 or 1) By default (with **USE_SMS_PRIVACY=0**), SMS privacy indications are ignored and not sent with the email messages. To have this information passed with the email, specify **USE_SMS_PRIVACY=1**. When passed along with email, the mapping from SMS to email is shown in Table 21–26:

Table 21–26 Privacy Flags Mapping from SMS to Email

SMS Privacy Flag	Email Sensitivity: Header Line
0 (Not restricted)	No header line.
1 (Restricted)	**Personal**.
2 (Confidential)	**Private**.
3 (Secret)	**Company-confidential**.

Note that the values of the email *Sensitivity:* header line are *Personal*, *Private*, and *Company-confidential*.

Configuration Example for Two-Way SMS

This section shows a configuration example for a two-way SMS. It provides assumptions on behavior and further assumptions and assignments. The section also describes an SMS channel configuration, SMS channel options, an SMS gateway server configuration, and additional sms_option file settings.

Assumptions on Behavior

For the sake of this example, let us assume that the following behavior is desired:

- Email messages addressed to

  sms-id@sms.domain.com

  are to be sent to the SMS address

  sms-id

  and given a unique SMS source address in the range *000nnnnnnnnn*.

- Mobile SMS messages addressed to the SMS address *000* are to be sent through the gateway to email with the email address extracted from the start of the SMS message text.

  For example, if the SMS message text is:

  jdoe@domain.com Interested in a movie?

  then the message "Interested in a movie?" is to be sent to jdoe@domain.com.

- SMS notifications sent to **000nnnnnnnnnn** are to be sent through the gateway to email and directed to the originator of the message being receipted.

In order to bring about this behavior, the following assumptions and assignments are made

Further Assumptions and Assignments

- The MTA's SMS channel uses the domain name *sms.domain.com*.
- The SMS Gateway Server runs on the host gateway.domain.com and uses:

- TCP port 503 for its SMPP relay
- TCP port 504 for its SMPP server

- The remote SMSC's SMPP server runs on the host **smpp.domain.com** and listens on TCP port 377.
- The remote SMSC's default character set is UCS2 (aka, UTF-16).

SMS Channel Configuration

To effect the above behavior, the following SMS channel configuration may be used (add these lines to the bottom of the channel blocks after running **msconfig edit channels**):

```
(blank line)
sms
sms.domain.com
```

SMS Channel Options

The channel's options would then be set as follows:

```
msconfig
msconfig> set channel:mway_sms.options.smpp_server gateway.domain.com
msconfig# set channel:mway_sms.options.smpp_port 503
msconfig# set channel:mway_sms.options.use_header_from 0
msconfig# set channel:mway_sms.options.default_source_address 000
msconfig# set channel:mway_sms.options.gateway_profile sms1
msconfig# set channel:mway_sms.options.smsc_default_charset UCS2
```

SMS Gateway Server Configuration

Finally, the Gateway Server configuration file, **sms_gateway.cnf**, should look like the following:

```
HISTORY_FILE_DIRECTORY=/sms_gateway_cache/
[SMPP_RELAY=relay1]
LISTEN_PORT=503SERVER_HOST=smpp.domain.com
SERVER_PORT=377

[SMPP_SERVER=server1]
LISTEN_PORT=504

[GATEWAY_PROFILE=sms1]
SELECT_RE=000([0-9]{10,10}){0,1}
SMSC_DEFAULT_CHARSET=UCS2
```

Testing this Configuration

If you do not have an SMSC to test on, you may want to perform some loopback tests. With some additional sms channel option settings, some simple loop back tests may be performed for the above configuration.

Additional sms_option File Settings

The additional settings for the sms channel options are:

```
msconfig
msconfig> set channel:mway_sms.options.FROM_FORMAT
msconfig# set channel:mway_sms.options.SUBJECT_FORMAT
msconfig# set channel:mway_sms.options.CONTENT_PREFIX
```

Without these settings, an email containing:

```
user@domain.com (Sample subject) Sample text
```

would get converted into the SMS message:

```
From:user@domain.com Subject:Sample Subject Msg:Sample text
```

That, in turn, would not be in the format expected by the mobile-to-email code, which wants to see:

```
user@domain.com (Sample subject) Sample text
```

Hence the need (for loopback testing) to specify empty strings for the *FROM_FORMAT*, *SUBJECT_FORMAT*, and *CONTENT_PREFIX* options.

Performing the Loopback test

Send test email messages addressed to *000@sms.domain.com*, such as:

user@domain.com (Test message) This is a test message which should loop back

The result is that this email message should be routed back to the email recipient *user@domain.com*. Be sure to have added *sms.domain.com* to your DNS or host tables for the test.

SMS Gateway Server Storage Requirements

To determine the amount of resources you will need for the SMS Gateway Server, use the numbers you generate from the requirements in Table 21–27 along with your expected number of relayed messages per second and the **RECORD_LIFETIME** setting.

Table 21–27 covers the requirements for the historical records, the SMPP relay, and SMPP server.

Table 21–27 SMS Gateway Server Storage Requirements

Component	Requirements

In-memory historical record	Each relayed message requires **33+m+s** bytes of virtual memory, where **m** is the length of the message's SMS message ID (**1 <= m <=64**) and **s** is the length of the message's SMS source address (**1 <=s <= 20**).
	When **MAKE_SOURCE_ADDRESS_UNIQUE=0**, then only **16+m** bytes are used. For 64-bit operating systems, **49+m+s** bytes of virtual memory are consumed per record [**24+m** when **MAKE_SOURCE_ADDRESS_UNIQUE=0**].
	Note also, that the heap allocator may actually allocate larger size pieces of virtual memory for each record. The maximum number of records is 43 billion:
	`(2**32-1)`
	For fewer than 16.8 million records:
	`(2**24)`
	the hash table consumes approximately 16 Mb. For fewer than 67.1 million records:
	`(2**26)`
	the hash table consumes approximately 64 Mb; for more than 67.1 million records, the hash table consumes approximately 256 Mb.
	Double the memory consumptions for 64 bit operating systems.
	These consumptions are in addition to the memory consumption required for each record itself.

On-disk historical record	Each relayed message requires on average the following number of bytes:
	`81+m+2s+3a+S+2i`
	where:
	■ **m** is the average length of SMS message IDs, and 1<=m<=64
	■ **s** is the average length of SMS source addresses, and 1 <= s <= 20
	■ **a** is the average length of email addresses, and 3 <= a <= 129
	■ **S** is the average length of **Subject:** header lines, and 0 <= S<= 80
	`78+m+3a+S+2i`

SMPP relay	Each relayed SMPP session consumes two TCP sockets: one with the local SMPP client and another with the remote SMPP server. Approximately 1 Kb of virtual memory is consumed per connection on 32 bit operating systems; 2 Kb on 64 bit operating systems.
SMPP server	Each incoming connection consumes a TCP socket. Approximately 1 Kb of virtual memory is consumed per connection on 32 bit operating systems; 2 Kb on 64 bit operating systems.

For instance, if on average 50 messages per second are expected to be relayed, SMS source addresses are 13 bytes long, SMS message IDs have a typical length of 12 bytes, email addresses 24 bytes, **Subject:** lines 40 bytes, email message and envelope IDs 40 bytes each, and historical data is retained for 7 days, then:

- There will be 30.24 million historical records to store, each on average 58 bytes in memory and 311 bytes long on disk;
- The in-memory consumption of the historical records will be about 1.70 Gb (1.63 Gb + 64 Mb); and
- The on-disk storage will be approximately 8.76 Gb.

While a sufficiency of disk may be supplied to handle any on disk requirements, the virtual memory requirement on a 32-bit machine will be a hard limit of approximately 2 Gb. To reduce the amount of virtual memory or disk storage required, use the **RECORD_LIFETIME** option to reduce the length of time records are retained.

SMS Configuration Examples

The following examples outline a couple of ways to configure one-way and two-way SMS:

- Configuring Messaging Server for One-Way SMS
- Configuring Messaging Server for One-Way SMS

22

Configuring Messaging Server for One-Way SMS

This example describes how to configure Oracle Communications Messaging Server for one-way SMS by using Unified Configuration recipes. You could also perform the same configuration manually by using the **msconfig** command, however, using recipes is faster and makes the task repeatable. You run one recipe on the system where you are adding the SMS channelsmpp. See "Using Recipes" and the discussion on recipe language in *Messaging Server Reference* and for more information on Unified Configuration recipes.

Use the "Example Recipe to Configure One-Way SMS with Messaging Server" to configure Messaging Server for one-way SMS.

1. Create the source filter file (**process_sms.filter**) with the Messaging Server unified configuration.
   a. To create a filter definition file for the source filter, enter a definition as in the following example:

```
require ["body","imap4flags"];
if body :raw :contains "Action: failed"
{
addflag "sms";
addflag "smserror";
}
```

   b. Save the filter definition file.
2. Make a copy of the example recipe file and save it as *recipe***.rcp**.
3. To run the recipe, type the following command:

```
cd MessagingServer_home/bin
msconfig run recipe_name
```

4. If running a compiled configuration, recompile the MTA configuration to capture the changes you have made:

```
cd MessagingServer_home/bin
imsimta cnbuild
```

5. Restart Messaging Server.

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

Example Recipe to Configure One-Way SMS with Messaging Server

```
set_option("mta.enable_sieve_body", "1");

if exists_channel("sms") {
delete_channel("sms");
}
if exists_channel("process_sms") {
delete_channel("process_sms");
}

add_channel("sms", ["notificationchannel", "process_sms", "backoff", "PT10M PT20M
PT30M", "notices", "1", "official_host_name", "sms-handle", "options.smpp_server",
"127.0.0.1", "options.smpp_port", "8500", "options.default_source_address", "000",
"options.smsc_default_charset", "UTF-16-BE", "options.esme_password", "password",
"options.esme_system_id", "smppclient"]);

add_channel("process_sms", ["sourcefilter",
"file:/opt/sun/comms/messaging64/config/process_sms.filter", "official_host_name",
"process_sms-daemon"]);

#Setting rewrite rules
append_rewrites(["sms","$U@sms-handle"]);
```

23

Configuring Messaging Server for Two-Way SMS

This example describes how to configure Oracle Communications Messaging Server for two-way SMS by using Unified Configuration recipes. You could also perform the same configuration manually by using the **msconfig** command, however, using recipes is faster and makes the task repeatable. You run one recipe on the system where you are adding the SMS channelsmpp. See "Using Recipes" and the discussion on recipe language in *Messaging Server Reference* and for more information on Unified Configuration recipes.

Use the following "Example Recipe to Configure Two-Way SMS with Messaging Server" to configure Messaging Server for two-way SMS.

1. Create the source filter file (**process_sms.filter**) with the Messaging Server Unified Configuration.

   a. To create a filter definition file for the source filter, enter a definition as in the following example. This is the source filter for the **process_sms** channel:

   ```
   require ["body","imap4flags"];
   if body :raw :contains "Action: failed"
   {
   addflag "sms";
   addflag "smserror";
   }
   ```

   b. Save the process definition file.

2. Make a copy of the example recipe file and save it as *recipe***.rcp**.

3. To run the recipe, type the following command:

   ```
   cd MessagingServer_home/bin
   msconfig run recipe_name
   ```

4. If running a compiled configuration, recompile the Messaging Server configuration to capture the changes you have made:

   ```
   imsimta cnbuild
   ```

5. Restart Messaging Server.

   ```
   cd MessagingServer_home/bin
   stop-msg
   start-msg
   ```

Example Recipe to Configure Two-Way SMS with Messaging Server

```
set_option("mta.enable_sieve_body", "1");
set_option("sms_gateway.enable", "1");

if exists_channel("sms") {
delete_channel("sms");
}
if exists_channel("process_sms") {
delete_channel("process_sms");
}

add_channel("sms", ["notificationchannel", "process_sms", "backoff", "PT10M
PT20M PT30M", "notices", "1", "sourcefilter",
"file://opt/sun/comms/messaging64/config/sms_channel.filter", "official_host_
name", "sms-handle", "options.smpp_server", "127.0.0.1", "options.smpp_port",
"8500", "options.gateway_profile", "GATEWAY", "options.default_source_
address", "000", "options.smsc_default_charset", "UTF-16-BE", "options.use_
header_from", "0", "options.esme_password", "password", "options.esme_system_
id", "smppclient"]);

add_channel("process_sms", ["sourcefilter",
"file://opt/sun/comms/messaging64/config/process_sms.filter", "official_host_
name", "process_sms-daemon"]);

Setting rewrite rules
append_rewrites(["sms","$U@sms-handle"]);

if exists_group("sms_gateway.smpp_server:SMPPSERVER") {
delete_group("sms_gateway.smpp_server:SMPPSERVER");
}

add_group("sms_gateway.smpp_server:SMPPSERVER", ["tcp_ports", "4080", "server_
host", "127.0.0.1", "server_port","950", "ESME_PASSWORD","password","ESME_
SYSTEM_ID","smppclient"]);

if exists_group("sms_gateway.smpp_relay:SMPPRELAY") {
delete_group("sms_gateway.smpp_relay:SMPPRELAY");
delete_group("sms_gateway.smpp_relay:SMPP_RELAY");
}

add_group("sms_gateway.smpp_relay:SMPPRELAY", ["server_port", "950", "tcp_
ports", "8500", "server_host", "127.0.0.1"]);

if exists_group("sms_gateway.gateway_profile:GATEWAY") {
delete_group("sms_gateway.gateway_profile:GATEWAY");
}

add_group("sms_gateway.gateway_profile:GATEWAY", ["select_re", "([0-9]*)",
"mta_channel", "sms", "smsc_default_charset", "UTF-16-BE", "email_body_
charset", "UTF-8", "email_header_charset", "UTF-8", "text_to_subject", "1"]);
```

6. Set the JMQ options to create non-delivery failure notifications of any SMS messages:
   - NewMsg.enable -v 1
   - jmqHost -v "127.0.0.1"
   - jmqPort -v "7777"
   - jmqUser -v "user
   - jmqpwd -v "xxx"

- DestinationType -v "queue"
- jmqQueue -v "ucsms1"
- Priority -v 3
- ttl -v 1000
- Persistent -v 1
- maxheadersize -v 1024
- noneinbox.enable -v 1
- msgflags.enable -v 0
- readmsg.enable -v 0

24

Using the iSchedule Channel to Handle iMIP Messages

This chapter describes how to configure Oracle Communications Messaging Server to use the iSchedule protocol to post a calendar event received in an iMIP (iCalendar Message-Based Interoperability Protocol) message to Oracle Communications Calendar Server. This capability enables "internal" users to automatically process calendar invitations from "external" users. To enable this interoperability between calendaring systems, you configure a Messaging Server "iSchedule" channel to process the iMIP messages. For additional information, see the discussion on enabling the iSchedule channel to handle iMIP messages in *Calendar Server System Administrators Guide*.

Inviting Users on Internal and External Calendar Systems Background

Calendar Server meetings often have multiple invitees. These invitees can be both *internal* users, who reside on the same Calendar Server deployment, or *external* users, who reside either on a different Calendar Server deployment administered by a separate group, or on an outside calendaring system, such as Exchange, Google Calendar, and so on. For "internal" invitees, Calendar Server automatically adds the meeting request to their calendars (referred to as *implicit scheduling*) and also sends them email notification about the meeting request. All "external" invitees are sent an iMIP (iCalendar Message-Based Interoperability Protocol) email with the meeting request as an attachment. External invitees must manually process these messages to add the invite to their calendars.

Manually Accepting External Invitations

Meeting invitations from external organizers are sent to the user's mailbox. Mail clients, such as Outlook or Thunderbird, enable users to process these invitations and add the invitation to their calendar. How the invitation is added to the user's calendar depends on the specific mail client, but the invitation is not added until the user has manually read the email and accepted the meeting request.

Automatically Accepting External Invitations

You can configure your Calendar Server deployment to automatically process invitations coming from external calendar systems. To users, handling an external invite then appears just like an internal invite.

This capability involves an intermediary in the form of Messaging Server. You configure the Messaging Server MTA to process the calendar invite email (which is an iMIP message), extract the pertinent calendar information, then use the iSchedule

protocol to add the invite to the attendee's calendar database. As a consequence, external event invitations automatically appear in the user's calendar without the need for a manual intervention, even when using a "non-calendar" aware client.

Once you have configured your deployment accordingly, users have a choice on how to process invitations. Users can either accept the "external" meeting invite directly from their calendar client (either desktop or mobile iOS CalDAV clients) or they can still accept it from their email client. That is, CalDAV clients now receive iMIP messages in their scheduling-inbox, and are able to process them just like regular CalDAV-based invitations and replies. Because the invitation is already in the user's calendar, invitation replies and cancel are also merged automatically. Thus, based upon the user accepting or rejecting the request, the calendar client merely has to update the attendee status in the invitation. That status change also enables Calendar Server to send a response to the organizer indicating the disposition of the meeting request. As the response is sent directly by Calendar Server, it does not matter how the user accepted the invitation (whether from a calendar client on a mobile device, desktop, or from an email client). Finally, because of the addition of meta data to the email message, the Convergence (web-based) client is able to display a scheduling-specific form to users that enables them to accept, decline, or indicate a "maybe" to meeting invitations directly from their email without having to switch to the calendar client.

Message Server iMIP Configuration Overview

A Messaging Server MTA channel (an "iSchedule" channel) handles automatic processing of external calendar invites by:

1. Intercepting incoming emails containing an iMIP message An iMIP message has an iCalendar attachment of type 'text/calendar' with a **method=<action>** option in the 'Content-Type:' header.
2. Injecting the corresponding iTIP message into the regular calendar server workflow
3. Adding meta-data (email X- headers) to the iMIP email before delivering it to its recipients
4. Posting the invitation request to a calendar iSchedule URL

The Calendar Server then consumes the invitation from the iSchedule URL just as it would have done if an external calendar server had posted an invitation to one of its users. On the Calendar Server side, an iSchedule database, which is a separate table from the Calendar Server database, acts as the global inbox and outbox for external invites.

Administering the iMIP configuration involves:

- Enabling or disabling iMIP messaging processing
- Configuring the iSchedule service URL
- Configuring the criteria for messages to be selected for processing

You should configure the iSchedule channel on the message store systems if you are not using LMTP. If you are using LMTP, configure the iSchedule on the MTAs.

For more information on the iCalendar Message-Based Interoperability Protocol, see RFC 6047 (http://www.faqs.org/rfcs/rfc6047.html). For more information on iCalendar Transport-independent Interoperability Protocol (iTIP), see RFC 5546 (http://www.faqs.org/rfcs/rfc5546.html).

Configuring the iSchedule Channel for iMIP Messages in Unified Configuration

You can use a Messaging Server Unified Configuration recipe to automate the configuration process or you can manually perform the necessary configuration. After completing the configuration, you also need to verify the Calendar Server configuration. See "Verifying the Calendar Server Configuration" for more information.

You do not need to perform any additional Convergence configuration for Convergence to automatically process invitations coming from external calendar systems. If you have configured Messaging Server and Calendar Server correctly, Convergence users see a UI form that they use to reply to external invites.

Using the iSchedule Recipe to Automate Configuring the iSchedule Channel in Unified Configuration

Unified Configuration provides a recipe language and some stock recipes to automate certain configuration tasks (see "Using Recipes"). To set up the iSchedule channel, you can use a recipe called **iSchedule.rcp**, which automatically sets up the channel definition, job controller configuration, channel options, Sieve rule, and CONVERSION mapping.

To use the **iSchedule.rcp** recipe:

1. Run the **msconfig** command with the recipe name.

   ```
   MessagingServer_home/bin/msconfig run iSchedule.rcp
   ```

2. Respond to the prompts, for example:

   ```
   HTTP URL for iSchedule server: http://host1.example.com:8080/dav/ischedule/
   Destination channel for messages to check (<RET> if no more): ims-ms
   ```

   Use the iSchedule URL and destination channels based on your deployment.

   Note: Be sure to add the trailing forward slash (/) in the iSchedule URL, otherwise you will receive the error message "HTTP Error 401 Unauthorized."

3. If you are using a compiled configuration, recompile the configuration.

   ```
   MessagingServer_home/bin/imsimta cnbuild
   ```

4. Restart Messaging Server.

   ```
   MessagingServer_home/bin/stop-msg
   MessagingServer_home/bin/start-msg
   ```

5. Verify the Calendar Server configuration. See "Verifying the Calendar Server Configuration" for more information.

Manually Configuring the iSchedule Channel in Unified Configuration

The high-level steps to manually configure the iSchedule channel by using the **msconfig** command involve:

- Adding the channel

- Configuring the conversion mapping
- Specifying messages to be processed by iSchedule

To manually configure the iSchedule Channel, job controller master command, **include_conversiontag** MTA option, and conversion mapping:

1. Use the **msconfig** command in interactive mode to configure the iSchedule channel, the job controller master command for the channel, the **include_conversiontag** MTA option if you want to have a **TAG=** clause included in your conversion mapping probes, and the conversion mapping.

   ```
   MessagingServer_home/bin/msconfig
   msconfig> set channel:ischedule.official_host_name ischedule-daemon
   msconfig# set channel:ischedule.official_host_name.single
   msconfig# set channel:ischedule.options.handle-imip 1
   msconfig# set channel:ischedule.options.ischedule-url
   http://host:port/dav/ischedule/
   msconfig# set instance.job_controller.channel_class:ischedule.master_command
   IMTA_BIN:ischedule
   msconfig# set mapping:conversion.rule
   "IN-CHAN=ischedule;OUT-CHAN=*;TAG=*;CONVERT" NO
   msconfig# set mapping:conversion.rule
   "IN-CHAN=*;OUT-CHAN=*;TAG=ISCHEDULE;CONVERT" "YES,CHANNEL=ischedule"
   msconfig# set include_conversiontag 2
   msconfig# write
   ```

   Use the host name and alternately the port for the Calendar Server configured for the iSchedule database.

   Note: Be sure to add the trailing forward slash (/) in the **ischedule-url**, otherwise you will receive the error message "HTTP Error 401 Unauthorized."

2. Edit the filters block to specify messages to be processed by iSchedule.

   ```
   msconfig> edit filter
   ```

   The filter block appears in the editor that is the default configured for your login.

3. Add the following lines to create a filter that selects all the messages that have "text/calendar" MIME as an attachment:

   ```
   require ["mime", "environment"];
   if allof(environment :is "vnd.sun.destination-channel" ["ims-ms"],
   header :mime :anychild :contenttype :is "content-type" "text/calendar",
   NOT header :contains "X-Oracle-CS-iSchedule-Ignore" "Yes") {
   addconversiontag "ISCHEDULE";
   }
   ```

   iMIP messages generated by Calendar Server contain a "X-Oracle-CS-iSchedule-Ignore: Yes" header to indicate that the event was already added to the user's calendar. So, the Sieve rule should ignore those iMIP messages by not tagging them with an ISCHEDULE conversion tag. Failing to do so results in an iSchedule post of the event that is already present in the user's calendar.

4. Write the configuration and exit the **msconfig** interactive mode.

   ```
   msconfig# write
   msconfig> exit
   #
   ```

5. If you are using a compiled configuration, recompile the configuration.

   ```
   MessagingServer_home/bin/imsimta cnbuild
   ```

6. Restart Messaging Server.

   ```
   MessagingServer_home/bin/stop-msg
   MessagingServer_home/bin/start-msg
   ```

7. Verify the Calendar Server configuration. See "Verifying the Calendar Server Configuration" for more information.

Verifying the Calendar Server Configuration

To ensure that your Calendar Server configuration is setup properly, use the following steps to verify SMTP settings and iMIP email notifications. iMIP email notifications need to also be configured for internal users, that is, users on the same Calendar Server host. If necessary, restart the GlassFish Server container on which Calendar Server is deployed.

1. Check the SMTP configuration for the following settings.

   ```
   cd CalendarServer_home/bin
   davadmin config list|grep smtp
   notification.dav.smtphost=host2.example.com
   notification.dav.smtpuser=user
   notification.dav.smtppassword=********
   notification.dav.smtpport=25
   notification.dav.smtpstarttls=true
   notification.dav.smtpusessl=false
   notification.dav.smtpdebug=false
   notification.dav.smtpauth=false
   ```

2. Check the email notifications configuration.

   ```
   davadmin config modify -o notification.dav.smtpstarttls -v false
   Enter Admin password:
   davadmin config list|grep imip
   notification.dav.enableimipemailnotif=false
   davadmin config modify -o notification.dav.enableimipemailnotif -v true
   Enter Admin password:
   ```

3. Check the whitelist configuration for the iSchedule port. The **service.dav.ischedulewhitelist** configuration option prevents denial of service attacks on the iSchedule port. See the discussion on enabling the iSchedule channel to handle iMIP messages in *Calendar Server System Administrators Guide* for more information.

4. If necessary, restart GlassFish Server. For example:

   ```
   /opt/SUNWappserver/bin/asadmin stop-domain domain1
   /opt/SUNWappserver/bin/asadmin start-domain domain1
   ```

Modifying iSchedule Channel Options

After you have configured the iSchedule channel, you might need to change iSchedule channel options as described in this section.

To Enable or Disable iMIP Message Processing

Use the **msconfig** command to enable or disable iMIP message processing by setting the **handle-imip** option to 1 or 0 respectively. For example, the following command disables iMIP message process:

```
MessagingServer_home/bin/msconfig
msconfig> set instance.channel:ischedule.options.handle-imip 0
msconfig# write
msconfig> exit
```

To Modify the iSchedule Service URL

Use the **msconfig** command to modify the iSchedule URL by editing the **ischedule-url** option. For example:

```
MessagingServer_home/bin/msconfig
msconfig> set instance.channel:ischedule.options.ischedule-url
http://host:port/dav/ischedule/
msconfig# write
msconfig> exit
```

Configuring the iSchedule Channel in Legacy Configuration

This section describes how to configure the iSchedule channel for Messaging Server in legacy configuration (that is, Messaging Server 7 Update 5 and greater but you either did not convert an existing deployment to Unified Configuration, or choose to install a fresh deployment using Unified Configuration.)

1. Create the iSchedule channel.
   a. Add following lines to the *MessagingServer_home*/**config/imta.cnf** file:

      ```
      ischedule single
      ischedule-daemon
      ```

   b. Add the following lines to the *MessagingServer_home*/**config/job_controller.cnf** file:

      ```
      [CHANNEL=ischedule]
      master_command=IMTA_BIN:ischedule
      ```

2. Configure CONVERSION mapping by adding the following lines to the *MessagingServer_home*/**config/mappings** file:

   ```
   CONVERSION

    IN-CHAN=ischedule;OUT-CHAN=*;TAG=*;CONVERT NO
    IN-CHAN=*;OUT-CHAN=*;TAG=ISCHEDULE;CONVERT YES,CHANNEL=ischedule
   ```

3. Enable or disable iMIP message processing. To enable or disable iMIP message processing, create a channel options file, *MessagingServer_home*/**config/ischedule_option**, and add the following line:

   ```
   handle-imip=1 (to enable)
   handle-imip=0 (to disable)
   ```

4. Configure the iSchedule service URL. In the channel options file, specify the iSchedule Service URL as follows:

   ```
   ischedule-url=http://host:port/dav/ischedule/
   ```

5. Configure the **include_conversiontag** MTA option if you want to have a {TAG=}} clause included in your conversion mapping probes by adding the following line to the *MessagingServer_home*/**config/option.dat** file:

   ```
   INCLUDE_CONVERSIONTAG=2
   ```

6. Specify messages to be processed by iSchedule. Run the following Sieve script to select all the messages that have text/calendar MIME as an attachment. This script should be placed in the location of your system-wide scripts.

   ```
   require ["mime", "environment"];
   if allof(environment :is "vnd.sun.destination-channel" ["ims-ms"],
   header :mime :anychild :contenttype :is "content-type" "text/calendar",
   NOT header :contains "X-Oracle-CS-iSchedule-Ignore" "Yes") {
   addconversiontag "ISCHEDULE";
   }
   ```

7. If you are using a compiled configuration, recompile the configuration.

   ```
   MessagingServer_home/imsimta cnbuild
   ```

8. Restart Messaging Server.

   ```
   MessagingServer_home/bin/stop-msg
   MessagingServer_home/bin/start-msg
   ```

9. Verify the Calendar Server configuration. See "Verifying the Calendar Server Configuration" for more information.

Troubleshooting the iSchedule Configuration

Use the following information to troubleshoot your iSchedule configuration:

- All messages processed through the iSchedule channel have a "Received:" header containing **ischedule-daemon.***host***.***domain*. This is true whether handling iMIP is enabled or not. If iMIP handing is enabled, the iMIP messages have an extra header, "X-Oracle-CS-iSchedule-Status:," which contains the HTTP status code sent by the iSchedule service in response to the posted iSchedule message.
- The iSchedule channel log files are located in *DataRoot***/log/ischedule_master.log.***
- Use the *MessagingServer_home***/bin/imsimta qm counters** command to list the number of messages processed by the iSchedule channel.
- One common misconfiguration is to specify a wrong destination channel in the Sieve rule. If you do not see the "X-Oracle-CS-iSchedule-Status:" header in iMIP e-mails, or do not see the iSchedule counters increase when you use the **imsimta qm counters** command, check if the destination channel that you specified in the Sieve rule matches the destination channel that you have configured for that user.

25

Handling sendmail Clients

This chapter describes how to configure Oracle Communications Messaging Server to work with **sendmail** clients. If users (or system utilities, for example, **cron**) send messages through **sendmail** clients, you can configure Messaging Server to work with those clients over protocol. Users can continue to use the UNIX **sendmail** client.

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

Each time a new **sendmail** patch is applied to your system, you must modify the **submit.cf** file. See "To Create the sendmail Configuration File on Oracle Solaris 9 Platforms" for more information.

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

To Create the sendmail Configuration File on Oracle Solaris 8 Platforms

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

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

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

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

   rhino.west.example.com is the localhost name and **west.example.com** is the default email domain as described in creating the initial Messaging Server runtime configuration in *Messaging Server Installation and Configuration Guide*. In an HA environment, use the logical host name. See *Messaging Server Installation and Configuration Guide* for information about logical hostnames for high availability.

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

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

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

4. Make a backup copy of the existing **sendmail.cf** file located in the **/etc/mail** directory.

   a. Copy and rename **/usr/lib/mail/cf/sunone-msg.cf** to **sendmail.cf** file.

   b. Move the new **sendmail.cf** file to the **/etc/mail** directory.

To Create the sendmail Configuration File on Oracle Solaris 9 Platforms

1. Find the file **submit.mc** file in directory **/usr/lib/mail/cf** and create a copy of this file.

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

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

   ```
   FEATURE(`msp')dn
   ```

   to

   ```
   FEATURE(`msp', `rhino.west.example.com')dnl
   ```

   rhino.west.example.com is the localhost name and **west.example.com** is the default email domain as described in creating the initial Messaging Server runtime configuration in *Messaging Server Installation and Configuration Guide*. In an HA environment, use the logical host name. See *Messaging Server Installation and Configuration Guide* for information about logical hostnames for high availability.

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

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

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

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

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

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

26

Handling Forged Email by Using the Sender Policy Framework

This chapter describes how to use Sender Policy Framework (SPF) with Oracle Communications Messaging Server to reduce instances of forged email.

About Sender Policy Framework

Spam producers and email scammers often forge email by using false domain names and email addresses, or by using legitimate domain names and email addresses to fool users into thinking that a message is from someone or some company they know. For example, a spammer could send email from an address such as **president@whitehouse.gov** and the user could be fooled into thinking the mail was actually from this address. Forging email might fool users into opening the unsolicited message, or worse, provide information to a false authority. Also, spammers prefer to send their email from legitimate domains that are not on an RBL list.

Sender Policy Framework (SPF) is a technology that can detect and reject forged email during the SMTP dialogue. Specifically, SPF is a protocol that allows a domain to explicitly authorize the hosts that may use its domain name. In addition, a receiving host may be configured to check this authorization. SPF can thus significantly reduce the instances of forged email.

Note: When using Unified Configuration, you use the **msconfig** command to configure options instead of editing the legacy configuration files.

SPF Theory of Operations

When a message comes into Messaging Server, the MTA does an SPF query to determine if the address actually came from the domain on the address. An SPF query consults the DNS for TXT records belonging to the domain of the message (domain). Domain is either the domain name specified as the argument for HELO or EHLO (if the **spfhelo** channel option is used) or the domain name in the originator's address given in the MAIL FROM: command (typically the part after the @ character). If no domain name is specified or available, the one specified during HELO/EHLO is used as domain. Most ISPs distribute an authorized list of IP addresses that match their domains. If the IP address does not match the domain name, then the message is assumed to be forged.

Note: Prior to querying the DNS, the software checks the SPF_LOCAL mapping table for a match for domain. If a match is found there, it will be used first.

If a record found from the mapping table contains a **redirect=** domain clause, then the redirection to domain will be done as a DNS query, skipping the recursive and redundant mapping file check.

An example of a resulting TXT record is:

```
v=spf1 +mx a:colo.example.com/28 -all
```

The **v=spf1** token is required for SPF records supported by this RFC.

+mx checks MX records for domain and confirms that the source IP address for this SMTP connection matches one of the IP addresses given as a result of an MX query for domain. If there is a match, the + means that the result of this is **Pass**.

a:colo.example.com/28 checks for A records for **colo.example.com**, then confirm that the source IP address for this SMTP connection is in the same specified CIDR subnet as the A records, comparing only 28 bits (masked against 255.255.255.240). No qualifier character was specified, so it defaults to + meaning that a match results in a **Pass**.

Finally, **-all** matches everything else and results in **Fail**. For a complete description of SPF records, refer to RFC 4408 at `http://www.ietf.org/rfc/rfc4408.txt`.

SPF processing can have one of several results. Table 26–1 shows the results and their descriptions.

Table 26–1 SPF Processing Results

Result	Description
Pass	The lookup passed, meaning that an SPF record was found and the record validated the originating system as being authorized to use domain.
Fail	The lookup found a matching SPF record, however, the record explicitly denied authorization for the SMTP client to use domain during the SMTP transaction. The default behavior of Messaging Server's SPF implementation is to reject the SMTP command with a 5xx reply.
SoftFail	The lookup found a matching SPF record, and the record also denies authorization for the SMTP client to use domain. However, the denial is less strict and the record does not direct an outright failure. The default behavior of Messaging Server's implementation is to accept the message, but note the **SoftFail** in the Received-SPF: header for subsequent evaluation such as during Sieve processing.
Neutral	The SPF record makes no claim to the SMTP client's authorization to use domain. The message will be accepted. The specification requires that **Neutral** be treated the same as **None**.
None	No matching SPF record was found, therefore no SPF processing was done.
PermError	A permanent error was encountered during SPF processing, such as syntax errors in the SPF record, DNS failures while processing nested SPF records (due to **include:** mechanism or a **redirect=** modifier), or exceeding configured limits for SPF processing while processing nested SPF records. The default behavior is to reject the SMTP command with a 5xx reply.

Table 26–1 (Cont.) SPF Processing Results

Result	Description
TempError	A temporary error was encountered during SPF processing, most likely due to DNS timeouts querying SPF records. The default behavior is to reject the SMTP command with a 4xx reply.

After SPF processing has completed, a **Received-SPF:** header is written to the message documenting the result of the SPF processing. This header can then be queried during Sieve processing for subsequent consideration. Extensive debugging is available if the MTA option **MM_DEBUG** is enabled (>0). In Unified Configuration, run the **msconfig** command to set the option. In legacy configuration, set the option in the **option.dat** file.

SPF Limitations

SPF is only one tool to use to fight spam, and it does not address all issues. A spammer can easily create a domain and add an SPF TXT record that makes the domain seem legitimate. On the other hand, SPF is very effective for detecting forged email from established ISPs, although many TXT records allow the SPF to not fail.

SPF Pre-Deployment Considerations

It is important to have a very fast DNS server on your system because a DNS query for every message is required.

Setting up the Technology

The two steps to set up SPF technology are:

- Place channel options on the incoming TCP channel (typically the **tcp_local** channel, although there might be other channels if you allow channel switching from **tcp_local** to another channel). See Table 26–2, " SPF Options" for more information.
- Set up the options. In Unified Configuration, run the **msconfig** command. In legacy configuration, edit the **option.dat** file. See Table 26–3, " SPF Limiting Options" for more information.

Reference Information

This section provides reference information for the SPF channel options and the SPF MTA options. SPF support is implemented through four channel options applied to the incoming **tcp_*** channel (typically **tcp_local**). Table 26–2 shows the options and their descriptions. In Unified Configuration, use the **msconfig edit channels** command to edit SPF channel options. In legacy configuration, edit the **imta.cnf** file.

Table 26–2 SPF Options

Option	Description
spfnone	Disables SPF processing
spfhelo	Enables SPF processing for the domain name specified as an argument to HELO or EHLO.

Table 26–2 (Cont.) SPF Options

Option	Description
spfmailfrom	Enables SPF processing for the domain name provided for the originator envelope address after receiving the MAIL FROM:.
spfrcptto	Enables SPF process for the domain name provided for the originator envelope address after receiving the RCPT TO:. Processing is the same as **spfmailfrom** except that it is delayed in the SMTP transaction until after the RCPT TO: command has been issued and the recipient has otherwise been confirmed to be a valid recipient.

Note: **spfmailfrom** and **spfrcptto** are conflicting options and you should only specify one of these two options on the channel. You can, however, use **spfhelo** in conjunction with either **spfmailfrom** or **spfrcptto** to perform both kinds of SPF checks.

Additional support to establish limits on SPF processing and to control whether SMTP commands will be accepted, failed with a 4xx response (temporary failure), or failed with a 5xx response (permanent failure) for the various SPF results includes: **Fail**, **SoftFail**, **PermError**, and **TempError**.

Table 26–3 shows the MTA options that place limits on SPF processing. In Unified Configuration, set options with the **msconfig** command. In legacy configuration, edit the **option.dat** file.

Table 26–3 SPF Limiting Options

Option	Description
SPF_MAX_RECURSION	Specifies the number of recursions that will be allowed into nested SPF records due to **include:** or **redirect=**. Exceeding this limit will result in a **PermError**. Default: 10 (mandated by the RFC)
SPF_MAX_DNS_ QUERIES	Specifies the number of mechanisms or modifiers that require DNS lookups (including **include:**, **a:**, **mx:**, **ptr:**, **exists:**, **redirect=**, and **exp=**). The limit is not counted as the number of actual DNS lookups, so one mechanism could lead to several DNS queries. Exceeding this limit will result in a **PermError**. Default: 10 (mandated by the RFC)
SPF_MAX_TIME	Specifies the number of seconds that will be allowed for the SPF processing to complete. Exceeding this value will result in a **TempError**. The default value is more generous than the RFC suggests. Default: 45

Additionally, the following MTA options can be configured to control the behavior of the SMTP server in response to SPF results of **Fail**, **SoftFail**, **PermError**, and **TempError**. In Unified Configuration, run the **msconfig** command. In legacy configuration, edit the **option.dat** file. For each of these results, the SMTP server can send back a 2xx (success) response, 4xx (temporary failure), or 5xx (permanent failure). Also, for **Fail** and **SoftFail**, the MTA can distinguish between an SPF result as the result of an "all" mechanism versus an otherwise explicitly referenced match. You can then make a distinction between a particular result and the SPF record's default result. The valid values for any of these options is 2, 4, or 5. The values of 2, 4, or 5 correspond to 2xx, 4xx, or 5xx responses from the SMTP server as a result of getting that particular SPF status. So, for example, if SPF_SMTP_STATUS_FAIL=2 and the SPF

record explicitly blocks us with a "-a:192.168.1.44" (our IP address), then instead of responding with a 5xx response, we'll accept the address with a "250 OK" instead.

Table 26–4 shows MTAP options that control SPF failures and error options.

Table 26–4 SPF Failure and Error Options

Option	Description
SPF_SMTP_STATUS_ FAIL	Used when the match of an SPF record is a "-" flagged mechanism other than "-all." Default: 5
SPF_SMTP_STATUS_ FAIL_ALL	Used when the matching mechanism is "-all." Default: 5
SPF_SMTP_STATUS_ SOFTFAIL	Used when the match of an SPF record is a "~" flagged mechanism other than "~all." Default: 2
SPF_SMTP_STATUS_ SOFTFAIL_ALL	Used when the matching mechanism is "~all." Default: 2
SPF_SMTP_STATUS_ TEMPERROR	Used when there is a temporary failure, usually related to DNS processing problems. Default: 4
SPF_SMTP_STATUS_ PERMERROR	Used when there is a permanent failure, usually due to syntax or other technical errors found during SPF processing. (This is due to a non-local error.) Default: 5

Testing SPF by Using spfquery

You can use the **spfquery** testing utility to test SPF processing.

> **Note:** **spfquery** does not test your SPF configuration. It tests what would be returned if you were to enable SPF processing.

Requirements: Must be run as a user who has access to run the Messaging Server binaries and access its libraries such as **root** or **mailsrv**, for example.

Location: *MessagingServer_home***/bin/**

Syntax

```
spfquery [-i ip-address] [-s sender-email] [-h helo-domain] [-e none | neutral |
pass | fail | temperror | permerror] [-v] [-V] [?] domain
```

Table 26–5 shows the **spfquery** options and their descriptions.

Table 26–5 spfquery Options

Option	Description
-i *ip address*	Specifies the IP address to be used as the remote address for the SPF query. Default is **127.0.0.1**. This option can also be --**ip-address**.

Table 26–5 (Cont.) spfquery Options

Option	Description
-s *domain*	The email address that will be used as if it were specified as **MAIL FROM:.** Default: **postmaster@domain**. This option can also be **--sender**.
-h *helo-domain*	The domain name as if it were specified for the HELO domain. This domain is not verified itself, but instead provided as supplemental information for macro processing. Default value is the same as the value you specified for domain. This option can also be **--helo-domain**.
-e *result*	**spfquery** compares the result of the SPF processing with what is expected and if the result is different, a message is printed and **spfquery** exits with a non-zero return status. Result can be one of: **none**, **neutral**, **pass**, **fail**, **softfail**, **temperror**, or **permerror**. This option can also be **--expect**.
-v	Enables verbose output during SPF processing. This option can also be **--verbose**.
-V	Prints the current version of the SPF library. This option can also be **--version**.
-?	Prints this usage information. This option can also be **--help**.

Example with Debugging Enabled

```
# /opt/sun/comms/messaging64/bin/spfquery -v -i 192.168.1.3
11.spf1-test.example.com
    Running SPF query with:
      IP address: 192.168.1.3
          Domain: 11.spf1-test.example.com
          Sender: postmaster@11.spf1-test.example.com (local-part: postmaster)
     HELO Domain: 11.spf1-test.example.com

    15:30:04.33: ----------------------------------------------------------------
    15:30:04.33: SPFcheck_host called:
    15:30:04.33:       source ip = 192.168.1.3
    15:30:04.33:          domain = 11.spf1-test.example.com
    15:30:04.33:          sender = postmaster@11.spf1-test.example.com
    15:30:04.33:      local_part = postmaster
    15:30:04.33:     helo_domain = 11.spf1-test.example.com
    15:30:04.33:
    15:30:04.33:   Looking up "v=spf1" records for 11.spf1-test.example.com
    15:30:04.35:     DNS query status: Pass
    15:30:04.35:       "v=spf1 mx:spf1-test.example.com                  -all"
    15:30:04.35:
    15:30:04.35:   Parsing mechanism: " mx : spf1-test.example.com"
    15:30:04.35:     Assuming a Pass prefix
    15:30:04.35:     Processing macros in spf1-test.example.com
    15:30:04.35:     Comparing against 192.168.1.3
    15:30:04.35:     Looking for MX records for spf1-test.example.com
    15:30:04.41:       mx02.spf1-test.example.com:
    15:30:04.41:         192.0.2.22 - No match
    15:30:04.41:         192.0.2.21 - No match
    15:30:04.41:         192.0.2.20 - No match
    15:30:04.41:         192.0.2.23 - No match
    15:30:04.41:       mx01.spf1-test.example.com:
    15:30:04.42:         192.0.2.13 - No match
    15:30:04.42:         192.0.2.11 - No match
    15:30:04.42:         192.0.2.12 - No match
```

```
15:30:04.42:         192.0.2.10 - No match
15:30:04.42:       mx03.spf1-test.example.com:
15:30:04.42:         192.0.2.32 - No match
15:30:04.42:         192.0.2.30 - No match
15:30:04.42:         192.0.2.31 - No match
15:30:04.42:         192.168.1.3 - Matched
15:30:04.42:   Mechanism matched; returning Pass
15:30:04.42:
15:30:04.42:   Parsing mechanism: "- all : " (not evaluated)
15:30:04.42:
15:30:04.42: SPFcheck_host is returning Pass
15:30:04.42: ----------------------------------------------------------------
```

Handling Forwarded Mail in SPF by Using the Sender Rewriting Scheme (SRS)

As described above, SPF is a mechanism that attempts to prevent email forgery by looking up special **TXT** records associated with the domain in the mail **FROM:** (envelope from) address. This operation, which can actually involve several DNS lookups, eventually produces a list of IP addresses that are authorized to send mail from the domain. The IP address of the SMTP client is checked against this list and if it isn't found, the message can be considered to be fraudulent.

SPF presents serious problems for sites that provide mail forwarding services such as universities (for their alumni) or professional organizations (for their members). A forwarder ends up sending out mail from essentially arbitrary senders, which can include senders who have implemented SPF policies and which, of course, do not list the IP addresses of the forwarding system or systems as being permitted to use addresses from their domain.

The Sender Rewriting Scheme, or SRS, provides a solution to this problem. SRS works by encapsulating the original sender's address inside a new address using the forwarder's own domain. Only the forwarder's own domain is exposed for purposes of SPF checks. When the address is used it routes the mail (usually a notification) to the forwarder, which removes the address encapsulation and sends the message on to the real destination.

Of course address encapsulation isn't exactly new. Source routes were defined in RFC 822 and provide exactly this sort of functionality, as does percent hack routing and bang paths. However, these mechanisms are all problematic on today's Internet since allowing their use effectively turns your system into an open relay.

SRS deals with this problem by adding a keyed hash and a timestamp to the encapsulation format. The address is only valid for some period of time, after which it cannot be used. The hash prevents modification of either the timestamp or the encapsulated address.

SRS also provides a mechanism for handling multi-hop forwarding without undue growth in address length. For this to work certain aspects of SRS address formatting have to be done in the same way across all systems implementing SRS.

The following MTA options have been added:

- **SRS_DOMAIN**. This must be set to the domain to use in SRS addresses. Email sent to this domain must always be routed to a system capable of SRS operations for the domain. SRS processing is handled as an overlay on top of normal address processing so nothing prevents a site from using their primary domain as the SRS domain.

- **SRS_SECRETS**. This is a comma separated list of secret keys used to encode and decode SRS addresses. The first key on the list is used unconditionally for encoding. For decoding, each key is tried in order to generate a different hash value. The decoding operation proceeds if any of the hashes match.

  The ability to use multiple keys makes it possible to change secrets without service disruption: Add a second key, wait for all previously issued addresses to time out, and then remove the first key.

- **SRS_MAXAGE**. Optionally specifies the number of days before a message times out. The default if the option isn't specified is 14 days.

  Every system that handles email for the selected SRS domain must be configured for SRS processing and must have all three SRS options set identically.

Setting these options is sufficient to enable SRS address decoding. Encoding is another matter and should only be done to envelope **From:** addresses you know are associated with forwarding activity. SRS encoding is controlled by six new channel options: **addresssrs**, **noaddresssrs**, **destinationsrs**, **nodestinationsrs**, **sourcesrs**, and **nosourcesrs**.

Three conditions have to be met for SRS encoding to occur:

1. The current source channel has to be marked with **sourcesrs**. (**nosourcesrs** is the default).
2. The current destination channel has to be marked with **destinationsrs**. (**nodestinationsrs** is the default).
3. The current address, when rewritten, has to match a channel marked **addresssrs**. (**noaddress** is the default).

Encoding only occurs when all of these conditions are true. The simplest setup consists of pure forwarding where all messages enter and exit on the **tcp_local** channel and all non-local addresses need SRS handling. In such a setup, **tcp_local** would be marked with the three options **sourcesrs**, **destinationsrs**, and **addresssrs**.

Finally, **imsimta test -rewrite** has been enhanced to show SRS encoding and decoding results for whatever address is input. For example, the address **foo@example.com** might produce the output similar to:

```
SRS encoding = SRS0=dnG=IS=example.com=foo@example.org
```

If this encoded address is rewritten it produces the following output:

```
SRS decoding = foo@example.com
```

imsimta test -rewrite also shows any errors that occur during SRS decoding.

27

Classic Message Store Directory Layout

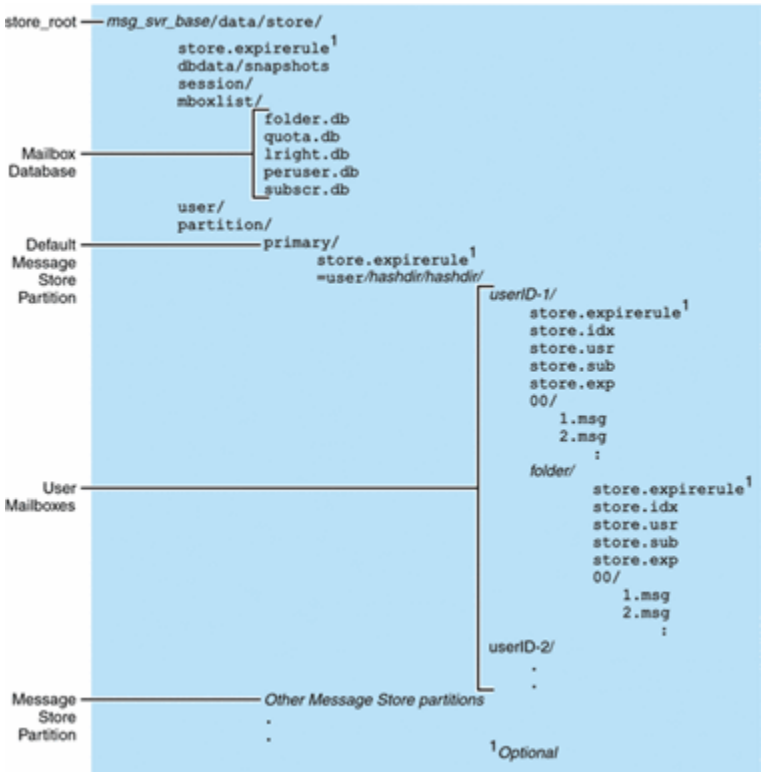
This chapter describes the directory layout of the Oracle Communications Message Server classic message store.

> **Note:** For overview and architecture information about Cassandra message store, see *Messaging Server Installation and Configuration Guide for Cassandra Message Store*.

About the Classic Message Store Directory Layout

Table 27–1 shows and describes the classic message store directory layout.

Figure 27–1 Message Store Directory Layout



The message store consists of several mailbox databases and the user mailboxes. The mailbox databases consists of information about users, mailboxes, partitions, quotas and other message store related data. The user mailboxes contain the user's messages

and folders. Mailboxes are stored in a *message store partition*, an area on a *disk* partition specifically devoted to storing the message store. See "Managing Message Store Partitions and Adding Storage" for details. Message store partitions are not the same as disk partitions, though for ease of maintenance, we recommend having one disk partition for each message store partition.

Mailboxes such as INBOX are located in the *store_root*. For example, a sample directory path might be:

```
store_root/partition/primary/=user/53/53/=mack1
```

Table 27–1 describes the message store directory.

Table 27–1 Message Store Directory Description

Location	Content/ Description
MessagingServer_home	Specifies installation location for the Messaging Server software. The default is **/opt/sun/comms/messaging64**. This is the directory on the Messaging Server machine that holds the server program, configuration, maintenance, and information files.
store_root	Specifies the top-level directory of the message store. The default is *DataRoot***/store**. Contains the **mboxlist**, **user**, and **partition** subdirectories.
./store.expirerule	Contains the automatic message removal rules (expire rules). This optional file can be at different locations. See "Message Store Message Expiration" for more information.
*store_root***/dbdata/snapshots**	Message store database backup snapshots that **stored** makes periodically.
*store_root***/mboxlist/**	Contains mailbox database, database (Berkeley DB) that stores information about the mailboxes and quota information. **annotate.db** supports the **ANNOTATE** extension to IMAP, which permits clients and servers to maintain "meta data" for messages, or individual message parts, stored in a mailbox on the server. For example, you could use **IMAP ANNOTATE** to attach comments and other useful information to a message, or to attach annotations to specific parts of a message, marking them as seen or important, or a comment added. **folder.db** contains information about mailboxes, including the name of the partition where the mailbox is stored, the ACL, and a copy of some of the information in **store.idx**. There is one entry in **folder.db** per mailbox. **quota.db** contains information about quotas and quota usage. There is one entry in **quota.db** per user. **lright.db** is an index for the folders by acl lookup rights. **peruser.db** contains information about per-user flags. The flags indicate whether a particular user has seen or deleted a message. **subscr.db** contains information about user subscriptions.
*store_root***/session/**	Contains active message store process information.
*store_root***/user/**	Not used.
*store_root***/partition/**	Contains the message store partitions. A default **primary** partition is created. Place any other partitions you define in this directory.
*store_root***/partition/primary/=user/**	Contains all the user mailboxes in the subdirectory of the partition. The mailboxes are stored in a hash structure for fast searching. To find the directory that contains a particular user's mailbox, use the **hashdir** utility.

Table 27–1 (Cont.) Message Store Directory Description

Location	Content/ Description
.../**=user/***hashdir/hashdir/userid/*	The top-level mail folder for the user whose ID is *userid*. This contains the user's INBOX. For the default domain, *userid* is *uid*. For hosted domains, *userid* is *uid@domain*. A user's incoming messages are delivered to the INBOX here.
.../userid/folder	A user-defined mailbox on the Messaging Server host.
.../userid/**store.idx**	An index that provides the following information about mail stored in the */userid/* directory: number of messages, disk quota used by this mailbox, the time the mailbox was last appended, message flags, variable-length information for each message including the headers and the MIME structure, and the size of each message. The index also includes a backup copy of **mboxlist** information for each user and a backup copy of quota information for each user.
.../userid/**store.usr**	Contains a list of users who have accessed the folder. For each user listed, contains information about the last time the user accessed the folder, the list of messages the user has seen, and the list of messages the user has deleted.
.../userid/**store.sub**	Contains information about user subscriptions.
.../userid/**store.exp**	Contains a list of message files that have been expunged, but not removed from disk. This file appears only if there are expunged messages.
`.../userid/ nn/ or .../userid/folder/nn/`	*nn* is a hash directory that contains messages in the format *message_id*.**msg**; *nn* can be a number from 00 to 99. *message_id* is also a number. Example: messages 1 through 99 are stored in the **.../00** directory. The first message is **1.msg**, the second is **2.msg**, third **3.msg**, and so on. Messages 100 through 199 are stored in the 01 directory; messages 9990 through 9999 are stored in the 99 directory; messages 10000 through 10099 are in the 00 directory, and so on.

28

Monitoring LDAP Directory Server

This chapter describes how to monitor and troubleshoot the LDAP directory server (**slapd**), which provides directory information for Oracle Communications Messaging Server. If **slapd** is down, the system does not work properly. If **slapd** response time is too slow, login speed and other transactions that require LDAP lookups are affected.

Symptoms of slapd Problems

- Client POP, IMAP, or Webmail Authentication fails or slower than expected.
- MTA not working properly

To Monitor slapd

- Check that **ns-slapd** process is running.
- Check **slapd** log files **access** and **errors** in **slapd-***instance***/logs/**.
- Check the **ns-slapd** response time while searching for a user.
- See also "immonitor-access".

29

Monitoring System Performance

This chapter focuses on Oracle Communications Messaging Server monitoring, however, you also need to monitor the system on which the server resides. A well-configured server cannot perform well on a poorly-tuned system, and symptoms of server failure may be an indication that the hardware is not powerful enough to serve the email load. This chapter does not provide all the details for monitoring system performance as many of these procedures are platform specific and may require that you refer to the platform specific system documentation.

Monitoring End-to-end Message Delivery Times

Email needs to be delivered on time. This may be a service agreement requirement, but also it is good policy to have mail delivered as quickly as possible. Slow end-to-end times could indicate several things. It may be that the server is not working properly, or that certain times of the day experience overwhelming message loads, or that the existing hardware resources are being pushed beyond their capacity.

Symptoms of Poor End-to-end Message Delivery Times

Mail takes a longer period of time to be delivered than normal.

To Monitor End-to-end Message Delivery Times

- Use any facility that sends a message and receives it. Compare the headers times between server hops, and times between point of origin and retrieval. See "immonitor-access" for more information.

Monitoring CPU Usage

High CPU usage is either a sign that there is not enough CPU capacity for the level of usage or some process is using up more CPU cycles than is appropriate.

Symptoms of CPU Usage Problems

Poor system response time. Slow logging in of users. Slow rate of delivery.

To Monitor CPU Usage

Monitoring CPU usage is a platform specific task. Refer to the relevant platform documentation.

30

Monitoring the Message Store

This chapter describes message store monitoring tasks. See "Managing the Message Store and Mailboxes" for conceptual information.

For more information about monitoring, see the following chapters:

- Monitoring Disk Space
- Monitoring User Access to the Message Store
- Using Message Store Log Messages

General Message Store Monitoring Procedures

This section outlines standard monitoring procedures for the message store. These procedures are helpful for general message store checks, testing, and standard maintenance.

Checking Hardware Space

A message store should have enough additional disk space and hardware resources. When the message store is near the maximum limit of disk space and hardware space, problems might occur within the message store.

Inadequate disk space is one of the most common causes of the mail server problems and failure. Without space to write to the message store, the mail server will fail. In addition, when the available disk space goes below a certain threshold, there will be problems related to message delivery, logging, and so forth. Disk space can be rapidly depleted when the clean up function of the **stored** process fails and deleted messages are not expunged from the message store.

See "Monitoring Disk Space" for information on monitoring disk space

Checking Log Files

Check the log files to make sure the message store processes are running as configured. Oracle Communications Messaging Server creates a separate set of log files for each of the major protocols, or services, it supports: SMTP, IMAP, POP, and HTTP. You can look at the log files in the *DataRoot***/log/** directory. You should monitor the log files on a routine basis.

Be aware that logging can impact server performance. The more verbose the logging you specify, the more disk space your log files will occupy for a given amount of time. You should define effective but realistic log rotation, expiration, and backup policies for your server. See "Using Message Store Log Messages" for information about defining logging policies for your server.

Checking User IMAP/POP/Webmail Session by Using Telemetry

Messaging Server provides a feature called telemetry that can capture a user's entire IMAP, POP or HTTP session into a file. This feature is useful for debugging client problems. For example, if users complain that their message access client is not working as expected, this feature can be used to trace the interaction between the access client and Messaging Server.

To capture a POP session, create the following directory:

*DataRoot***/telemetry/***pop_or_imap_or_http/userid*

To capture a POP session, create the following directory:

*DataRoot***/telemetry/pop/***userid*

To capture an IMAP session, create the following directory:

*DataRoot***/telemetry/imap/***userid*

To capture a Webmail session, create the following directory:

*DataRoot***/telemetry/http/***userid*

Note: *userid* is "uid" for default domain and "uid@domain" for hosted domains.

Note that the directory must be owned or writable by the messaging server userid.

Messaging Server will create one file per session in that directory. Example output is shown below.

```
LOGIN redb 2003/11/26 13:03:21
>0.017>1 OK User logged in
<0.047<2 XSERVERINFO MANAGEACCOUNTURL MANAGELISTSURL MANAGEFILTERSURL
>0.003>* XSERVERINFO MANAGEACCOUNTURL {67}
http://redb@cuisine.blue.planet.com:800/bin/user/admin/bin/enduser
MANAGELISTSURL NIL MANAGEFILTERSURL NIL
2 OK Completed
<0.046<3 select "INBOX"
>0.236>* FLAGS (\Answered flagged draft deleted \Seen $MDNSent Junk)
* OK [PERMANENTFLAGS (\Answered flag draft deleted \Seen $MDNSent Junk \*)]
* 1538 EXISTS
* 0 RECENT
* OK [UNSEEN 23]
* OK [UIDVALIDITY 1046219200]
* OK [UIDNEXT 1968]
3 OK [READ-WRITE] Completed
<0.045<4 UID fetch 1:* (FLAGS)
>0.117>* 1 FETCH (FLAGS (\Seen) UID 330)
* 2 FETCH (FLAGS (\Seen) UID 331)
* 3 FETCH (FLAGS (\Seen) UID 332)
* 4 FETCH (FLAGS (\Seen) UID 333)
* 5 FETCH (FLAGS (\Seen) UID 334)
<etc>
```

You can gather command telemetry that does not include end-user information by using the **imap.logcommands msconfig** option (or in legacy configuration **local.imap.logcommands**). See *Messaging Server Reference* for additional information.

To disable the telemetry logging, move or remove the directory that you created.

Checking stored Processes

The **stored** function performs a variety of important tasks such as deadlock and transaction operations of the message database, enforcing aging policies, and expunging and erasing messages stored on disk. If **stored** stops running, Messaging Server will eventually run into problems. If **stored** does not start when **start-msg** is run, no other processes will start.

- Check that the **stored** process is running. See "imcheck" for more information.
- Check for the log file build up in *store_root*/**mboxlist**.
- Check for **stored** messages in the default log file *DataRoot*/**log/default/default**.
- Check that the time stamps of the following files (in directory *MessagingServer_home*/**config/**) in Table 30–1 are updated whenever one of the following functions are attempted by the **stored** process:

Table 30–1 stored Operations

stored Operation	Function
stored.ckp	Touched when a database checkpoint was initiated. Stamped approximately every 1 minute.
stored.lcu	Touched at every database log cleanup. Time stamped approximately every 5 minutes.
stored.per	Touched at every spawn of peruser db write out. Time stamped once an hour.

See "stored" and "Monitoring the stored Process" for more information on the **stored** process.

Checking Database Log Files

Database log files refer to sleepycat transaction checkpointing log files (in directory *store_root*/**mboxlist**). If log files accumulate, then database checkpointing is not occurring. In general, there are two or three database log files during a single period of time. If there are more files, it could be a sign of a problem.

Checking User Folders

If you want to check the user folders, you might run the command **reconstruct -r -n** (recursive no fix) which will review any user folder and report errors. See "Repairing Mailboxes and the Mailboxes Database (reconstruct Command)" for more information on the **reconstruct** command.

Checking for Core Files

Core files only exist when processes have unexpectedly terminated. It is important to review these files, particularly when you see a problem in the message store. On Oracle Solaris, use **coreadm** to configure **core** file location.

Monitoring imapd, popd and httpd

These processes provide access to IMAP, POP and Webmail services. If any of these is not running or not responding, the service will not function appropriately. If the service is running, but is over loaded, monitoring will allow you to detect this and configure it more appropriately.

Symptoms of imapd, popd and httpd Problems

Connections are refused or system is too slow to connect. For example, if IMAP is not running and you try to connect to IMAP directly you will see something like this:

telnet 0 143 Trying 0.0.0.0... telnet: Unable to connect to remote host: Connection refused

If you try to connect with a client, you will get a message such as:

"Client is unable to connect to the server at the location you have specified. The server may be down or busy."

To Monitor imapd, popd and httpd

- Can be monitored with **watcher** and **msprobe**. See "Automatic Restart of Failed or Unresponsive Services" and "Monitoring Using msprobe and watcher Functions" for more information.
- Can be monitored with SNMP. If you have the SNMP set up, this is a very good way to monitor these processes (see "SNMP Support"). The server information is in the Network Services Monitoring MIB.
- Check log files. Look in the directory *MessagingServer_home*/**log**/*service* where *service* can be HTTP, IMAP, or POP. One filename is the name of the *service* (imap, pop, http) and the others are the name of the service plus a sequence number and a date concatenated to the service name. For example:

  imap imap.29.1010221593 imap.31.1010394412 imap.33.1010567224

The file with just the service name is the latest log. The other ones are ordered by the sequence number (here 29, 31, 33) and the one with the highest sequence number is the next newest one (see "Using Message Store Log Messages").

If a server was shut down you might see something like this:

```
imap.12.1065431243:[07/Oct/2003:01:15:43 -0700] gotmail-2 imapd[20525]: General
Warning: Sun Java System Messaging Server IMAP4 6.1 (built Sep 24 2003)
shutting down
```

- Can be checked with "counterutil".

  See "Gathering Message Store Counter Statistics by Using counterutil".
- Run the platform-specific command to verify that the imapd, popd and httpd processes are running. For example, in Oracle Solaris you can use the **ps** command and look for **imapd**, **popd** and **mshttpd**.
- You can set alarms for specified server performance thresholds by setting the server response configuration options described in "Alarm Messages".
- See "immonitor-access".

Monitoring the stored Process

"stored" performs a variety of important tasks such as deadlock and transaction operations of the message database, enforcing aging policies, and expunging and erasing messages stored on disk. If **stored** stops running, the messaging server will eventually run into problems. If **stored** does not start when **start-msg** is run, no other processes will start. See "stored" for more information.

Symptoms of stored Problems

There are no outward symptoms.

To Monitor stored

- Check that the **stored** process is running. **stored** creates and updates a **pid** file in *DataRoot*/**proc** called **store**. The **pid** file shows an **init** state when recovering and a **ready** state when ready. For example:

  ```
  231: cat store
  28250
  ready
  ```

  The number on the first line is the process ID of **stored**.

  ```
  232: ps -eaf | grep stored
  inetuser 28250 1 0 Jan 05 ? 8:44
  /opt/sun/comms/messaging64/lib/stored -d
  ```

- Check for log file build up in *MessagingServer_home*/**store/mboxlist**. Note that not every log file build up is caused by direct **stored** problems. Log files may also build up if **imapd** dies or there is a database problem.
- Check the timestamp on the following files in *MessagingServer_home*/**config**:

  stored.ckp - Touched when attempt at checkpointing is made. Should get time stamped every 1 minute.

  stored.lcu - Touched at every db log cleanup. Should get time stamped every 5 minutes.

  stored.per - Touched at every spawn of peruser db writeout. Should get time stamped every 60 minutes.
- Check for **stored** messages in the default log file *DataRoot*/**log/default/default**
- Can be monitored with **watcher** and **msprobe**. See "Automatic Restart of Failed or Unresponsive Services" and "Monitoring Using msprobe and watcher Functions" for more information.

Monitoring the State of Message Store Database Locks

The state of database-locks is held by different server processes. These database locks can affect the performance of the message store. In case of deadlocks, messages will not be getting inserted into the store at reasonable speeds and the **ims-ms** channel queue will grow larger as a result. There are legitimate reasons for a queue to back up, so it is useful to have a history of the queue length in order to diagnose problems.

Symptoms of Message Store Database Lock Problems

Number of transactions are accumulating and not resolving.

To Monitor Message Store Database Locks

Use the command "imcheck" **-s** (used to be **counterutil -o db_lock**).

To Monitor Mailbox Quotas and Usage

You can monitor mailbox quota usage and limits by using the "imquotacheck" utility. The **imquotacheck** utility generates a report that lists defined quotas and limits, and provides information on quota usage.

For example, the following command lists all user quota information:

```
imquotacheck
------------------------------------------------------------------------
Domain red.example.com (diskquota = not set msgquota = not set) quota usage
------------------------------------------------------------------------
diskquota         size(K)    %use    msgquota      msgs    %use    user
# of domains = 1
# of users = 705
no quota          50418             no quota      4392             ajonk
no quota              5             no quota      2                andrt
no quota         355518             no quota      2500             ansri
 ...
```

The following example shows the quota usage for user sorook:

```
imquotacheck -u sorook
------------------------------------------------------------------------
quota usage for user sorook
------------------------------------------------------------------------
diskquota      size(K)    %use    msgquota      msgs     %use    user
no quota       1487               no quota      305              sorook
```

To list the usage of all users whose quota exceeds the least threshold in the rule file:

imquotacheck

To list quota information for a the domain **example.com**:

imquotacheck -d example.com

To send a notification to all users in accordance to the default rule file:

imquotacheck -n

To send a notification to all users in accordance to a specified *rulefile*, *myrulefile*, and to a specified mail template file, *mytemplate.file* (for more information, refer to "imquotacheck"):

imquotacheck -n -r *myrulefile* **-t** *mytemplate.file*

To list per folder usages for one user user1 (will ignore the rule file):

imquotacheck -u user1 -e

To Monitor Message Store Database Statistics with imcheck

Use **imcheck -s** to monitor database statistics including logs and transactions. See "imcheck" for more information.

Note: The **imcheck -s** command is only valid for the classic message store.

Gathering Message Store Counter Statistics by Using counterutil

This section describes how to use the **counterutil** utility to gather message store statistics.

To Get a Current List of Available Counter Objects

This utility provides statistics acquired from different system counters (see "counterutil").

Here is how to get a current list of available counter objects:

```
counterutil -l
Listing registry (/opt/sun/comms/messaging64/data/counter/counter)
numobjects = 7
refcount = 20
created = 17/Mar/2015:14:10:03 +0000
modified = 24/Aug/2015:13:00:24 +0000
counterobjects:
  imapstat
  popstat
  alarm
  serverresponse
  diskusage
  httpstat
  mmpstat
```

Each entry represents a counter object and supplies a variety of useful counts for this object. In this section we will only be discussing the **alarm**, **diskusage**, **serverresponse**, **popstat**, **imapstat**, and **httpstat** counter objects. See "counterutil" for details on **counterutil** command usage.

counterutil Output

"counterutil" has a variety of flags. A command format for this utility may be as follows:

counterutil -o *CounterObject***-i 5 -n 10**

where,

-o *CounterObject* represents the counter object **alarm**, **diskusage**, **serverresponse**, **popstat**, **imapstat**, and **httpstat**.

-i 5 specifies a 5 second interval.

-n 10 represents the number of iterations (default: infinity).

An example of **counterutil** usage is as follows:

```
counterutil -o imapstat -i 5 -n 10
Monitor counteroobject (imapstat)
registry /gotmail/iplanet/server5/msg-gotmail/counter/counter opened
counterobject imapstat opened
count = 1 at 972082466 rh = 0xc0990 oh = 0xc0968
global.currentStartTime [4 bytes]: 17/Oct/2000:12:44:23 -0700
global.lastConnectionTime [4 bytes]: 20/Oct/2000:15:53:37 -0700
global.maxConnections [4 bytes]: 69
global.numConnections [4 bytes]: 12480
global.numCurrentConnections [4 bytes]: 48
global.numFailedConnections [4 bytes]: 0
global.numFailedLogins [4 bytes]: 15
```

```
global.numGoodLogins [4 bytes]: 10446
...
```

Gathering Alarm Statistics by Using counterutil

These alarm statistics refer to the alarms sent by **stored**. Table 30–2 shows the statistics provided by the alarm counter.

Table 30–2 counterutil alarm Statistics

Suffix	Description
alarm.countoverthreshold	Number of times crossing threshold.
alarm.countwarningsent	Number of warnings sent.
alarm.current	Current monitored valued.
alarm.high	Highest ever recorded value.
alarm.low	Lowest ever recorded value.
alarm.timelastset	The last time current value was set.
alarm.timelastwarning	The last time warning was sent.
alarm.timereset	The last time reset was performed.
alarm.timestatechanged	The last time alarm state changed.
alarm.warningstate	Warning state (yes(1) or no(0)).

IMAP, POP, HTTP, and MMP Connection Statistics by Using counterutil

To get information on the number of current IMAP, POP, HTTP, and MMP connections, number of failed logins, total connections from the start time, and so forth, you can use the command **counterutil -o***CounterObject***-i 5 -n 10**. Where *CounterObject* represents the counter object **popstat**, **imapstat**, **httpstat**, or **mmpstat**. For **mmpstat**, we have modified the counter names to differentiate the services IMAP and POP since the MMP proxies both. The meaning of the **imapstat** suffixes is shown in Table 30–3. The **popstat** and **httpstat** objects provide the same information in the same format and structure.

Table 30–3 counterutil imapstat Statistics

Suffix	Description
currentStartTime	Start time of the current IMAP server process.
lastConnectionTime	The last time a new client was accepted.
maxConnections	Highest recorded number of concurrent TCP connections handled by IMAP server since the last counter reset.
numConnections	Total number of TCP connections successfully accepted by the current IMAP server. **numConnections** can include failed connections, but not always.
numCurrentConnections	Current number of active TCP connections.

Table 30–3 (Cont.) counterutil imapstat Statistics

Suffix	Description

numFailedConnections	Total number of failed TCP connections by the current IMAP server. This number accumulates until the server restart or reset by "counterutil". **numFailedConnections** counts connections abnormally terminated, including unsuccessful accepts and connections successfully accepted but which had an error later. An error message is logged when a connection failed with an expected error. You can check your IMAP log files for error messages such as the following:
	`Unable to accept client connection: <error message>`
	`Socket error : <error message>`

numFailedLogins	Number of failed system logins served by the current IMAP server.
numGoodLogins	Number of successful system logins served by the current IMAP server.

Disk Usage Statistics by Using counterutil

Table 30–4 shows the information generated by the **counterutil -o diskusage** command.

Table 30–4 counterutil diskusage Statistics

Suffix	Description
diskusage.availSpace	Total space available in the disk partition. The values are scaled to fit in the 4 byte counter. If you have a very large file system, the actual number will be divided by 1024 until it is small enough to fit in the 32-bit integer.
diskusage.lastStatTime	The last time statistic was taken.
diskusage.mailPartitionPath	Mail partition path.
diskusage.percentAvail	Disk partition space available percentage.
diskusage.totalSpace	Total space in the disk partition. The values are scaled to fit in the 4 byte counter. If you have a very large file system, the actual number will be divided by 1024 until it is small enough to fit in the 32-bit integer.

Server Response Statistics

Table 30–5 shows the information generated by the **counterutil -o serverresponse** command. This information is useful for checking if the servers are running, and how quickly they're responding.

Table 30–5 counterutil serverresponse Statistics

Suffix	Description
http.laststattime	Last time http server response was checked.
http.responsetime	Response time for the http.
imap.laststattime	Last time imap server response was checked.
imap.responsetime	Response time for the imap.
pop.laststattime	Last time pop server response was checked.

Table 30–5 (Cont.) counterutil serverresponse Statistics

Suffix	Description
pop.responsetime	Response time for the pop.

31

Monitoring User Access to the Message Store

This chapter describes the **imsconnutil** command, which enables you to monitor user's message store access via IMAP, POP and HTTP. You can also determine the last login and logout of users. This command works on a per message store basis and does not work across message stores.

> **Note:** Use of this function or other Oracle Communications Messaging Server functions to monitor, read or otherwise access user's email may constitute a potential source of liability if used in violation of applicable laws or regulations or if used in violation of the customer's own policies or agreements.

This command requires root access by the system user (default: **mailsrv**), and you must set the configuration variables **imap.enableuserlist** and **http.enableuserlist** to 1.

To list users currently logged on via IMAP or any web mail client, use the following command:

imsconnutil -c

To list the last IMAP, POP, or HTTP access (log in and log out) of every user on the message store use:

imsconnutil -a

The following command does two things: 1) it determines whether the specified user is currently logged on via IMAP or HTTP or any client that connects via mshttp (note that this does not work for POP because POP users generally do not stay connected), and 2) it lists the last time the users have logged on and off:

imsconnutil -c -a -u *user_ID*

Note that a list of users can be input from a file, one user per line, using the following command:

imsconnutil -c -a -f *filename*

You can also specify a particular service (**imap** or **http**) using the *-s* flag. For example, to list whether a particular user ID is logged onto IMAP or not, use the following command:

imsconnutil -c -s imap -u *user_ID*

Note that the **-k** option uses ENS to send the disconnect message to the servers.

See "imsconnutil" for a complete description of the **imsconnutil** syntax. Here is some example output:

```
imsconnutil -a -u soroork
UID     IMAP last access    HTTP last access    POP last access
========================================================================
ed   08/Jul/2003:10:49:05   10/Jul/2003:14:55:52  ---NOT-RECORDED---
$ imsconnutil -c
IMAP
UID    TIME                AUTH            TO                  FROM
========================================================================
ed   17/Jun/2003:11:24:03  plain     172.58.73.45:193   129.157.12.73:2631
bil  17/Jun/2003:04:28:43  plain     172.58.73.45:193   129.158.16.34:2340
mia  17/Jun/2003:09:36:54  plain     172.58.73.45:193   192.18.184.103:3744
jay  17/Jun/2003:05:38:46  plain     172.58.73.45:193   129.159.18.123:3687
pau  17/Jun/2003:12:23:28  plaintext 172.58.73.45:193   192.18.194.83:2943
ton  17/Jun/2003:05:38:46  plain     172.58.73.45:193   129.152.18.123:3688
ani  17/Jun/2003:12:26:40  plaintext 172.58.73.45:193   192.18.164.17:1767
ani  17/Jun/2003:12:25:17  plaintext 172.58.73.45:193   129.150.17.34:3117
jac  17/Jun/2003:12:26:32  plaintext 172.58.73.45:193   129.150.17.34:3119
ton  17/Jun/2003:12:25:32  plaintext 172.58.73.45:193   192.18.148.17:1764
========================================================================
10 users were logged in to imap.
Feature is not enabled for http.
------------------------------------------------------------------------
```

32

Message Archiving

This chapter describes archiving concepts for Oracle Communications Messaging Server. It does not provide instructions on how to set up an archiving system.

Microsoft Exchange Envelope Journaling

Messaging Server is able capture sieve action to produce Microsoft Exchange's "envelope journaling" format. This format consists of a multipart MIME message where the first part contains envelope information is a semi-structured format and the second part is the actual message. This new format by specifying a **:journal** option to capture:

```
capture :journal "trigger-address";
```

Exchange Journal Format Archiving for IMAP APPEND with LDAP Attributes

Support has been added to the archiving library to produce Microsoft Exchange Journal format archive messages. Note that this support extends to store compliance archiving of IMAP APPENDs as well as the archiving plugin.

In the case of the archiving plugin, the **STYLE** option accepts a value of 3, indicating that Microsoft Exchange Journal format messages should be produced. Additionally, two option file options have been added:

- **SOURCE_CHANNEL** (channel name; no default; required for Microsoft Exchange Journal format only)

  The **SOURCE_CHANNEL** option specifies the name of the channel used to submit Microsoft Exchange Journal format messages. We recommend that you create a separate channel for this purpose so that such submissions are clearly identifiable in the logs.

- **DESTINATION** (string; no default; Microsoft Exchange Journal format only).

  If set, this option specifies the address where Exchange Journal format archive messages are to be sent. If the option is not set archive messages are sent to the various capture attributes associated with the message's authorized sender, envelope from, and envelope recipient addresses.

In the case of store compliance archiving, there are three options:

- **store.archive.style** - Same semantics as the **STYLE** option file option.
- **store.archive.source_channel** - Same semantics as the **SOURCE_CHANNEL** option file option.
- **store.archive.destination** - Same semantics as the **DESTINATION** option file option.

Archiving Overview

A message archiving system saves all or specified incoming and outgoing messages on a system separate from Messaging Server. Sent, received, deleted, and moved messages can all be saved and retrieve in an archive system. Archived messages cannot be modified or removed by email users, so the integrity of incoming and outgoing messages is maintained. Message archiving is useful for compliance record keeping, but it is also useful for message store management. For example, some customers may use archiving to perform message back-up or to move older messages from more expensive message store storage to less expensive archive storage.

Archived messages can be accessed through a separate archiving software GUI client or through Messaging Server. If the messages are deleted from Messaging Server, then the archiving client can be used to search for and retrieve those deleted messages since archived messages are never deleted. Note, however, that archived messages are not stored in mailbox folders as they are in the Messaging Server.

The system can also be set up so that archived messages can be accessed from Messaging Server. For example, you can set up a system to archive messages over 2 years old. Instead of having message bodies reside in the message store, they would instead reside in the archive system. From the users standpoint, the message appears no different from a regular email message. The same header and subject information will appear (this is still stored in the message store storage), but the message body is downloaded from the archive server by the message store when needed. Thus, there may be a slight delay as messages are downloaded from the archive server. In addition, archived messages cannot be searched from the email client. Searching must be done from the archiving GUI.

Message Archiving Systems: Compliance and Operational

There are two types of archiving, compliance and operational. Compliance archiving is used when you have a legal obligation to maintain strict retrievable email record keeping. Selected email (selected by user(s), domain, channel, incoming, outgoing and so on) coming into the MTA is copied to the archive system before being delivered to the message store or the internet. Archiving can be set to occur either before or after spam and virus filtering.

Operational archiving is used for mail management purposes. For example:

- To reduce storage usage on the Messaging Server message store by moving less used (older) messages to an archiving system which uses lower cost storage.
- As an alternative for data backup.

Note that compliance and operational archiving are not exclusive. That is, you can set up your system so that it does both compliance and operational archiving.

33

Unified Messaging

This chapter describes how to use Oracle Communications Messaging Server Unified Messaging solution to receive, store, and manage all types of messages--voice, fax, email, video--on one, off-the-shelf, cost-effective system.

Using Messaging Server to Manage Unified Messaging

This document describes Oracle's solution to receive, store, and manage all types of messages--voice, fax, email, video--on one, off-the-shelf, cost-effective system. This paper is intended for telephony engineers and decision-makers interested in using standard Internet email software to manage voicemail, video, and fax communications.

See "Designing and Coding Your Unified Messaging Application" for implementation details.

What Is the Challenge?

As broadband service providers compete to integrate internet communications (email and instant messaging) and voice communications (voice portal, voicemail, fax, and voice conferencing), they must reduce the costs of creating and maintaining those integrated services. To do this, telephone companies and their network equipment manufacturers (NEPs) must adopt open standards and use cost-effective, off-the-shelf storage technologies.

The Oracle Solution

Oracle's leadership in this area allows it to provide, with its partners, a solution that supports the convergence of these communication services, called unified messaging.

Unified messaging provides the following benefits:

- Multi-modal access: the ability to access different types of messaging services using different types of technology. For example, using the phone to access email messages or using the computer to access voices messages.
- A cost-effective, off-the-shelf technology for storing voice, email, video, and fax messages on the same system.
- The ability to manage all types of messages using the same administrative procedures. This includes expiring old messages, setting message space quotas, archiving, logging, usage profiles, and so on.
- Access to technologies like text-to-speech (TTS) and automated speech recognition (ASR).

Using open standards and best practices, Messaging Server offers a single back-end service that receives, stores, manages, and expires messages, no matter what their type.

Open Standards and Regulatory Requirements

In response to the need for open standards, engineering leaders have created the Voice Profile for Internet Mail (VPIM) specification, which defines the interfaces between voice front-ends and IMAP-based message stores. These standards, integrated into and implemented by Messaging Server, provide a substantial savings over traditional proprietary voice storage. Furthermore, unified messaging applications can take advantage of the efficient mailbox operations and system-performance knowledge provided by Messaging Server, leveraging the expertise and technology that have deployed hundreds of millions of Messaging Server mailboxes around the world.

Today's new legal and regulatory realities require an effective and unobtrusive mechanism for legal mail interception (LMI). Messaging Server enables service-provider personnel to apply the same LMI mechanisms already used on email to voice and fax interception and recovery. These mechanisms both capture messages on the network and archive messages stored on disk.

Architectural Overview of a Unified Messaging Application

The following sections define a high-level architecture for a unified messaging application using Messaging Server for communications storage, management, and notifications.

This architectural view follows the lifecycle of the message types (voice, fax, email), divided into the following stages:

1. An incoming message is deposited in the message store, and a message-waiting indicator is turned on.
2. The end user retrieves the message via the Telephone User Interface (TUI). The message-waiting indicator is turned off.
3. Alternatively, the end user retrieves the message via an IMAP messaging client (such as Thunderbird) or Convergence.
4. The Messaging Server message store administrator uses the Messaging Server's Mailbox Administration and Operations Management (MAOM) functions to administer and expire the message.

The sections that follow take snapshots of the communications workflow at each one of these stages.

Message Deposit

Figure 33–1 shows an overview of the message flow of a voice message or fax message in Unified Messaging.

Figure 33–1 Message Deposit Function of Unified Messaging for Telecommunications

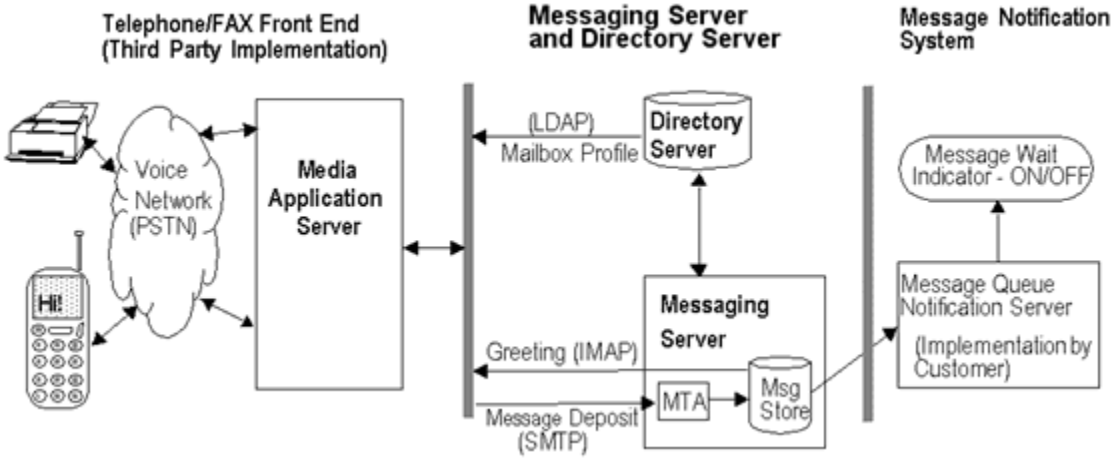
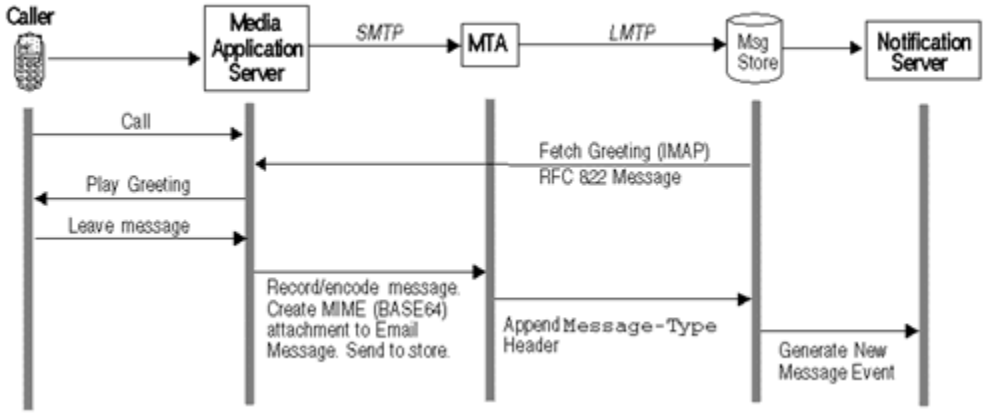


Figure 33–2 shows the message flow of a voice message or fax message when it first arrives and is deposited in the message store.

Figure 33–2 Message Deposit Function--State Diagram



1. **Receive the call.**

   The Public-Switched Telephone Network (PSTN) receives a phone call or FAX intended for the end user and passes it on to the Media Application Server.

   A Media Application Server is a third-party system that services voice and FAX callers. It is similar to a home answering machine in that it picks up calls, plays a pre-recorded message, handles touch-tone interactions, and retrieves and stores voice messages and so on. In many cases, the Media Application Server sits on a voice trunk with many ports; some servers support thousands of ports. A port is equivalent to one telephone line.

2. **Profile the call with LDAP data.**

   Using the end user's phone number, the Media Application Server looks up the LDAP entry for that number in Directory Server. Directory Server returns the LDAP profile to the Front-end Server.

   The types of profile information retrieved can include:

   - Status of the mailbox: Available, Disabled, Vacation

- Allowed services
- Current greeting for the user.

3. **Retrieve and play voicemail greeting.**

   The Media Application Server retrieves the greeting voice file from the message store and plays it to the caller. For example: "Hello, you've reached..." It prompts the caller to leave a message.

4. **Record the message and release caller.**

   The Media Application Server records the caller's message in binary file format, usually WAV format for voice and TIFF for FAX data. The caller hangs up.

5. **Encode the voice/FAX message.**

   The Media Application Server creates a email message, attaches the caller's encoded message file to it, and addresses it to the user's mailbox.

   - It creates an RFC2822 email message.
   - It attaches the voice message file as a binary attachment (base64). This format conforms to the Voice Profile for Internet Mail (VPIM) standard.
   - It addresses the message to the user's mailbox, using a standard email address--for example, 555-555-5555@example.com.

     The voice message is now an email message with a large attachment.

6. **Send email to message store.**

   The Media Application Server sends the email message via SMTP to the Messaging Server. The Messaging Server Message-Transfer Agent (MTA) accepts the email and routes it to the user's mailbox in the message store.

   The message store, a component of Messaging Server, stores and manages user mailboxes.

7. **Generate a new-message event notification.**

   This is to notify the system that a new message has arrived for this mailbox and is available for retrieval. The notification, and the actions taken in response to the notification--for example, turning on the message wait indicator--must be implemented by the customer.

   Message Queue provides the infrastructure for producing and distributing event notifications.

Message Retrieval via Telephone User Interface

Figure 33–3 shows the message flow and data states of a voice or fax message as the end user picks it up (listens to it or has the fax machine print it).

Figure 33–3 Message Retrieval via Telephone User Interface

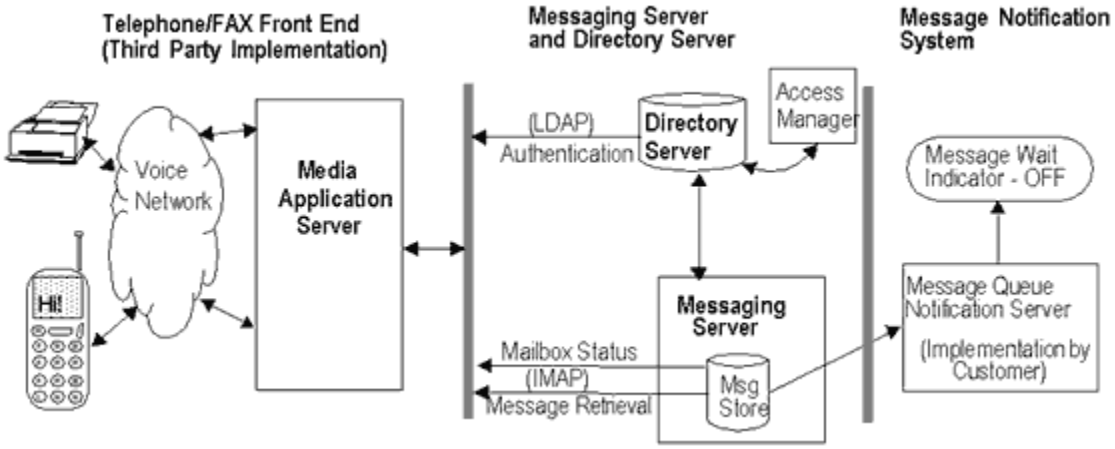
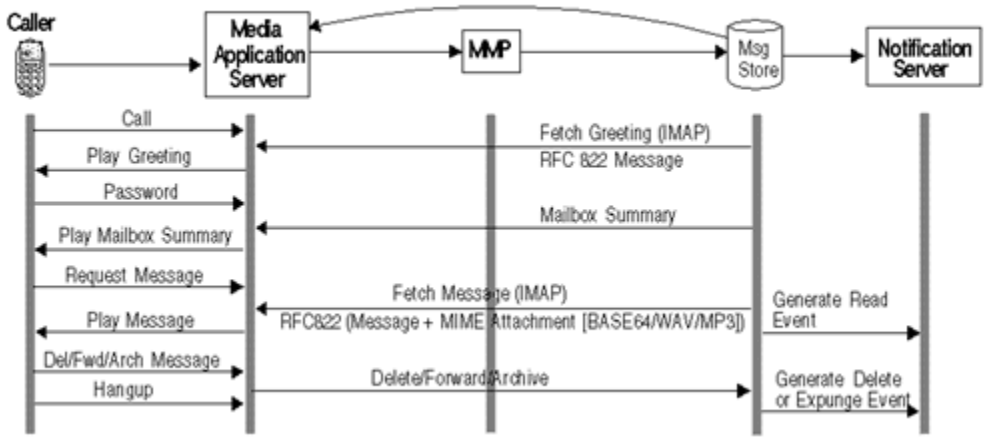


Figure 33–4 shows the message flow of the message retrieval function of a voice or fax message as the end user answers it.

Figure 33–4 Message Access and Retrieval via Telephone User Interface--State Diagram



These diagrams illustrate the following actions:

1. **User dials in and system profiles user mailbox.**

   In this implementation of the Media Application Server, the end user dials into the voice mail system. Using the end user's phone number, the Telephone/FAX Media Application Server looks up the LDAP entry for that number in Directory Server. Directory Server with Access Manager authenticates the user.

   After the authentication, the Media Application Server then looks at the user's Message Store server from the user's LDAP mailHost attribute. The system prompts the user for the password.

2. **User enters password.**

3. **System retrieves new mailbox summary.**

   The Media Application Server performs an IMAP connection to the mailbox owner's Message Store. The IMAP connection may pass through a component called the Messaging Multiplexor (MMP) which routes connection requests to the

appropriate message store. It then opens the user's folders to check for number of new messages (for example, number of new voicemail, fax and email messages). The Media Application Server then plays the number of new messages of each type to the user over the phone.

Note: Steps 3 and 4 are separate procedures that are executed at the same time.

4. **System retrieves and plays messages.**

   The mailbox caller selects either voicemail or fax mailbox. Using IMAP, the Media Application Server retrieves the message from its message store The Media Application Server then plays each Message header. For voicemail, it plays the message.

   - To play the voice message, the Media Application Server retrieves the RFC822 message. It decodes the attachment and plays the audio file (typically .WAV or mp3) over the phone line.
   - To play fax messages, the Media Application Server sends information about the fax (for example, date/time, caller number, number of pages, urgency, etc.). Typically, mailbox owners will forward these fax messages to fax machines nearby to them.
   - To play email messages, the Media Application Server sends info about the email (for example sender, subject, time/date, urgency). If text-to-speech has been implemented, the server will attempt to "read" the message to the caller. Some implementations allow the caller to forward the fax and the attachments to a nearby fax machine. After playing the message, the caller can delete, replay or forward the message.

   The message itself remains in the user's mailbox in the message store, but the status of the message flag is changed to "seen/read."

5. **User replays, deletes, forwards or archives the message.**

6. **System generates a message-read or message-deleted event notification.**

   This is to notify the system that the message has been read or deleted. The notification, and the actions taken in response to the notification--for example, turning off the message wait indicator--must be implemented by the customer.

   Message Queue provides the infrastructure for producing and distributing event notifications.

Message Retrieval via PC

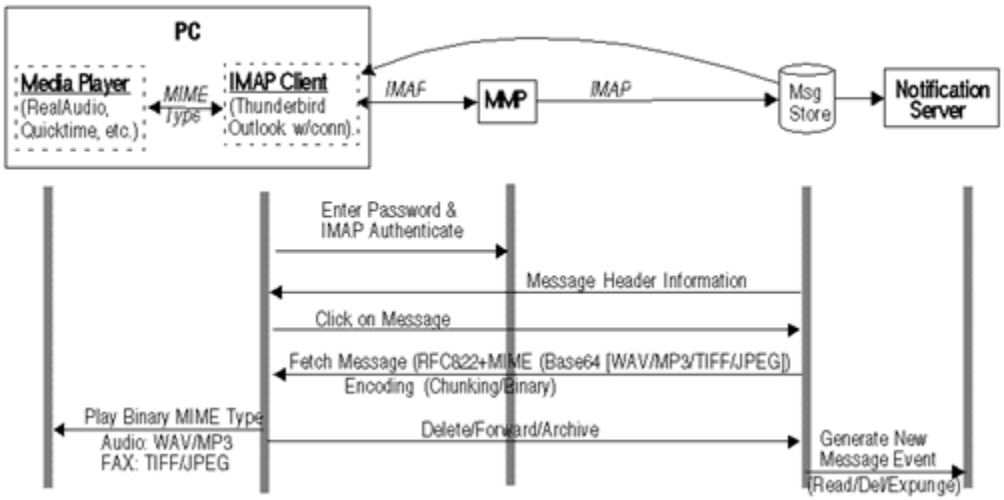
Messaging Server provides two ways to access messages through a PC: through an IMAP client such as Thunderbird or Outlook (with Connector) and through the, a web-based communication client, which uses an HTTP connection.

These will be described in separate sections.

Message Retrieval Through an IMAP Client

Figure 33–5 shows the message flow and data state of a voice or fax message as it is retrieved through an IMAP client.

Figure 33–5 IMAP Client Message Retrieval--Component and State Diagram



This diagram illustrates the following actions:

1. **User opens IMAP Client which connects to Messaging Server.**
2. **User enters the password and is authenticated by the MMP.**

   The MMP finds the appropriate message store and sets up a direct connection between the store and the client.
3. **Message store sends message header information to the IMAP client.**
4. **User clicks the message header to open.**

   Client issues an IMAP FETCH command and the message store returns the RFC822 email message with the MIME attachment containing the voice/FAX data. The format of the data could be in .WAV, MP3, TIFF or JPEG.
5. **User clicks the attachment.**

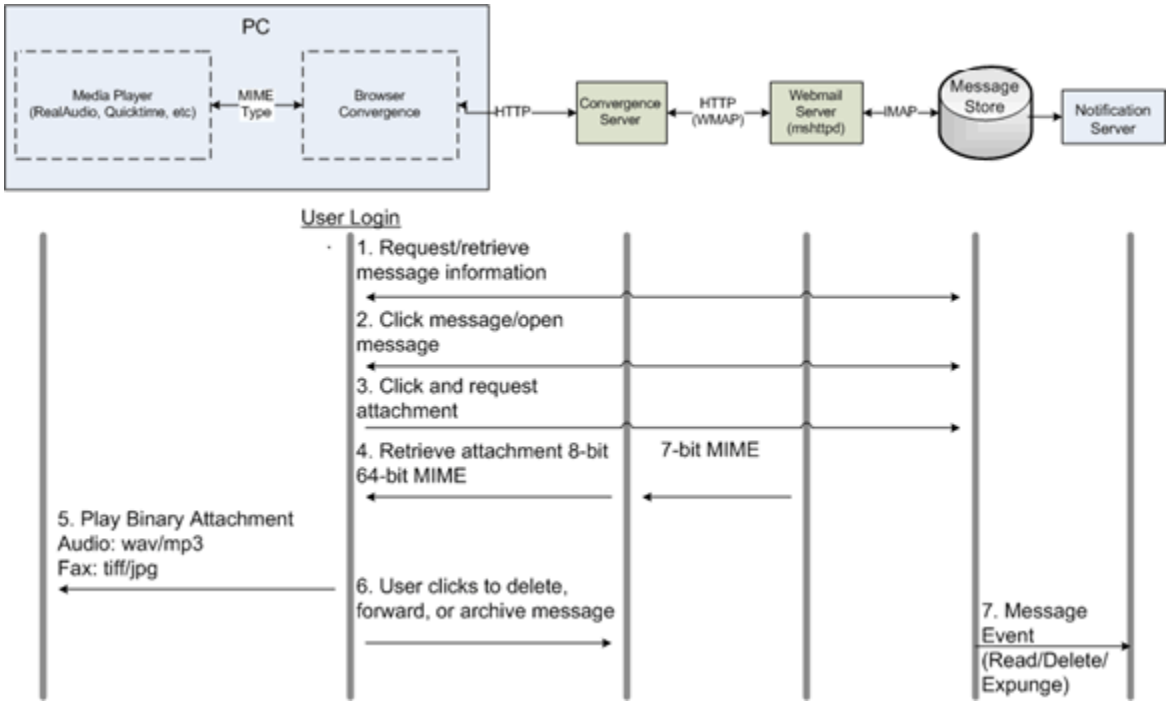
   IMAP client launches the media player on the PC, then strips off the wrappers and encoding in the attachment, and sends the resulting binary file to the media player.
6. **User deletes, forwards, or archives the message.**

   IMAP client sends command to the message store to delete, forward or archive the message.
7. **Message store generates an event notification indicating the changed status (deleted/forwarded/archived) of the message.**

Message Retrieval Through Convergence

Figure 33–6 shows the message flow and data state of a voice or fax message as it is retrieved through the Convergence client.

Figure 33–6 Convergence Message Retrieval--Component and State Diagram



This diagram illustrates the following actions:

1. **User logs in with password to the Convergence Server through a browser and retrieves the message headers.**

   Convergence Server makes a HTTP request to the Webmail Server for the message headers. The Webmail Server requests and retrieves the headers via IMAP and returns it to Convergence in HTTP where the user can view it.

   The Webmail Server translates HTTP commands to IMAP, and IMAP commands to HTTP. Note that the message store stores data in 7-bit format, and that http transfers data in 8-bit binary format.

2. **User clicks the message header to open.**

   Again, Convergence Server makes a HTTP request to the Webmail Server for the message. The Webmail Server requests and retrieves the message via IMAP and returns it to Convergence in HTTP where the user can view it. The message does not contain the MIME attachment, but contains a link to the attachment.

3. **User clicks the attachment.**

   Again, an HTTP request is made to the Webmail Server, which translates the request to IMAP and sends it to the message store.

4. **Retrieve the MIME attachment.**

   The Webmail Server retrieves the MIME attachment, strips off the wrappers and encoding in the attachment, and sends the resulting base64 binary file via Convergence to the browser which launches media player.

5. **The media player plays and displays the attachment.**

6. **User deletes, forwards, or archives the message.**

   Convergence sends command to the message store to delete, forward or archive the message.

7. **Message store generates an event notification indicating the changed status of the message (deleted/forwarded/archived).**

Designing and Coding Your Unified Messaging Application

This section describes how to design and configure Messaging Server to implement a unified messaging application.

Planning the Message-Type Configuration

To administer messages of different types, all components of the UM system must use the same message-type definitions and the same header fields to identify the messages.

Before you configure Messaging Server to support message types, you must

- Plan which message types you intend to use
- Decide on the definition for each message type
- Decide which header field to use

For example, if the application includes phone messages, you can define this message type as "multipart/voice-message" and use the Content-Type header field to identify message types.

You would then configure the Media Application Server to add the following header information to each phone message to be delivered to the message store:

```
Content-Type: multipart/voice-message
```

Next, you would configure the MTA and/or message store to recognize the **multipart/voice-message** message type.

Coding and Configuring Your UM System

This section takes a closer look at the message life cycle described in earlier sections. Here we focus on the stages in the life cycle where you must design, code, and configure components of the UM system. We give you an idea of the Media Server coding and Messaging Server configuration that you will perform to implement your UM system.

1. **Unanswered call is routed to the Media Server.**

   Before the Media Server sends a message to Messaging Server, it must check the status of the user's mailbox to ensure that the user's mailbox is not full or busy.

   The Media Server must be coded to check the mailuserstatus attribute in the user's entry in the directory server to see, for example, if the user's mailbox is full or not. You also must configure quota enforcement on the Message Store to enable this feature.

   See "Administering Quotas for Message Types" for an introduction to quota enforcement by message type. See "Managing Message Store Quotas" for details about configuring quotas for the message store.

2. **The Media Server encodes the voicemail as a binary attachment to an email message, which it sends to the MTA. The message is labeled by type.**

   (Of course, you must code the Media Server to perform the voicemail-to-email state transformation.)

When a message comes into the MTA via the Media Server, the first thing that must be done is to add a Content-type header to the message. The value of this header identifies the message type.

In this way, the messages can be managed according to their type. For example, voicemail types may have different quotas than text types.

You must define each message type with a unique identifier such as **multipart/voice-message**.

Content-type headers can be inserted in the message by the Media Server or by the MTA.

Typing by Media Server. When the Media Server constructs the email message to send to the MTA, it can be coded to add the **Content-type** header with the appropriate message-type value. For example, a voicemail message could require that **Content-type: multipart/voice-message** be added to the message.

Typing by MTA. The alternative is to set up your system so that the MTA adds the **Content-type** header. This can be done by configuring separate, nonstandard ports for the various message types.

Typically, email comes into the MTA via port 25. Thus, for example, you can configure text email to go to the standard port 25, voicemail to port 225, FAX to port 325. You must configure the Media Server to send the message to the appropriate port. Also, you must configure the MTA to append the appropriate **Content-type** header to messages arriving at each port.

3. **The MTA deposits the message in the message store, and an IMAP flag identifying the message type is appended to the message.**

   The message store reads the **Content-type** header to identify the message type.

   You can configure the MTA or the message store to append a message-type flag to the message. You must define unique values for each message-type flag.

   Messaging Server presents the message-type flag as a user flag to IMAP clients. (This flag cannot be modified by end users.)

   Mapping the message type to a user flag allows mail clients to use simple IMAP commands to manipulate messages by message type.

4. **Client retrieves mailbox status and displays the message together with its type.**

   From the user's perspective, the client email software may display an icon indicating the type of each message (if the client supports this feature). For example, a phone icon appears next to a voicemail message.

   The IMAP SEARCH command, using the message-type flag as a keyword, retrieves a count of each type of message. The IMAP FETCH command retrieves the message headers with the message status and message-type flag name.

   See "Sample IMAP Sessions Using Message-Type Flags" for examples.

5. **User clicks on the voicemail attachment, and the voicemail is played by a media player on the PC.**

   See "Message Retrieval Through Convergence" for more information.

6. **Message store generates a notification indicating the changed status of the message.**

   If the status of a messages changes, the message store can generate a notification that can be retrieved by the email client or the Media Server. A notification can

deliver a count of the messages in a user's mailbox for each message type and for each change in message status.

For example, a notification can deliver a count of all new (unread) voicemail messages and all new text messages in a user's mailbox. When the user listens to a voicemail, another notification can be generated indicating all voicemail messages that have been read and all text messages that have been read. In this case, the number of new voicemail messages is reduced by one.

You must configure Messaging Server to determine how and when to generate notifications. Also, you must write Media Server code to retrieve the notification and take appropriate action. For example, when a new message arrives in the store, you may want to send an indicator to the customer's phone. See "Delivering Notifications for Message Types" for details.

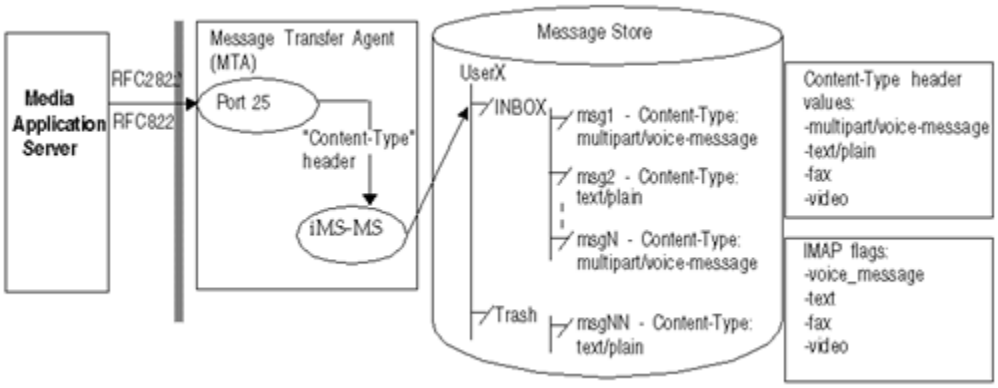
Mailbox Administration and Operations

The Messaging Server message store can be configured to identify and manage message types.

The customer does not have to maintain different message types in individual mailbox folders. The message store can identify a message type, no matter where the message is stored. Thus, you can store heterogeneous message types in the same folder.

Figure 33–7 illustrates how an incoming message is identified by its type.

Figure 33–7 Message Management by the Messaging Server Message Store



In this diagram, the message store identifies the message type of an incoming message by reading the **Content-type** header. In addition, the message store appends an IMAP flag identifying the message type (if the IMAP flag has not already been appended by the MTA).

Once message types have been configured, the message store lets you

- Set flags that allow IMAP commands to fetch and search for information about message types
- Configure quota roots that apply to each message type
- Write expire rules to expire and purge messages according to message type

Sample IMAP Sessions Using Message-Type Flags

This section describes sample IMAP sessions using IMAP FETCH and IMAP SEARCH.

Example 1: IMAP FETCH Session

The following IMAP session fetches messages for the currently selected mailbox:

```
2 fetch 1:2 (flags rfc822)
* 1 FETCH (FLAGS (\Seen text) RFC822 {164}

Date: Wed, 8 July 2006 03:39:57 -0700 (PDT)
From: bob.smith@example.com
To: john.doe@example.com
Subject: Hello
Content-Type: TEXT/plain; charset=us-ascii

* 2 FETCH (FLAGS (\Seen voice_message) RFC822 {164}

Date: Wed, 8 July 2006 04:17:22 -0700 (PDT)
From: sally.lee@example.com
To: john.doe@example.com
Subject: Our Meeting
Content-Type: MULTIPART/voice-message; ver=2.0

2 OK COMPLETED
```

In the preceding example, two messages are fetched, one text message and one voice mail.

The **Content-type** header fields identify the message types. The message-type names are displayed as they were received in the incoming messages.

Example 2: IMAP SEARCH Session

The following IMAP session searches for voice messages for the currently selected mailbox:

```
3 search keyword voice_message
* SEARCH 2 4 6
3 OK COMPLETED
```

In the preceding example, messages 2, 4, and 6 are voice messages. The keyword used in the search, **voice_message**, is the flag name defined for voice messages.

Administering Quotas for Message Types

When you set a quota for a message type, you include that value in a quota root. A quota root specifies quotas for a user.

You can specify the following quotas for a user's mailbox tree:

- Quota values for specific folders in the user's mailbox
- Quota values for specific message types such as voice mail or text messages. A message type quota applies to messages of that type in all folders in the user's mailbox.
- A default quota value that applies to all folders and message types in the user's mailbox that are not explicitly assigned quotas.

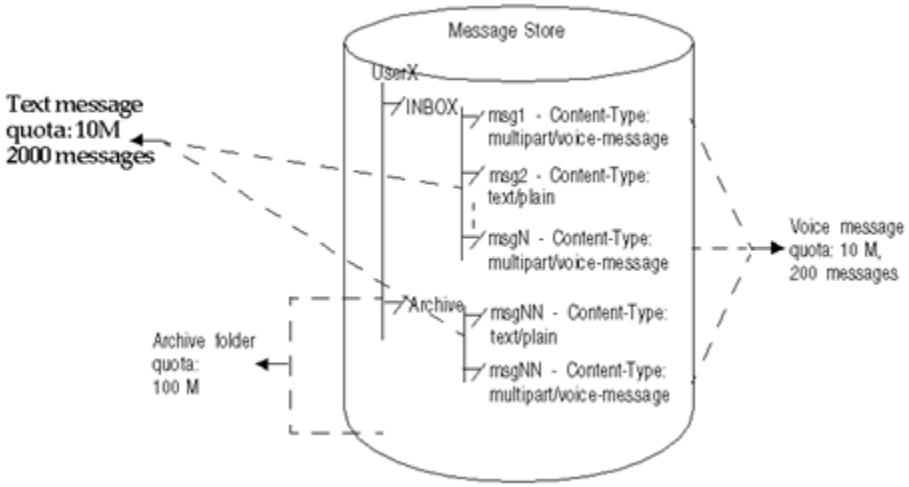
Quotas can be configured for the number of messages allowed and for the maximum amount of disk storage used.

Example of a Message-Type Quota Root

Suppose a customer wants to configure separate quotas for text messages and voice messages in each user's mailbox. Yet another quota is to be set for the user's Archive folder.

Figure 33–8 illustrates this example.

Figure 33–8 Administering Quotas in the Message Store



This example sets the following quotas for a user:

- The storage quota for the Archive folder is 100M
- The storage quota for text message types is 10M
- The message quota for text message types is 2000
- The storage quota for voice message types is 10M
- The message quota for voice message types is 200
- The default mailbox storage quota is 40M
- The default mailbox message quota is 5000

This quota root permits greater storage in the Archive folder (100 M) than in all the other folders and message types combined (60 M). Also, no message limit is set for the Archive folder; in this example, only storage limits matter for archiving. The message types have both storage and number-of-message quotas.

The message-type quotas apply to the sum of all messages of those types, whether they are stored in the Archive folder or in any other folder.

The default mailbox quotas apply to all messages that are not text or voice message types and are not stored in the Archive folder. That is, the message-type quotas and Archive quota are not counted as part of the default mailbox quotas.

For example, the default mailbox quota would apply to a message that arrives in the user's INBOX without a Content-Type header and message-type definition. When that message is archived, the Archive folder storage quota would apply.

Guidelines for Specifying Multiple Quota Values

The following guidelines apply when you assign multiple quota values for a user:

- Quotas do not overlap. For example, when there is a quota for a particular message type or folder, messages of that type or messages in that folder are not counted toward the default quota. Each message counts toward one and only one quota.
- The total quota for the whole user mailbox equals the sum of the values of all the quotas specified by default, type, and folder.
- Message-type quotas take precedence over folder quotas. For example, suppose one quota is specified for a user's memos folder and another quota is specified for voice messages. Now suppose the user stores eight voice messages in the memos folder. The eight messages are counted toward the voice-mail quota and excluded from the memos folder quota.

Sample IMAP Session Returning Quota Root Values

When you run the **getquotaroot** IMAP command, the resulting IMAP session displays all quota roots for the user's mailbox, as shown here:

```
1 getquotaroot INBOX
* QUOTAROOT INBOXuser/joe user/joe/#text user/joe/#voice
* QUOTA user/joe (STORAGE 12340 20480 MESSAGE 148 5000)
* QUOTA user/joe/#text (STORAGE 1966 10240 MESSAGE 92 2000)
* QUOTA user/joe/#voice (STORAGE 7050 10240 MESSAGE 24 200)

2 getquotaroot Archive
* QUOTAROOT user/joe/Archive user/joe/#text user/joe/#voice
* QUOTA user/joe/Archive (STORAGE 35424 102400)
* QUOTA user/joe/#text (STORAGE 1966 10240 MESSAGE 92 2000)
* QUOTA user/joe/#voice (STORAGE 7050 10240 MESSAGE 24 200)
```

Expiring Messages by Message Type

The expire and purge feature allows the customer to move messages from one folder to another, archive messages, and remove messages from the message store, according to criteria the customer defines in expire rules. These tasks are performed with the **imexpire** utility.

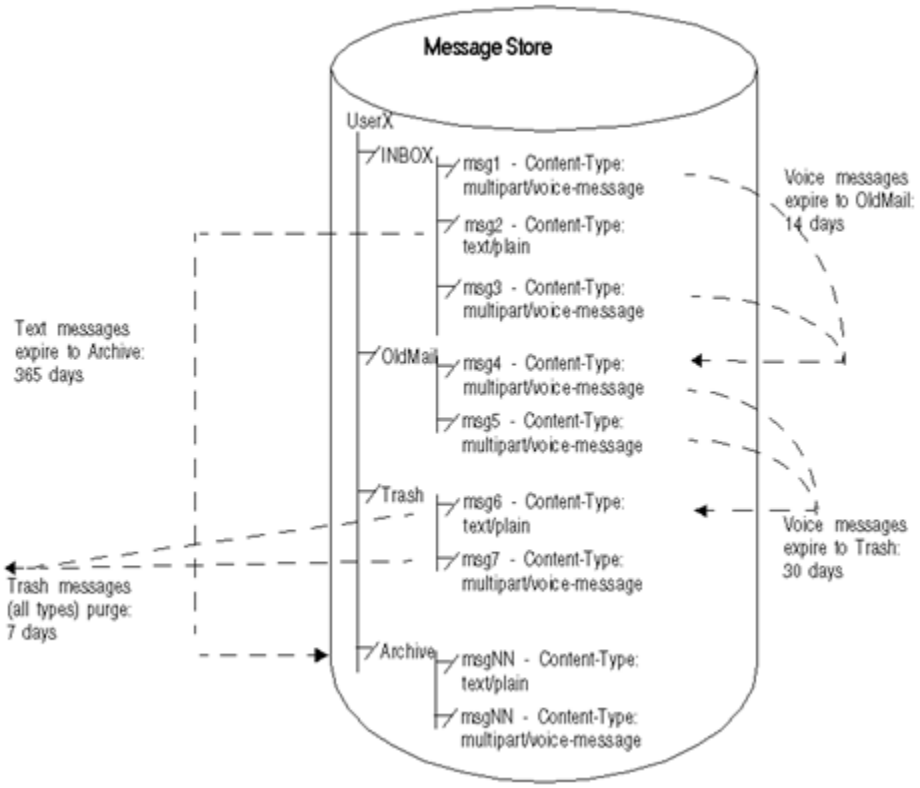
Because the **imexpire** utility is run by the administrator, it bypasses quota enforcement.

The customer can write expire rules so that messages of different types are expired according to different criteria.

The expire feature is extremely flexible, offering many choices for setting expire criteria. This section describes one example in which text and voice messages are expired according to different criteria.

Figure 33–9 illustrates an example of expiring messages by type.

Figure 33–9 Expiring Messages by Message Type



In this example, text messages and voice mail are expired in different ways, and they follow different schedules, as follows:

- Text messages are moved from a user's inbox to the user's Archive folder one year after they arrive in the message store.
- Voice mail is moved from the inbox to the OldMail folder after two weeks. If the user saves a voice message, the saved date is reset, and the message is moved two weeks after the new date.
- Voice mail is moved from the OldMail folder to the Trash folder after 30 days. The user also can save a voice message in the OldMail folder, which postpones the removal of the message for another 30 days after the new saved date.
- Messages of all types are discarded seven days after they are moved to the Trash folder.

The expire rules move voice mail to Trash automatically. Text messages are moved to Trash when a user deletes them.

Sample Rules for Expiring Different Message Types

You can implement the example described in this section by writing the following expire rules:

```
TextInbox.folderpattern: user/%/INBOX
TextInbox.messageheader.Content-Type: text/plain
TextInbox.messagedays: 365
TextInbox.action: fileinto:Archive

VoiceInbox.folderpattern: user/%/INBOX
```

```
VoiceInbox.messageheader.Content-Type: multipart/voice-message
VoiceInbox.savedays: 14
VoiceInbox.action: fileinto:OldMail

VoiceOldMail.folderpattern: user/%/OldMail
VoiceOldMail.messageheader.Content-Type: multipart/voice-message
VoiceOldMail.savedays: 30
VoiceOldMail.action: fileinto:Trash

Trash.folderpattern: user/%/Trash
Trash.savedays: 7
Trash.action: discard
```

Delivering Notifications for Message Types

Messaging Server, together with Message Queue, can produce notifications that deliver status information about messages of different types, such as voice mail, text messages, fax data, and image data.

For example, suppose a new phone message arrives in a user's mailbox, as described in "Message Deposit".

You can configure Messaging Server to generate a new-message notification for the Message Queue service. You can *also* configure Messaging Server to identify particular message types, including voice mail.

Now, when the voicemail is deposited in the user's mailbox, the message store triggers a notification that says, essentially:

- "This user has a new message."
- "The new message is voicemail."

After Messaging Server delivers the new-message notification to the Message Queue service, Message Queue sends it to a consumer (client interface), which filters and delivers the message to its destination.

You can write your Message Queue client program to interpret notification messages by message type and deliver status information about each type. In this example, the client program, recognizing the new message is voice mail, could trigger the Message Wait Indicator to turn on the "New Messages" light on the end-user's phone.

Now suppose new messages of different types--say, voice messages and email--arrive in the user's mailbox.

Once you have configured message types, a new-message notification carries data that counts the number of each type--in this case, the number of new voice messages and new email (text) messages. Your Message Queue client program can deliver the count by message type to the Media Application Server, which would notify the user that there are, for example, seven new voice mail messages and four new text messages in the user's cell phone inbox.

Notifications for Particular Message States

Table 33–1 shows the notifications that can be generated when a message changes state. For example, when a user reads a message or listens to voicemail, a **ReadMsg** notification can be generated. These notifications can carry information that tracks particular message types.

Table 33–1 Notification Message Descriptions

Notification Message	Description
NewMsg	New message was received by the system into the user's mailbox. Can contain message headers and body.
UpdateMsg	Message was appended to the mailbox by an IMAP operation. For example, the user copied an email message to the mailbox. Can contain message headers and body.
ReadMsg	Message in the mailbox was read. (In the IMAP protocol, the message was marked **Seen**.)
TrashMsg	Message was marked for deletion by IMAP or HTTP. The user may still see the message in the folder, depending on the mail client's configuration. The messages are to be removed from the folder when an expunge is performed.
DeleteMsg	Messages marked as **Deleted** are removed from the mailbox. This is the equivalent to IMAP expunge
PurgeMsg	Message expunged (as a result of an expired date) from the mailbox by the server process imexpire. This is a server side expunge, whereas {{DeleteMsg} is a client side expunge. This is not a purge in the true sense of the word.
OverQuota	Operation failed because the user's mailbox exceeded one of the quotas (**diskquota**, **msgquota**). The MTA channel holds the message until the quota changes or the user's mailbox count goes below the quota. If the message expires while it is being held by the MTA, it will be expunged.
UnderQuota	Quota went back to normal from **OverQuota** state.

How Do You Implement Notifications for Message Types?

To implement notifications that count by message type, you must do these things:

- Configure the message store to identify message types
- Configure a Messaging Server component, the JMQ notification plug-in, to produce notifications that identify message types
- Write a Message Queue client that retrieves, filters, and delivers the notification
- Design/write your Unified Messaging system (for example, the Media Application Server) to receive the notification and deliver it to the end user or other destination

Configuration Details

So far, you've seen the changes in message state that can trigger notifications. But how do the notifications carry information about message types?

We need to explain a few configuration details to show how these components work together. The following sections discuss the first two items listed above--the Messaging Server components.

Configuring the Message Store to Recognize Message Types

You use the Messaging Server **msconfig** or **configutil** utility to configure two options that enable the message store to identify message types:

- **store.messagetype.enable** (same for both Unified Configuration and legacy configuration)
- **store.messagetype.mtindex:***x***.contenttype** (Unified Configuration)

or

store.messagetype.*x* (legacy configuration)

First, you set the **store.messagetype.enable** option to on (**-v 1**) to enable message types.

Next, you define one **store.messagetype.mtindex:***x***.contenttype**(Unified Configuration) or **store.messagetype.***x* (legacy configuration) option for each message type. For example, to identify four message types, you define four iterations of this option with four different values. You do these things:

- Set an integer value for variable *x*.
- Specify a text string that is the value of the message type used in your Unified Messaging system with the **Content-Type** header.

For example, to define a text message, you can enter:

```
configutil -o store.messagetype.1 -v text/plain
```

To define a voicemail type, you can enter:

```
configutil -o store.messagetype.2 -v multipart/voice-message
```

Now the message store can identify any message type with a **Content-Type** header value of **text/plain** or **voice-message**.

Also, the message store will identify **messagetype.1** with text messages and **messagetype.2** with voicemail. (You'll need to know this when you see how notification properties carry information about message types.)

Note: You can configure other **configutil** options to define IMAP flag names, quota root names, and even an alternate message header name (other than **Content-Type**).

Configuring the JMQ Notification Plug-In to Generate Messages

You use the **configutil** utility to define options that configure the Messaging Server JMQ notification plug-in. The plug-in can produces a notification whenever a message changes state.

For each state change that should trigger a notification, you define a **configutil** option. For example, to enable notifications for new messages, you enter:

```
configutil -o local.store.notifyplugin.jmqnotify.NewMsg.enable -v 1
```

where **jmqnotify** is the name of the plug-in and **-v 1** enables notifications for this message.

To generate notifications for messages that the end user has read, you define this option:

```
local.store.notifyplugin.jmqnotify.ReadMsg.enable
```

and so on.

Note: To fully configure a JMQ notification plug-in, you must define several other **configutil** options.

How the Message Store and JMQ Notification Plug-in Work Together

Once you configure the message store's message-type feature and the JMQ notification plug-in, *both* components recognize and carry information about message types.

We've already discussed how the message store can enforce quotas and expire messages by message type.

Now, when a message changes state, the message store can generate a notification *and* the JMQ notification plug-in can automatically recognize the message type as well as the changed state.

Notification Properties for Message Types

Every notification carries additional information defined in properties. Different properties are present for different messages. For example, a **NewMsg** notification indicates the IMAP **uid** of the new message.

If message types are configured, the following properties are carried with notifications. These properties deliver a count of the messages in a particular state for each message type you have defined:

- **numMsgs***nn*
- **numSeen***nn*
- **numDeleted***nn*
- **numSeenDeleted***nn*

Suppose a new-message (**NewMsg**) notification is generated.

The Messaging Server JMQ notification function counts the number of new messages currently in the mailbox, by message type. Instead of sending one count with the NewMsg notification, an array specifying the count for each message type is sent.

The message-specific count is carried in the **numMsgs***nn* property and delivered with the notification.

The message-type number (*nn*) identifies a particular type. For example, you can configure message type 2 (**store.messagetype.2**) to identify voice messages, message type 3 (**store.messagetype.3**) to identify text messages, and so on.

For **ReadMsg** and **TrashMsg** notifications, the number of messages seen (**numSeen***nn* and the number marked as deleted (**numDeleted***nn*) are also counted by message type.

Table 33–2 describes these properties.

Table 33–2 Notification Properties for Message Types

Property	Data Type	Description
NumMsgs*nn*	**MQInt32**	The total number of messages now in the mailbox, specified for each message type. If message types are configured, a **numMsgs***nn* property carries a count for each message type *nn*. The **numMsgs** property is always sent; it counts the total number of all messages in the mailbox, including all types. For example, if 20 messages are currently in the mailbox, 10 are of type 3, 7 are of type 16, and the rest are not of any recognized type, the following properties and counts are carried with the notification: numMsgs=20 numMsgs3=10 numMsgs16=7
NumSeen*nn*	**MQInt32**	The total number of messages in the mailbox marked as seen (read), specified for each message type. If message types are configured, a **numSeen***nn* property carries a count for each message type *nn*. The **numSeen** property is always sent; it counts the total number of all messages marked as seen, including all types. For example, if 20 messages are marked as seen, 10 are of type 3,7 are of type 16, and the rest are not of any recognized type, the following properties and counts are carried with the notification: numSeen=20 numSeen3=10 numSeen16=7
NumDeleted*nn*	**MQInt32**	The total number of messages in the mailbox marked as deleted, specified for each message type. If message types are configured, a **numDeleted***nn* property carries a count for each message type nn. The **numDeleted** property is always sent; it counts the total number of all messages marked as deleted, including all types. For example, if 20 messages are marked as deleted, 10 are of type 3, 7 are of type 16, and the rest are not of any recognized type, the following properties and counts are carried with the notification: numDeleted=20 numDeleted3=10 numDeleted16=7
NumSeen Deleted*nn*	**MQInt32**	The total number of messages in the mailbox marked as seen (read) and marked as deleted, specified for each message type. If message types are configured, a **numSeenDeleted***nn* property carries a count for each message type *nn*. The **numSeenDeleted** property is always sent; it counts the total number of all messages marked as seen and deleted, including all types. For example, if 20 messages are marked as seen and deleted, 10 are of type 3, 7 are of type 16, and the rest are not of any recognized type, the following properties and counts are carried with the notification: numSeenDeleted=20 numSeenDeleted3=10 numSeenDeleted16=7

Additional Unified Messaging Support Features

The following features might provide additional support for implementing Unified Messaging Solutions.

Set IMAP Flag Based on Header Value at Delivery

The IMAP flag setting is done through the **imap4flags** sieve extension. (Testing header values is, of course, a basic sieve capability.) This feature is fully specified in RFC 5232. The goal is to be able to represent compound message context states, such as **URGENT. VOICEMAIL, BROADCAST FAX** and so on via numeric IMAP Flags.

To apply these actions to multiple users, you probably want to use system, source channel, and destination channel, or perhaps domain sieve rather than user sieves.

Implementation Description: The LMTP and SMTP processes allow IMAP flags to be set upon delivery based on header value. Upon insertion of a message into the message store, a sieve rule in conformance with SIEVE-IMAPFLAGS- 05 will modify the IMAP flag to represent the context of the message with an integer value from 1 to 32. The JMS will publish a notification event consequent to this event to the message queue.

In deployment, a SIEVE rule will perform a logical AND operation resulting in an IMAP flag and **X-HEADER** numeric value which would represent a context via a particular compound operation (for example,17 to represent **URGENT** and **EMAIL** and so on). **EDITHEADER-09** is supported, which, with configuration changes, will allow this to be used during SMTP delivery to allow devices without the capability to construct an RFC3458 header to construct one via an **ADDHEADER** rule, which in turn will allow preprocessing of message type identification prior to arrival in the final mailbox.

Modifications to IMAP Commands to Provide Message Counts

Messaging Server provides quick message counts and message states to clients for improving response time.The IMAP STATUS and SEARCH commands have been modified to provide return value data representing counts for [RFC3458] message types.

IMAP Unauthenticate

The IMAP command **UNAUTHENTICATE** is supported. This will be advertised by the **UNAUTHENTICATE CAPABILITY** and response. When invoked, the command will return the connection to an unauthenticated state, where **AUTHENTICATE** can be invoked without creating another connection to enable its reuse. This provides more effective use of IMAP pooling techniques via reuse of Network Sockets. The requirements of the IMAP **AUTHENTICATE** mechanism to create a new connection is bypassed and thereby enables connection reuse.

Modify IMAP APPEND to bypass quotas

Administrative users with appropriate privileges shall be able to bypass quota enforcement when they append messages to mailboxes with the IMAP **APPEND** command. The configuration option is **local.imap.adminbypassquota** which, when enabled, will bypass quota enforcement. Messages will be added to quota usage. Messages will not be rejected when the mailbox has exceeded its quota. Over quota warnings will still be delivered.

SMTP Future Release

Mail clients can indicate a future time for a message to be released for delivery up to one calendar year in advance of the current date. This support for RFC4865 has been added. The maximum values specified in the RFC for the option HOLDFOR shall be supported (+999999999 seconds). Equivalent **HOLDUNTIL** timestamp values shall be supported. This support will be enabled by placing the **futurerelease** channel option on the source channel used for initial message submission. The keyword shall take a single integer argument: the maximum number of seconds a message can be held.

Care should be used when enabling future release since it allows messages to be in effect stored in the MTA's queues. Future release should only be used for channels handling initial message submission and authentication should be required.

34

Messaging Server Command-Line Reference

This chapter describes the Oracle Communications Messaging Server **configtoxml** command that you use to manage a Unified Configuration. You must be logged in either as **root** or **mailsrv** to run these commands. These commands are located by default in the *MessagingServer_home*/**bin** directory.

See "Overview of Messaging Server Unified Configuration" for a description of Unified Configuration.

configtoxml Command

The **configtoxml** command converts a legacy configuration to a Unified Configuration.

Syntax

```
configtoxml [options]
```

Options

Table 34–1 shows the options for the **configtoxml** command.

Table 34–1 Options for configtoxml Command

Option	Description
-32,-64	Installation is 32-bit (**-32**) or 64-bit (**-64**) Messaging Server. Default is 64-bit when the installation type cannot be inferred from the **SERVERROOT** environment variable or from the location of this script.
-f | **--force**	Ignores safety checks, enables running as non-root and permits overwriting of any pre-existing Unified Configuration files. **Caution**: Using this option may result in a non-functioning configuration. The **restricted.cnf** file must always be owned by **root**.
-h | **--help**	Shows this help.
-i *INSTANCE*	Inserts instance name in the generated configuration files. The default is **ims**.
-l *DIR* | --location *DIR*	Reads the legacy configuration files from the specified directory. The default to use is determined by the following: 1. The **SERVERROOT/config/** path, if the **SERVERROOT** environment variable is defined 2. The OS-specific default path

Table 34–1 (Cont.) Options for configtoxml Command

Option	Description
-n | **--noactive**	Does not generate an active configuration and does not move the legacy configuration files to the *ConfigRoot*/**legacy-config** directory. The generated Unified Configuration files have the names **config_.xml**, **xpass_.xml**, and **restricted_.cnf**, and are written to *ConfigRoot*. This option cannot be used in conjunction with the **--output** or **--undo** options.
-o *CONFIG-FILEPASSWORD -FILERESTRICTED-FILE* | **--output** *CONFIG-FILEPASSWORD -FILERESTRICTED-FILE*	Directs the Unified Configuration file output to the designated files. By default, the files **config.xml**, **xpass.xml**, and **resricted.cnf** are written to *ConfigRoot* or the **SERVERROOT/config/** directory. This option cannot be used in conjunction with the **--noactive** or **--undo** options.
-r *ROLE* | **--role** *ROLE*	Inserts the role name in the generated configuration files. The default is **ims**.
-y |**--yes**	Pre-answer any confirmation questions with a "yes" response so that this script can be run without user intervention.
-u |**--undo**	Removes any active Unified Configuration files and restores any legacy configuration files.

The key environment variable for this command is:

- **SERVERROOT** (The default is **/opt/sun/comms/messaging64/**.)

Example

The following example shows the **configtoxml** converting a legacy configuration to a Unified Configuration.

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

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

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

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

Notes on the configtoxml Command

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

- When generating an active Unified Configuration, the **configtoxml** command moves all the processed legacy configuration files to the **$***ConfigRoot***/legacy-config** directory. The **--undo** option removes the Unified Configuration and restores the legacy configuration files.
- The **--undo** option leaves the Unified Configuration **restricted.cnf** password file in place.

Part II

Improving Performance

Part II of *Messaging Server System Administrator's Guide* describes how to improve Oracle Communications Messaging Server performance.

Part II contains the following chapters:

- Messaging Server Tuning and Best Practices
- Tuning the mboxlist Database Cache in Unified Configuration
- Best Practices for Messaging Server and ZFS

35

Messaging Server Tuning and Best Practices

This chapter describes tuning and best practices for Oracle Communications Messaging Server.

Log Files Tips

Use the following tips to configure and manage log files:

- Keep historical logging data.
- Configure Messaging Server log file rotation.

  Messaging Server provides logging facilities for the Messaging Server MTA, the Message Store, and services. The logging facilities provide you with time-stamped and labeled information about your system's messaging services. Using log files, you can gather message statistics, perform trend determination, troubleshoot problems, and so forth.

  While the system performs automatic rollovers to maintain the current log file, you must determine and manage log file rotation aspects such as how large a single log file may be, how large cumulative log files may be, how many log files to retain, and so forth.

  This section focuses on configuring log file rotation for the Messaging Server service logs, such as the IMAP service.

  For more information on managing Messaging Server log files, see the discussion on managing logging in *Messaging Server System Administrator's Guide*.

  Log File Options (Unified Configuration)

  The following **msconfig** options pertain to log file rollover:

  – *service*.**logfile.maxlogfilesize**

  – *.service*.**logfile.maxlogsize**

  – *.service*.**logfile.rollovertime**

  – *.service*.**logfile.maxlogfiles**

    where *service* is admin, pop, imap, imta, or http.

  – **maxlogfilesize** sets the largest size for a given logfile. (The limit is 2 Gbytes, or 2147483648 bytes.)

  – **maxlogsize** sets the maximum value for the sum of the log file sizes.

  – **maxlogfiles** sets the number of svc.seqNum.timestamp files to keep.

  – **rollovertime** sets the interval or age of a file before its get rotated.

You use the **msconfig** command to set these options. For example, the following command sets the maximum size of an IMAP service log file to 1 Gbyte:

```
msconfig set imap.logfile.maxlogfilesize 1073741824
```

Log File Options (legacy configuration)

The following **configutil** options pertain to log file rollover:

- **logfile**.*service*.**maxlogfilesize**
- **logfile**.*service*.**maxlogsize**
- **logfile**.*service*.**rollovertime**
- **logfile**.*service*.**maxlogfiles**

  where *service* is admin, pop, imap, imta, or http.

- **maxlogfilesize** sets the largest size for a given logfile. (The limit is 2 Gbytes, or 2147483648 bytes.)
- **maxlogsize** sets the maximum value for the sum of the log file sizes.
- **maxlogfiles** sets the number of svc.seqNum.timestamp files to keep.
- **rollovertime** sets the interval or age of a file before its get rotated.

  You use the **configutil** command to set these options. For example, the following command sets the maximum size of an IMAP service log file to 1 Gbyte:

```
configutil -o logfile.imap.maxlogfilesize -v 1073741824
```

Log File Examples

If the conditions at your site meet any one of the above controls, then the system rolls over the log file. Therefore, based on your site's traffic usage, you can set three of these **msconfig** options (or **configutil** options in legacy configuration) to be very large and unreachable while you set the fourth parameter to a value that forces the log to rollover in a manner that you want.

For example, most sites want to retain logs covering specific time periods or time spans, say one week. In order to keep a week's worth of data, set the **maxlogfiles** option equal to *N* times the frequency with which you are performing the log file rotations.

For instance, with the 1 Gbyte limit, you could rotate the files each hour (the rollover time is in seconds, so this is 3600 seconds) and just keep 168 copies of those individual files. Your option settings would then look like the following:

- **maxlogfilesize=1073741824**
- **maxlogsize**=*VERY_BIG_NUMBER*
- **rollovertime=3600**
- **maxlogfiles=168**

Such settings give you 168 hours (7 days) worth of service log files which are at most 1 Gbyte in size. For *VERY_BIG_NUMBER*, start with 180388626432 (168 x 1073741824).

Use your own custom settings if you prefer smaller log files, quicker log file rollover, and so forth. For example, you will get 336 log files if you set the rollover time to 1800 seconds, or each 30 minutes.

Note: Setting the following options to zero (0) (or less than zero) results in the default value being used:

maxlogfiles (default is 10)

maxlogsize (default is 20971520)

LMTP Tips

Configure LMTP between front-end MTA and back-end store. For more information, see "LMTP Delivery".

Message Store Tips

On classic message store, move **mboxlist** to separate ZFS or UFS file system with own LUN's. Instructions for Messaging Server on Oracle Solaris:

1. Stop Messaging Server and ensure **stored** process is stopped.
2. Rename **mboxlist** directory to **mboxlist.backup**.

   ```
   cd MessagingServer_home/data/store/
   mv mboxlist mboxlist.backup
   ```

3. Copy **mboxlist** database to *new_location*.

   ```
   cp -Rp mboxlist.backup/* new_location
   ```

4. Ensure the directory permissions for *new location* are set to the Messaging Server user.

   ```
   chown mailsrv:mail new_location
   ```

5. Create symlink to *new_location*.

   ```
   ln -s new_location mboxlist
   ```

6. Start Messaging Server.

MTA Tips

Use the following tips for the MTA:

- Put MTA out "front:"

  For more information, see the discussion about Understanding the Two-tiered Messaging Architecture in *Messaging Server Installation and Configuration Guide*.

- Separate ZFS file system for email:

  https://blogs.oracle.com/factotum/entry/messaging_server_and_mailstore_best

- Use **imslog.pl** to review MTA traffic and tune accordingly.
  - **imslog.pl** is available in the *MessagingServer_home***/examples/unsupported/** directory
- MTA RBL lookup tuning in the discussion about performance tuning real-time blackhole list (RBL) lookups in *Messaging Server Installation and Configuration Guide*.

Performance Tuning Tips

Use the following performance tuning tips:

- Read the discussion about performance tuning considerations for a Messaging Server architecture in *Messaging Server Installation and Configuration Guide*.
- Reduce spam load.
- Cache tuning:
  - Do not set excessive store **mboxlist** cache size.
  - The default 16 MB cache is sufficient for most medium sized environments.
  - Larger environments should review the following documentation to determine the best cache size:

    For more information, see the discussion on **store.dbtmpdir**, **base.lockdir** (Unified Configuration) or **local.lockdir** (legacy configuration), and **store.dbcachesize** in the section about performance tuning considerations for a Messaging Server architecture in *Messaging Server Installation and Configuration Guide*.
- Number of Messaging Server processes:
  - Set **imap.numprocesses**(Unified Configuration) or **service.imap.numprocesses** (legacy configuration) to (the number of cores)
  - Increase **imap.maxsessions** (Unified Configuration) or **service.imap.maxsessions** (legacy configuration) as needed.
  - Set **http.numprocesses** (Unified Configuration) or **service.http.numprocesses** (legacy configuration) to 1. Increase **http.numprocesses** (Unified Configuration) or **service.http.numprocesses** (legacy configuration) as needed.
  - Monitor the number of threads (LWPs). If during peak load times the number of LWPs in the **mshttpd** or **imapd** processes is constantly over 200, you might need to increase **numprocesses**.
- In a Convergence deployment, the webmail/mshttpd process should run on the same system as the Convergence instance:
  - Reduces network delays
  - Simplifies trouble-shooting
- Linux: Change the **IMTA_TMP** option in the MTA tailor file (**imta_tailor**) to use a **tmpfs** such as **/dev/shm**. The default value is **/tmp**, which is a **tmpfs** on Solaris but on Linux is a disk file system.

36

Tuning the mboxlist Database Cache in Unified Configuration

This chapter describes how to tune the Oracle Communications Messaging Server mboxlist database cache in Unified Configuration.

Setting the Mailbox Database Cache Size

Messaging Server makes frequent calls to the mailbox database. For this reason, it helps if this data is returned as quickly as possible. A portion of the mailbox database is cached to improve Message Store performance. Setting the optimal cache size can make a big difference in overall Message Store performance. You set the size of the cache with the **store.dbcachesize** option.

You should use the **store.dbtmpdir** option to redefine the location of the mailbox temporary files to a **tmpfs**, that is, **/tmp/msgDBtmpdir**. On Linux, the **tmpfs** location is usually **/dev/shm**. The default **store.dbtmpdir** value (**/tmp/.xxx**) is not appropriate, so you should change it manually to use the **tmpfs** location. On reboot, the **tmpfs** will be cleared, therefore be sure that permissions are stored so that the correct directory location can be recreated.

Note: The defaults for the **store.dbtmpdir** option and the **local.lockdir** option are now **/tmp/***encodedsubdirectory***/store** and **/tmp/***encodedsubdirectory***/lockm** respectively. The *encodedsubdirectory* is *.mailuserMSinstall-location*, where *mailuser* is the Message Server **mailsrv** user and *MSinstall-location* is the install location of Messaging with slashes ("/") replaced by underscores ("_").

For example, if the **mailsrv** user is **mailuser** and Messaging Server is installed in the **/opt/sun/comms/messaging64** directory, then **/tmp/***encodedsubdirectory* is **/tmp/.mailuser_opt_sun_comms_messaging64/**.

Because of this, you do not need to set these options on a Solaris platform.

To define the location of the mailbox temporary files:

1. Create the directory, for example:

```
mkdir /tmp/msgDBtmpdir
```

2. Assign the appropriate ownership and permissions so that this directory is owned, readable, and writable by the **mailsrv** user. For example, if the **mailsrv** user name is **mailuser** and the group name is **mail**:

```
chown mailuser:mail /tmp/msgDBtmpdir
chmod 700 /tmp/msgDBtmpdir
```

3. Set the **store.dbtmpdir** option, for example:

```
msconfig
msconfig> set store.dbtmpdir /tmp/msgDBtmpdir
msconfig# write
msconfig> exit
```

> **Note:** Be sure to set the **store.dbtmpdir** option to a uniquely named subdirectory of a **tmpfs** file system such as **/tmp**. In Oracle Solaris Cluster environments, or any situation where it could be possible for multiple instances of Messaging Server to be running on the same system, it is essential that you set this to a name that is unique to that instance. Make sure that no two Messaging Server instances can ever try to use the same **tmpdir** directory on the same system, for example, **/tmp/msg-***instance***-dbtmp** rather than just **/tmp/mboxlist**.

The files stored in the **store.dbtmpdir** location are temporarily memory mapped files used by all processes connecting to the database. Due to their usage pattern, the pages of these files will most likely be in memory all the time. So setting this to be on a **tmpfs** will not really increase memory usage. What it does is save I/O. When the Oracle Solaris virtual memory system sees a memory mapped file on a **tmpfs**, it knows it does not really need to write the modified pages back to the file. So there is only one copy in memory and it saves I/O.

The mailbox database is stored in data pages. When the various daemons make calls to the database (**stored**, **imapd**, **popd**), the system checks to see if the desired page is stored in the cache. If it is, the data is passed to the daemon. If not, the system reads the desired page and writes it in the cache. If there is no clean page available, the system must write one page from the cache back to disk.

Lowering the number of disk read/writes helps performance, so setting the cache to its optimal size is important:

- If the cache is too small, the desired data will have to be retrieved from disk more frequently than necessary.
- If the cache is too large, dynamic memory (RAM) is wasted, and it takes longer to synchronize the disk to the cache.

Of these two situations, a cache that is too small will degrade performance more than a cache that is too large.

Cache efficiency is measured by **hit rate**. Hit rate is the percentage of times that a database call can be handled by cache. An optimally sized cache will have a 98 to 99 percent hit rate (that is, 98 to 99 percent of the desired database pages will be returned to the daemon without having to grab pages from the disk). The goal is to set the smallest cache so that it holds several pages such that the cache will be able to return at least 98 to 99 percent of the requested data. If the direct cache return is less than 98 percent, then you must increase the cache size.

To Adjust the Mailbox Database Cache Size

Use the **msconfig** command to set the size of the cache with the **store.dbcachesize** option, for example:

```
msconfig
msconfig> set store.dbcachesize 25165825
msconfig# write
msconfig> exit
```

It is important to tune the cache size to smallest size that will accomplish the desired hit rate.

The **store.dbcachesize** option controls the size of a shared memory segment used by all processes connected to the database, including **stored**, **imap**, **popd**, **imsbackup**, **imsrestore**, **ims_master**, **tcp_lmtp_server**, and so on. While the maximum value for **store.dbcachesize** is 2 GB, setting it to the maximum wastes memory. Instead, start with the default value of 64 MB and monitor the cache hit rate over a period of days. Increase the value only if the hit rate is under 98%.

Also consider the transaction checkpoint function (performed by **stored**). Set the **store.checkpoint.debug** option and refresh **stored** to see log messages to provide more exact data about transaction checkpoint function time. For example:

```
msconfig
msconfig> set -restricted store.checkpoint.debug 1
msconfig# write
msconfig> exit
refresh store
Refreshing store server 7585 ... done
```

This process must examine all buffers in the cache and hold a region lock during the checkpoint. Other threads needing the lock must wait.

To Monitor the Mailbox Database Cache Size

1. Use the **imcheck** command to measure the cache hit rate.

   ```
   imcheck -s mpool > imcheck-s.out
   ```

   Note: The **imcheck -s** command is only valid for Berkeley Database message store.

2. Find the cache information section in the output file, for example:

   ```
   2MB 513KB 604B Total cache size.
   1 Number of caches.
   1       Maximum number of caches
   2MB 520KB Pool individual cache size.
   ```

   Then there will be several blocks of output - a summary and one for each database file - look for these lines in each block:

   ```
   0 Requested pages mapped into the process' address space.
   55339 Requested pages found in the cache (99%).
   ```

   In this case, the hit rate is 99 percent. This could be optimal or, more likely, it could be that the cache is too large. To test, lower the cache size until the hit rate moves

to below 99 percent. When you hit 98 percent, you have optimized the DB cache size. Conversely, if see a hit rate of less than 95 percent, then you should increase the cache size with the **store.dbcachesize** option.

As your user base changes, the hit rate can also change. Periodically check and adjust this option as necessary.

37

Best Practices for Messaging Server and ZFS

This chapter describes best practices for Oracle Communications Messaging Server and Oracle ZFS. ZFS provides the following features that make it ideal for backing up the Messaging Server message store:

- Snapshot backup
- Enables the use of less expensive SATA drives
- Built-in volume manager that enables you to grow file systems dynamically

Before You Begin

Read "Classic Messaging Server and Tiered Storage Overview" before using ZFS to back up the Messaging Server message store, which describes message store operation, its performance characteristics, and how to plan for and allocate store partitions.

Configuration Recommendations for ZFS and Messaging Server

The basic recommendations for configuring ZFS and Messaging Server are:

1. Separating the Messaging Server **mboxlist** database, message file, and index cache files on different file systems
2. Configuring the index cache record file system to use the recordsize of 4 Kbytes
3. Disabling file access time record
4. Keeping ZFS pool space under 80 percent utilization to maintain pool performance

The following information provides more context on these recommendations.

mboxlist Database, Message File and Index Cache Files Overview

The **mboxlist** database is a sleepycat database that contains mailbox meta data. The index cache records (**store.idx** and **store.c***) contain meta information about mailboxes and messages. Messaging Server accesses and modifies this meta information, although the modifications tend to be more random and smaller.

The location of the index cache records is controlled by setting the **partition:***partition_name*.**path** option, where *partition_name* is the name of the partition.

Each message file (***.msg**) represents a single email. Each message file is written to disk once, never modified, read many times (for example, when a user accesses the email, when the messages are backed up, and so on) and may be deleted. By default, these files are stored with the index and cache files.

The location of message files is controlled by setting the **partition:***partition_name***.messagepath** option, where *partition_name* is the name of the partition.

Separating the message files from the index cache records to different partitions (and underlying file systems) enables you to configure the file system with properties appropriate for the access type.

Index Cache Record File System

The index recordsize is 128 bytes. Cache recordsize is usually less than 2 Kbytes. The **mboxlist** database maximum page size is 8 Kbytes. The default ZFS recordsize is 128 Kbytes. Reducing the recordsize to 4 Kbytes for these file systems can improve performance and reduce incremental snapshot backup size.

Access Time Record

The message store does not utilize the file access time. By disabling file access time updates, you reduce unnecessary overhead.

ZFS Pool Space Utilization

ZFS pool performance can degrade when a pool is very full. As the pool approaches 100 percent full, more time is needed to find free space and it is more likely that the free space is available only in small chunks.

Configure the disk usage alarm threshold **alarm.system:diskavail.threshold** option to at least 20, to receive a warning before the disk becomes full. (The default value is 10). To enable message throttling sooner, configure the **store.diskusagethreshold** to 90.

To Configure ZFS and Messaging Server

The following steps implement the previously discussed recommendations.

1. Separate the Messaging Server **mboxlist** database, message file, and index cache files on different file systems. For example:

```
zfs create store/mboxlist
zfs set mountpoint=/var/opt/sun/comms/messaging/store/mboxlist store/mboxlist
zfs create store/primary-idx
msconfig set partition:primary.path /store/primary-idx
zfs create store/primary-msg
msconfig set partition:primary.messagepath /store/primary-msg
```

2. Configure the index cache record file system to use the recordsize of 4 Kbytes. For example:

```
zfs set recordsize=8k store/primary-idx
zfs set recordsize=128k store/primary-msg
zfs set recordsize=8k store/mboxlist
```

Note:

- This setting controls the recordsize of newly created files only.
- Do not set recordsize smaller than the system page size (8 Kbytes on SPARC and 4 Kbytes on Intel).
- Set recordsize=128k on the **-msg** file system even though that is the default so that it does not accidentally get overridden by a setting on a parent file system at a later time.

The default recordsize of 128k is appropriate for the message store message file system.

3. Disable file access time record. For example:

```
zfs set atime=off store/mboxlist
zfs set atime=off store/primary-idx
zfs set atime=off store/primary-msg
```

4. Configure the disk usage alarm threshold **alarm.system:diskavail.threshold** option to at least 20, to receive a warning before the disk becomes full. (The default value is 10). For example:

```
msconfig set alarm.system:diskavail.threshold 20
```

5. To enable message throttling sooner, configure the **store.diskusagethreshold** option to 90. (The default is 99). For example:

```
msconfig set store.diskusagethreshold 90
```

ZFS Administration Recommendations

- Perform snapshot backup regularly. Back up the **mboxlist** database, index, and message file systems atomically by using the **zfs snapshot -r** command. Then use the **zfs send** and **receive** commands, or an enterprise-level backup solution to save the data. For example:

```
zfs snapshot -r store@now
zfs send store/mboxlist@now | ssh host2 zfs recv store/mboxlist
zfs send store/primary-idx@now | ssh host2 zfs recv store/primary-idx
zfs send store/primary-msg@now | ssh host2 zfs recv store/primary-msg
```

- Perform incremental backups. You can use **zfs send -i** to perform incremental backups. Destroy the snapshots when they are not needed. For example:

```
zfs destroy -r store@now
```

Note: ZFS snapshots are for backing up and restoring the entire message store files system. You cannot back up and restore individual mailboxes. However, you can use **imsbackup** to back up the snapshot and **imsrestore** to restore the mailboxes.

Part III

Troubleshooting

Part III of *Messaging Server System Administrator's Guide* describes how to troubleshoot Oracle Communications Messaging Server components.

Part III contains the following chapters:

- Troubleshooting the MTA
- Troubleshooting the Message Store

38

Troubleshooting the MTA

This chapter describes common tools, methods, and procedures for troubleshooting the Message Transfer Agent (MTA) in Unified Configuration.

Also, see the discussion about monitoring procedures in "Monitoring Messaging Server".

> **Note:** This information assumes that you are familiar with the MTA, both from a conceptual and administration perspective.

Troubleshooting Overview

One of the first steps in troubleshooting the MTA is to determine where to begin the diagnosis. Depending on the problem, you might look for error messages in log files. In other situations, you might check all the standard MTA processes, review the MTA configuration, or start and stop individual channels. Whatever approach you use, consider the following questions when troubleshooting the MTA:

- Did configuration or environmental problems prevent messages from being accepted (for example, disk space or quota problems)?
- Were MTA services such as the Dispatcher and the Job Controller present at the time the message entered the message queue?
- Did network connectivity or routing problems cause messages to be stuck or misrouted on a remote system?
- Did the problem occur before or after a message entered into the message queue?

This information addresses these questions in the subsequent sections.

Standard MTA Troubleshooting Procedures

This section outlines standard troubleshooting procedures for the MTA. Follow these procedures if a problem does not generate an error message, if an error message does not provide enough diagnostic information, or if you want to perform general wellness checks, testing, and standard maintenance of the MTA.

Check the MTA Configuration

Test your address configuration by using the **imsimta test -rewrite** utility. With this utility, you can test the MTA's address rewriting and channel mapping without actually having to send a message. Refer to *Messaging Server Reference* for more information.

The utility will normally show address rewriting that will be applied as well as the channel to which messages will be queued. However, syntax errors in the MTA configuration will cause the utility to issue an error message. If the output is not what you expect, you may need to correct your configuration.

Check the Message Queue Directories

Check if messages are present in the MTA message queue directory, typically *DataRoot*/**queue/**. Use command-line utilities like **imsimta qm** to check for the presence of expected message files under the MTA message queue directory. See *Messaging Server Reference* for more information.

If the **imsimta test -rewrite** output looks correct, check that messages are actually being placed in the MTA message queue subdirectories. To do so, enable message logging and check the log files in the directory *DataRoot*/**log/**. See "Managing MTA Message and Connection Logs" for more information on MTA logging You can track a specific message by its message ID to ensure that it is being placed in the MTA message queue subdirectories. If you are unable to find the message, you may have a problem with file disk space or directory permissions.

Check the Ownership of Critical Files

You should have selected a mail server user account (**mailsrv** by default) when you installed Oracle Communications Messaging Server. The following directories, subdirectories, and files should be owned by this account:

```
DataRoot/queue/
DataRoot/log
DataRoot/tmp
```

Commands, like the ones in the following UNIX system example, can be used to check the protection and ownership of these directories:

```
ls -l -p -d /opt/sun/comms/messaging64/data/queue
drwx------ 2 mailsrv mail 512 Sep 18 21:17 /opt/sun/comms/messaging64/data/queue

ls -l -p -d /opt/sun/comms/messaging64/data/log
drwx------ 2 mailsrv mail 2560 Oct 15 05:25 /opt/sun/comms/messaging64/data/log

ls -l -p -d /opt/sun/comms/messaging64/data/tmp
drwx------ 2 mailsrv mail 512 Sep 18 21:17 /opt/sun/comms/messaging64/data/tmp
```

Check that the files in *DataRoot*/**queue** are owned by the MTA account by a using command such as the following UNIX system example:

```
ls -l -p -R /opt/sun/comms/messaging64/data/queue
```

Check that the Job Controller and Dispatcher Are Running

The MTA Job Controller handles the execution of the MTA processing jobs, including most outgoing (master) channel jobs.

Some MTA channels, such as the MTA's multi-threaded SMTP channels, include resident server processes that process incoming messages. These servers handle the slave (incoming) direction for the channel. The MTA Dispatcher handles the creation of such MTA servers. Dispatcher configuration options control the availability of the servers, the number of created servers, and how many connections each server can handle.

To check that the Job Controller and Dispatcher are present, and to see if there are MTA servers and processing jobs running, use the command **imsimta process**. Under idle conditions the command should result in **job_controller** and *dispatcher* processes. For example:

```
imsimta process
USER PID S VSZ RSS STIME TIME COMMAND
mailsrv2 308 S 50168 26896 23:05:00 00:01 /opt/sun/comms/messaging64/lib/tcp_smtp_
server
mailsrv2 4187 S 46880 17704 Feb_17 01:51 /opt/sun/comms/messaging64/lib/dispatcher
mailsrv2 5887 S 50160 26976 23:25:00 00:00 /opt/sun/comms/messaging64/lib/tcp_
smtp_server
mailsrv2 19018 S 47008 21640 Jun_20 01:21 /opt/sun/comms/messaging64/lib/job_
controller
```

If the Job Controller is not present, the files in the *DataRoot*/**queue** directory get backed up and messages are not delivered. If you do not have a Dispatcher, then you are unable to receive any SMTP connections.

See *Messaging Server Reference* for more information.

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

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

If neither the Job Controller nor the Dispatcher is present, you should review the **dispatcher.log-*** or **job_controller.log-*** file in the *DataRoot*/**log** directory.

If the log files do not exist or do not indicate an error, start the processes by using the **start-msg** command.

> **Note:** You should not see multiple instances of the Dispatcher or Job Controller when you run **imsimta process**, unless the system is in the process of forking (**fork()**) child processes before it executes (**exec())** the program that needs to run. However, the time frame during such duplication is very small.

Check the Log Files

If MTA processing jobs run properly but messages stay in the message queue directory, you can examine the log files to see what is happening. All MTA log files are created in the *DataRoot*/**log** directory. Log file name formats for various MTA processing jobs are shown in Table 38–1.

Table 38–1 MTA Log Files

File Name	Log File Contents
*channel*_ **master.log-***uniqueid*	Output of master program (usually client) for *channel*.
*channel*_**slave.log-***uniqueid*	Output of slave program (usually server) for *channel*.
dispatcher.log-*uniqueid*	Dispatcher debugging. This log is created regardless if the Dispatcher **DEBUG** option is set. However, to get detailed debugging information, you should set the **DEBUG** option to a non-zero value.

Table 38–1 (Cont.) MTA Log Files

File Name	Log File Contents
imta	**ims-ms** channel error messages when there is a problem in delivery.
job_ controller.log-*uniqueid*	Job controller logging. This log is created regardless if the Job Controller **DEBUG** option is set. However, to get detailed debugging information, you should set the **DEBUG** option to a non-zero value.
tcp_smtp_ server.log-*uniqueid*	Debugging for the **tcp_smtp_server**. The information in this log is specific to the server, not to messages.
return.log-*uniqueid*	Debug output for the periodic MTA message bouncer job; this log file is created if the **return_debug** MTA option is set.

Note: Each log file is created with a unique ID (**uniqueid**) to avoid overwriting an earlier log created by the same channel. To find a specific log file, you can use the **imsimta view** utility. You can also purge older log files by using the **imsimta purge** command. However, by default this command is run on a regular basis (see "Pre-defined Automatic Tasks"). See *Messaging Server Reference* for more information.

The *channel***master.log-***uniqueid* and *channel***slave.log-***uniqueid* log files are created in any of the following situations:

- There are errors in your current configuration.
- The **master_debug** or **slave_debug** options are set on the channel.
- If **mm_debug** is set to a non-zero value (**mm_debug > 0**).

For more information on debugging channel master and slave programs, see *Messaging Server Reference*.

Running a Channel Program Manually

When diagnosing an MTA delivery problem, it is helpful to manually run an MTA delivery job, particularly after you enable debugging for one or more channels.

The command **imsimta submit** notifies the MTA Job Controller to run the channel. If debugging is enabled for the channel in question, **imsimta submit** creates a log file in the *DataRoot***/log** directory as shown in Table 38–1, " MTA Log Files".

The command **imsimta run** performs outbound delivery for the channel under the currently active process, with output directed to your terminal. This might be more convenient than submitting a job, particularly if you suspect problems with job submission itself.

Note: To manually run channels, the Job Controller must be running.

See *Messaging Server Reference* for information on syntax, options, and examples of **imsimta submit** and **imsimta run** commands.

Starting and Stopping Individual Channels

In some cases, stopping and starting individual channels may make message queue problems easier to diagnose and debug. Stopping a message queue allows you to examine queued messages to determine the existence of loops or spam attacks.

To Stop Outbound Processing (dequeueing) for a Specific Channel

1. Use the **imsimta qm stop** command to stop a specific channel. Doing so prevents you from having to stop the Job Controller and having to recompile the configuration. In the following example, the **conversion** channel is stopped:

   ```
   imsimta qm stop conversion
   ```

2. To resume processing, use the **imsimta qm start** command to restart the channel. In the following example, the **conversion** channel is started:

   ```
   imsimta qm start conversion
   ```

   See *Messaging Server Reference* for more information on **imsimta qm start** and **imsimta qm stop** commands.

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 commands:

```
msconfig set job_controller.job_pool:DEFAULT.urgent_delivery
08:00-20:00
msconfig set job_controller.job_pool:DEFAULT.normal_delivery
08:00-20:00
msconfig set job_controller.job_pool:DEFAULT.nonurgent_delivery
08:00-20:00
```

To Stop Inbound Processing from a Specific Domain or IP Address (Enqueuing to a Channel)

You can run one of the following processes if you want to stop inbound message processing for a specific domain or IP address, while returning temporary SMTP errors to client hosts. By doing so, messages are not held on your system.

- To stop inbound processing for a specific host or domain name, add the following access rule to the ORIG_SEND_ACCESS mapping table by running the **msconfig edit mapping** command:

  ```
  ORIG_SEND_ACCESS

  *|*@example.com|*|* $X4.2.1|$NHost$ temporarily$ blocked
  ```

By using this process, the sender's remote MTA holds messages on their systems, continuing to resend them periodically until you restart inbound processing.

- To stop inbound processing for a specific IP address, add the following access rule to the PORT_ACCESS mapping table by running the **msconfig edit mapping** command:

```
PORT_ACCESS

TCP|*|25|_IP_address_to_block_|* $N400$ can't$ connect$ now
```

When you want to restart inbound processing from the domain or IP address, be sure to remove these rules from the mapping tables and recompile your configuration. In addition, you may want to create unique error messages for each mapping table. Doing so enables you to determine which mapping table is being used.

An MTA Troubleshooting Example

This section explains how to troubleshoot a particular MTA problem step-by-step. In this example, a mail recipient did not receive an attachment to an email message. Note: In keeping with MIME protocol terminology, the "attachment" is referred to as a "message part" in this section. The aforementioned troubleshooting techniques are used to identify where and why the message part disappeared (See "Standard MTA Troubleshooting Procedures"). By using the following steps, you can determine the path the message took through the MTA. In addition, you can determine if the message part disappeared before or after the message entered the message queue. To do so, you will need to manually stop and run channels, capturing the relevant files.

Note: The Job Controller must be running when you manually run messages through the channels.

Identify the Channels in the Message Path

By identifying which channels are in the message path, you can apply the **master_debug** and **slave_debug** options to the appropriate channels. These options generate debugging output in the channels' master and slave log files. In turn, the master and slave debugging information will assist in identifying the point where the message part disappeared.

1. Set the **log_message_id** MTA option to 1 by running the **msconfig set log_message_id 1** command. With this option set, you will see message ID: header lines in the **mail.log_current** file.
2. Run **imsimta cnbuild** to recompile the configuration.
3. Run **imsimta restartdispatcher** to restart the SMTP server.
4. Have the end user resend the message with the message part.
5. Determine the channels that the message passes through. While there are different approaches to identifying the channels, the following approach is recommended:
    1. On UNIX platforms, use the **grep** command to search for message ID: header lines in the **mail.log_current** file in directory *DataRoot***/log**.
    2. Once you find the message ID: header lines, look for the E (enqueue) and D (dequeue) records to determine the path of the message. Refer to "Understanding the MTA Log Entry Format" for more information on logging entry codes. See the following E and D records for this example:

```
29-Aug-2001 10:39:46.44 tcp_local conversion E 2 ...
29-Aug-2001 10:39:46.44 conversion tcp_intranet E 2 ...
29-Aug-2001 10:39:46.44 tcp_intranet D 2 ...
```

The channel on the left is the source channel, and the channel on the right is the destination channel. In this example, the E and D records indicate that the message's path went from the **tcp_local** channel to the **conversion** channel and finally to the **tcp_intranet** channel.

Manually Start and Stop Channels to Gather Data

This section describes how to manually start and stop channels (see also "Starting and Stopping Individual Channels"). By starting and stopping channels in the message's path, you are able to save the message and log files at different stages in the MTA process. These files are later used to "To Identify the Point of Message Breakdown".

To Manually Start and Stop Channels

1. Set the **mm_debug** MTA option to 5 by running the **msconfig set mm_debug 5** command to provide substantial debugging information.
2. Add the **slave_debug** and **master_debug** options to the appropriate channels by running the **msconfig edit channels** command and modifying the appropriate channel definitions.
   a. Use the **slave_debug** option on the inbound channel (or any channel where the message is switched to during the initial dialog) from the remote system that is sending the message with the message part. In this example, the **slave_debug** option is added to the **tcp_local** channel.
   b. Add the **master_debug** option to the other channels that the message passed through and were identified in "Identify the Channels in the Message Path". In this example, it would be added to the **conversion** and **tcp_intranet** channels.
   c. Recompile the configuration by using **imsimta cnbuild** if running a compiled configuration.
   d. Run the command **imsimta restart dispatcher** to restart the SMTP server.
3. Use the **imsimta qm stop** and **imsimta qm start** commands to manually start and stop specific channels. See "Starting and Stopping Individual Channels" for more on information by using these channel options.
4. To start the process of capturing the message files, have the end user resend the message with the message part.
5. When the message enters a channel, the message will stop in the channel if it has been stopped with the **imsimta qm stop** command. For more information, see Step 3 in this section.
   a. Copy and rename the message file before you manually run the next channel in the message's path. See the following UNIX platform example:

      cp ZZ01K7LXW76T7O9TD0TB.00 ZZ01K7LXW76T7O9TD0TB.KEEP1

      The message file typically resides in directory similar to *DataRoot***/queue/***destination_channel***/001**. The *destination_channel* is the next channel that the message passes through (such as: **tcp_intranet**). If you want to create subdirectories (like **001**, **002**, and so on) in the *destination_channel* directory, add the **subdirs** option to the channels.
   b. It is recommended that you number the extensions of the message each time you trap and copy the message to identify the order in which the message is processed.

6. Resume message processing in the channel and enqueue to the next destination channel in the message's path. To do so, use the **imsimta qm start** command.
7. Copy and save the corresponding channel log file (for example: **tcp_intranet_master.log-***) located in the *DataRoot*/**log** directory. Choose the appropriate log file that has the data for the message you are tracking. Make sure that the file you copy matches the timestamp and the subject header for the message as it comes into the channel. In the example of the **tcp_intranet_master.log-***, you might save the file as **tcp_intranet_master.keep** so the file is not deleted.
8. Repeat steps 5 - 7 until the message has reached its final destination.

   The log files you copied in Step 7 should correlate to the message files that you copied in Step 5. If, for example, you stopped all of the channels in the missing message part scenario, you would save the **conversion_master.log-*** and the **tcp_intranet_master.log-*** files. You would also save the source channel log file **tcp_local_slave.log-***.

   In addition, you would save a copy of the corresponding message file from each destination channel: **ZZ01K7LXW76T7O9TD0TB.KEEP1** from the **conversion** channel and **ZZ01K7LXW76T7O9TD0TB.KEEP2** from the **tcp_intranet** channel.
9. Remove debugging options once the message and log files have been copied.
   a. Remove the **slave_debug** and the **master_debug** options from the appropriate channels by running the **msconfig channels** command.
   b. Reset the **mm_debug** MTA option and remove the setting for the **log_message_id** MTA option by running the **msconfig set mm_debug 0** and **msconfig set log_message_id 0** commands or by running the **msconfig unset mm_debug** and **msconfig unset log_message_id** commands.
   c. Recompile the configuration by using **imsimta cnbuild** if running a compiled configuration.
   d. Run the command **imsimta restart dispatcher** to restart the SMTP server.

To Identify the Point of Message Breakdown

1. By the time you have finished starting and stopping the channel programs, you should have the following files with which you can use to troubleshoot the problem:
   a. All copies of the message file (for example: **ZZ01K7LXW76T7O9TD0TB.KEEP1)** from each channel program
   b. A **tcp_local_slave.log-*** file
   c. A set of *channel*_**master.log-*** files for each destination channel
   d. A set of **mail.log_current** records that show the path of the message All files should have timestamps and message ID values that match the message ID: header lines in the **mail.log_current** records.

      Note that the exception is when messages are bounced back to the sender; these bounced messages will have a different message ID value than the original message.
2. Examine the **tcp_local_slave.log-*** file to determine if the message had the message part when it entered the message queue.

   Look at the SMTP dialog and data to see what was sent from the client machine.

If the message part did not appear in the **tcp_local_slave.log-*** file, then the problem occurred before the message entered the MTA. As a result, the message was enqueued without the message part. If this the case, the problem could have occurred on the sender's remote SMTP server or in the sender's client machine.

3. Investigate the copies of the message files to see where the message part was altered or missing.

   If any message file showed that the message part was altered or missing, examine the previous channel's log file. For example, you should look at the **conversion_master.log-*** file if the message part in the message entering the **tcp_intranet** channel was altered or missing.

4. Look at the final destination of the message.

   If the message part looks unaltered in the **tcp_local_slave.log**, the message files (for example: **ZZ01K7LXW76T7O9TD0TB.KEEP1**), and the *channel*_**master.log-*** files, then the MTA did not alter the message and the message part is disappearing at the next step in the path to its final destination. If the final destination is the **ims-ms** channel (the Message Store), then you might download the message from the server to a client machine to determine if the message part is being dropped during or after this transfer.

   If the destination channel is a **tcp_*** channel, then you must go to the MTA in the message's path. Assuming it is an Messaging Server MTA, you will need to repeat the entire troubleshooting process (see "Identify the Channels in the Message Path" and "Manually Start and Stop Channels to Gather Data" and this section). If the other MTA is not under your administration, then the user who reported the problem should contact that particular site.

Common MTA Problems and Solutions

This section lists common problems and solutions for MTA configuration and operation.

TLS Problems

If, during SMTP dialog, the **STARTTLS** command returns the following error:

454 4.7.1 TLS library initialization failure

and if you have certificates installed and working for POP and IMAP access, check the following:

- Protections/ownerships of the certificates have to be set so **mailsrv** account can access the files.
- The directory where the certificates are stored need to have protections/ownerships set such that the **mailsrv** account can access the files within that directory.

After changing protections and installing certificates, you must run:

```
stop-msg dispatcher
start-msg dispatcher
```

Restarting should work, but it is better to shut it down completely, install the certificates, and then start things back up.

Changes to Configuration Files or MTA Databases Do Not Take Effect

If changes to your configuration are not taking effect, check to see if you have performed the following steps:

1. Recompile the configuration (by running **imsimta cnbuild**).
2. Restart the appropriate processes (like **imsimta restart dispatcher**).
3. Re-establish any client connections.

The MTA Sends Outgoing Mail but Does Not Receive Incoming Mail

Most MTA channels depend upon a slave or channel program to receive incoming messages. For some transport protocols that are supported by the MTA (like TCP/IP and UUCP), you must make sure that the transport protocol activates the MTA slave program rather than its standard server. Replacing the native **sendmail** SMTP server with the MTA SMTP server is performed as a part of the Messaging Server installation.

For the multi-threaded SMTP server, the startup of the SMTP server is controlled by the Dispatcher. If the Dispatcher is configured to use a **MIN_PROCS** value greater than or equal to one for the SMTP service, then there should always be at least one SMTP server process running (and potentially more, according to the **MAX_PROCS** value for the SMTP service). The **imsimta process** command may be used to check for the presence of SMTP server processes. See *Messaging Server Reference* for more information.

Dispatcher (SMTP Server) Won't Start Up

If the dispatcher won't start up, first check the **dispatcher.log-*** for relevant error messages. If the log indicates problems creating or accessing the **/tmp/.***path***.dispatcher.socket** file, then verify that the **/tmp** protections are set to 1777. This would show up in the permissions as follows:

```
drwxrwxrwt 8 root sys 734 Sep 17 12:14 tmp/
.
```

Also do an **ls -l** of the *path.version-specific-name***.dispatcher.socket** file and confirm the proper ownership. For example, if this is created by **root**, then it is inaccessible by **mailsrv**.

Do not remove the *.path***.dispatcher.file** and do not create it if it's missing. The dispatcher will create the file. If protections are not set to 1777, the dispatcher will not start or restart because it won't be able to create/access the socket file. In addition, there may be other problems occurring not related to the Messaging Server.

Messaging Server: *MessagingServer_home***/config/.var_opt_sun_comms_messaging64_config.dispatcher.socket**

Timeouts on Incoming SMTP Connections

Timeouts on incoming SMTP connections are most often related to system resources and their allocation. The following techniques can be used to identify the causes of timeouts on incoming SMTP connections.

To Identify the Causes of Timeouts on Incoming SMTP Connections

1. Check how many simultaneous incoming SMTP connections you allow. This is controlled by the **MAX_PROCS** and **MAX_CONNS** Dispatcher settings for the SMTP service. The number of simultaneous connections allowed is **MAX_**

PROCS*MAX_CONNS. If you can afford the system resources, consider raising this number if it is too low for your usage.

2. Another technique you can use is to open a TELNET session. In the following example, the user connects to **127.0.0.1** port 25. Once connected, 220 banner is returned. For example:

```
telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 budgie.example.com --Server ESMTP (Sun Java System Messaging Server 6.1
(built May 7 2001))
```

   If you are connected and receive a 220 banner, but additional commands (like **ehlo** and **mail from**) do not illicit a response, then you should run **imsimta test -rewrite** to ensure that the configuration is correct.

3. If the response time of the 220 banner is slow, and if running the **pstack** command on the SMTP server shows the following **iii_res*** functions (these functions indicate that a name resolution lookup is being performed):

```
febe2c04 iii_res_send (fb7f4564, 28, fb7f4de0, 400, fb7f458c, fb7f4564) +
42c febdfdcc iii_res_query (0, fb7f4564, c, fb7f4de0, 400, 7f) + 254
```

   then it is likely that the host has to do reverse name resolution lookups, even on a common pair like **localhost/127.0.0.1**. To prevent such a performance slowdown, you should reorder your host's lookups in the **/etc/nsswitch.conf** file. Also, it is recommended to remove the **nis** keyword from the **hosts** line in the **/etc/nsswitch.conf** file. To do so, change the following line in the **/etc/nsswitch.conf** file from:

```
hosts: dns nis [NOTFOUND=return] files
```

   to:

```
hosts: files dns [NOTFOUND=return]
```

   Making this change in the **/etc/nsswitch.conf** file can improve performance as fewer SMTP servers have to handle messages instead of multiple SMTP servers having to perform unnecessary lookups.

4. You can also put the **slave_debug** option on the channels handling incoming SMTP over TCP/IP mail, usually **tcp_local** and **tcp_intranet**. After doing so, review the most recent *tcp_local_slave.log*-**uniqueid** files to identify any particular characteristics of the messages that time out. For example, if incoming messages with large numbers of recipients are timing out, consider using the **expandlimit** option on the channel. Remember that if your system is overloaded and overextended, timeouts will be difficult to avoid entirely.

Messages Are Not Dequeued

Errors encountered during TCP/IP delivery are often transient; the MTA will generally retain messages when problems are encountered and retry them periodically. It is normal on large networks to experience periodic outages on certain hosts while other host connections work fine. To verify the problem, examine the log files for errors relating to delivery attempts. You may see error messages such as, "Fatal error from smtp_open." Such errors are not uncommon and are usually associated with a transient network problem. To debug TCP/IP network problems, use utilities like PING, TRACEROUTE, and NSLOOKUP.

The following example shows the steps you might use to see why a message is sitting in the queue awaiting delivery to **xtel.co.uk**. To determine why the message is not being dequeued, you can recreate the steps the MTA uses to deliver SMTP mail on TCP/IP.

```
nslookup -query=mx example.com (Step 1)

Server: LOCALHOST
Address: 127.0.0.1

Non-authoritative answer:
example.com preference = 10, mail exchanger = mailhost.example.com (Step 2)

telnet mailhost.example.com 25 (Step 3)
Trying... [10.1.1.1]
telnet: Unable to connect to remote host: Connection refused
```

1. Use the NSLOOKUP utility to see what MX records, if any, exist for this host. If no MX records exist, then you should try connecting directly to the host. If MX records do exist, then you must connect to the designated MX relays. The MTA honors MX information preferentially, unless explicitly configured not to do so. For more information, see the discussion on TCP/IP nameserver and MX record support in *Messaging Server Reference*.
2. In this example, the DNS (Domain Name Service) returned the name of the designated MX relay for **xtel.co.uk**. This is the host to which the MTA will actually connect. If more than one MX relay is listed, the MTA will try each MX record in succession, with the lowest preference value tried first.
3. If you do have connectivity to the remote host, you should check if it is accepting inbound SMTP connections by using TELNET to the SMTP server port 25.

   Note: If you use TELNET without specifying the port, you will discover that the remote host accepts normal TELNET connections. This does not indicate that it accepts SMTP connections; many systems accept regular TELNET connections but refuse SMTP connections and vice versa. Consequently, you should always do your testing against the SMTP port.

   In the previous example, the remote host is refusing connections to the SMTP port. This is why the MTA fails to deliver the message. The connection may be refused due to a misconfiguration of the remote host or some sort of resource exhaustion on the remote host. In this case, nothing can be done to locally to resolve the problem. Typically, you should let the MTA continue to retry the message.

If you are running Messaging Server on a TCP/IP network that does not use DNS, you can skip the first two steps. Instead, you can use TELNET to directly access the host in question. Be careful to use the same host name that the MTA would use. Look at the relevant log file from the MTA's last attempt to determine the host name. If you are using host files, you should make sure that the host name information is correct. It is strongly recommended that you use DNS instead of host names.

If you test connectivity to a TCP/IP host and encounter no problems using interactive tests, it is quite likely that the problem has simply been resolved since the MTA last tried to deliver the message. You can re-run the **imsimta submit tcp_channel** on the appropriate channel to see if messages are being dequeued.

Creating a New Channel

In certain circumstances, a remote domain can break down and the volume of mail addressed to this server can be so great that the outgoing channel queue fills up with messages that cannot be delivered. The MTA tries to redeliver these messages periodically (the frequency and number of the retries is configurable using the **backoff** channel option) and under normal circumstances, no action is needed. However, if too many messages get stuck in the queue, other messages may not get delivered in a timely manner because all the channel jobs are working to process the backlog of messages that cannot be delivered.

In this situation, you can reroute these messages to a new channel running in its own job controller pool. This will avoid contention for processing and allow the other channels to deliver their messages. This procedure is described in the following procedure. Assume a domain called **example.org**.

To Create a New Channel

1. Create a new channel called **tcp_example-daemon** and add a new value for the **pool** option.

   Channels are created in by running the **msconfig edit channels** command. The channel should have the same channel options on your regular outgoing **tcp_*** channel. Typically, this is the **tcp_local** channel, which handles all outbound (internet) traffic. Since **example.org** is out on the Internet, this is the channel to emulate. The new channel might look something like this:

   ```
   tcp_example smtp nomx single_sys remotehost inner allowswitchchannel \
   dentnonenumeric subdirs 20 maxjobs 7 pool SMTP_example maytlsserver \
   maysaslserver saslswitchchannel tcp_auth missingrecipientpolicy 0 \
   tcp_example-daemon
   ```

   Note the new option-value pair pool **SMTP_example**. This specifies that messages to this channel will only use computer resources from the **SMTP_example** pool. There is a blank line before and after the new channel.

2. Add two rewrite rules by running the **msconfig edit rewrite** to direct email destined for **example.org** to the new channel.

   The new rewrite rules look like this:

   ```
   example.org $U%$D@tcp_example-daemon
   .example.org $U%$H$D@tcp_example-daemon
   ```

   These rewrite rules direct messages to **example.org** (including addresses like **host1.example.org** or **hostA.host1.example.org**) to the new channel whose official host name is **tcp_example-daemon**. The rewriting part of these rules, **$U%$D** and **$U%$H$D**, retain the original addresses of the messages. **$U** copies the user name from original address. % is the separator---the @ between the username and domain. **$H** copies the unmatched portion of host/domain specification at the left of dot in pattern. **$D** copies the portion of domain specification that matched.

3. Define a new job controller pool called **SMTP_example**.

   Run the **msconfig set job_controller.job_pool:SMTP_example.job_limit 10** command to create a new pool called **SMTP_example** with a job_limit of 10. You can verify the addition of the new pool by running the **msconfig show job_controller.job_pool** command which will show output similar to the following:

   ```
   msconfig show job_controller.job_pool
   role.job_controller.job_pool:DEFAULT.job_limit = 10
   role.job_controller.job_pool:DEFAULT.urgent_delivery = help
   ```

```
role.job_controller.job_pool:IMS_POOL.job_limit = 2
role.job_controller.job_pool:SMTP_POOL.job_limit = 10
role.job_controller.job_pool:SMTP_example.job_limit = 10
```

This creates a message resource pool called **SMTP_example** that allows up to 10 jobs to be simultaneously run. See "The Job Controller" for details on jobs and pools.

4. Restart the MTA.

   Issue the commands: **imsimta cnbuild; imsimta restart**

   This recompiles the configuration and restarts the job controller and dispatcher.

   In this example, a large quantity of email from your internal users is destined for a particular remote site called **example.org**. For some reason, **example.org**, is temporarily unable to accept incoming SMTP connections and thus cannot deliver email. (This type of situation is not a rare occurrence.)

   As email destined for **example.org** comes in, the outgoing channel queue, typically **tcp_local**, will fill up with messages that cannot be delivered. The MTA tries to redeliver these messages periodically (the frequency and number of the retries is configurable using the **backoff** options) and under normal circumstances, no action is needed.

   However, if too many messages get stuck in the queue, other messages may not get delivered in a timely manner because all the channel jobs are working to process the backlog of **example.org** messages. In this situation, you may want reroute **example.org** messages to a new channel running in its own job controller pool (see "The Job Controller"). This will allow the other channels to deliver their messages without having to contend for processing resources used by **example.org** messages. Creating a new channel to address this situation is described in the following information.

MTA Messages Are Not Delivered

In addition to message transport problems, there are two common problems which can result in unprocessed messages in the message queues:

1. The queue cache is not synchronized with the messages in the queue directories. Message files in the MTA queue subdirectories that are awaiting delivery are entered into an in-memory queue cache. When channel programs run, they consult this queue cache to determine which messages to deliver in their queues. There are circumstances where there are message files in the queue, but there is no corresponding queue cache entry.

   a. To check if a particular file is in the queue cache, you can use the **imsimta cache -view** utility. If the file is not in the queue cache, then the queue cache needs to be synchronized.

      The queue cache is normally synchronized every four hours. If required, you can manually resynchronize the cache by using the command **imsimta cache -sync.** Once synchronized, the channel programs will process the originally unprocessed messages after new messages are processed. If you want to change the default (4 hours), you should modify the job_controller configuration by running the **msconfig set job_controller.synch_time***timeperiod* command where *timeperiod* reflects how often the queue cache is synchronized. The *timeperiod* must be greater than 30 minutes. In the following example, the queue cache synchronization is modified to 2 hours by running the following command:

```
msconfig set job_controller.synch_time 02:00
```

You can run **imsimta submit***channel* to clear out the backlog of messages after running **imsimta cache -sync**. Clearing out the channel may take a long time if the backlog of messages is large (greater than 1000).

For summarized queue cache information, run **imsimta qm -maint dir -database -total**.

b. If after synchronizing the queue cache, messages are still not being delivered, you should restart the Job Controller. To do so, use the **imsimta restart job_controller** command.

Restarting the Job Controller causes the message data structure to be rebuilt from the message queues on disk.

Caution: Restarting the Job Controller is a drastic step and should only be performed after all other avenues have been thoroughly exhausted.

Refer to "The Job Controller" for more information on the Job Controller.

2. Channel processing programs fail to run because they cannot create their processing log file. Check the access permissions, disk space and quotas.

Messages are Looping

If the MTA detects that a message is looping, that message will be sidelined as a HELD file. See "Diagnosing and Cleaning up .HELD Messages" for more information. Certain cases can lead to message loops which the MTA can not detect.

The first step is to determine why the messages are looping. You should look at a copy of the problem message file while it is in the MTA queue area, MTA mail log entries (if you have the **logging** channel option enabled in your MTA configuration for the channels in question) relating to the problem message, and MTA channel debug log files for the channels in question. Determining the From: and To: addresses for the problem message, seeing the Received: header lines, and seeing the message structure (type of encapsulation of the message contents), can all help pinpoint which sort of message loop case you are encountering.

Some of the more common cases include:

1. A postmaster address is broken. The MTA requires that the postmaster address be a functioning address that can receive email. If a message to the postmaster is looping, check that your configuration has a proper postmaster address pointing to an account that can receive messages.
2. Stripping of Received: header lines is preventing the MTA from detecting the message loop. Normal detection of message loops is based on Received: header lines. If Received: header lines are being stripped (either explicitly on the MTA system itself, or on another system like a firewall), it can interfere with proper detection of message loops. In these scenarios, check that no undesired stripping of Received: header lines is occurring. Also, check for the underlying reason why the messages are looping. Possible reasons include: a problem in the assignment of system names or a system not configured to recognize a variant of its own name, a DNS problem, a lack of authoritative addressing information on the system in question, or a user address forwarding error.

3. Incorrect handling of notification messages by other messaging systems are generating reencapsulated messages in response to notification messages. Internet standards require that notification messages (reports of messages being delivered, or messages bouncing) have an empty envelope From: address to prevent message loops. However, some messaging systems do not correctly handle such notification messages. When forwarding or bouncing notification messages, these messaging systems may insert a new envelope From: address. This can then lead to message loops. The solution is to fix the messaging system that is incorrectly handling the notification messages.

Diagnosing and Cleaning up .HELD Messages

If the MTA detects a serious problem having to do with delivery of a message, the message is stored in a file with the suffix **.HELD** in *DataRoot*/**queue/***channel*. For example:

```
ls
ZZ0HXZ00G0EBRBCP.HELD
ZZ0HY200C0O6LGHU.HELD
ZZ0HYA006LP66O3H.HELD
ZZ0HZ7003EOQSE37.HELD
```

.HELD files can occur due to three major reasons:

- Looping messages. The MTA detected that the messages were looping via build-up of one or another sort of **Received:** header lines).
- User or domain status set to **hold**. These are messages that are, by intent of the MTA administrator, intentionally being side-lined, typically while some maintenance procedure is being performed, (for example, while moving user mailboxes).
- Suspicious messages. Messages that met some suspicious threshold and were held for later manual inspection by the MTA administrator. Messages can be **.HELD** due to exceeding a configured maximum number of envelope recipients (see the **holdlimit** channel option in *Messaging Server Reference*), due to running the **imsmita qclean** or **clean** or **hold** commands based on some suspicion of the message(s) in question, or due to use of a **hold** action in a Sieve script.

Messages .HELD Due to Looping

Messages bouncing between servers or channels are said to be looping. Typically, a message loop occurs because each server or channel thinks the other is responsible for delivery of the message. Looping messages usually have a great many ***Received:** header lines. The **Received:** header lines illustrate the exact path of the message loop. Look carefully at the host names and any recipient address information (for example, **for recipient** clauses or **ORCPT recipient** comments) appearing in such header lines. One cause of such message loops is user error.

For example, an end users might set an option to forward messages on two separate mail hosts to one another. On their **example.com** account, the users enable mail forwarding to their **example.edu** account. And, forgetting that they have enabled this setting, they set mail forwarding on their **example.edu** account to their **example.com** account.

A loop can also occur with a faulty MTA configuration. For example, MTA Host X thinks that messages for **mail.example.com** go to Host Y. However, Host Y thinks that Host X should handle messages for **mail.example.com**. As a result, Host Y returns the mail to Host X.

In these cases, the message is ignored by the MTA and no further delivery is attempted. When such a problem occurs, look at the header lines in the message to determine which server or channel is bouncing the message. Fix the entry as needed.

Another common cause of message loops is the MTA receiving a message that was addressed to the MTA host using a network name that the MTA does not recognize (has not been configured to recognize) as one of its own names. The solution is to add the additional name to the list of names that your MTA recognizes as *its own*. The MTA's threshholds for determining that a message is looping are configurable; see the **MAX_*RECEIVED_LINES** MTA options (http://docs.oracle.com/cd/E19566-01/819-4429/index.html). Also note that the MTA may optionally be configured. See the **HELD_SNDOPR** MTA option to generate a syslog notice whenever a message is forced into **.HELD** state due to exceeding such a threshold. If syslog messages of **Received count exceeded; message held.** are present, then you know that this is occurring.

You can resend the **.HELD** message by using the **imsimta qm release** command **or** by following these steps:

> **Note:** The **imsimta qm release** command is the preferred method.

1. Rename the **.HELD** extension to any 2 digit number other than 00. For example, .HELD to .06.

   > **Note:** Before renaming the .HELD file, be sure that the message is not likely to continue looping.

2. Run *imsimta cache -sync*.

   Running this command updates the cache.

3. Run **imsimta submit***channel* or **imsimta run***channel*.

You might need to perform these steps multiple times, since the message may again be marked as **.HELD**, because the **Received:** header lines accumulate. If the problem still exists, the ***.HELD** file is recreated under the same channel with as before. If the problem has been addressed, the messages are dequeued and delivered.

See **clean** if you determine that the messages can simply be deleted with no attempt to deliver them,

Messages .HELD Due to User or Domain hold Status

Messages that are **.HELD** due to a user or domain status of **hold**, and only messages **.HELD** for such a reason, are normally stored in the hold channel's queue area. That is, **.HELD** message files in the hold channel's queue area can be assumed to be **.HELD** due to user or domain status.

Messages .HELD Due to a Suspicious Characteristic

Messages **.HELD** due to some suspicious characteristic exhibit that characteristic. The characteristic could be anything that the site has chosen to characterize as **suspicious**. MTA Administrators should stay aware of these configuration choices and actions. However, if you are not the only or original administrator of this MTA, then check the MTA configuration for any configured use of the **holdlimit** channel option (see the discussion on expansion of multiple addresses in *Messaging Server Reference*), any use of the **$H** flag in address-based ***_ACCESS** mapping tables, or any use of the **hold**

action in any system Sieve file, or any channel level Sieve filters configured and named by use of **sourcefilter** or **destinationfilter** channel options. See the discussion on the filter file location in *Messaging Server Reference*. Additionally, ask any fellow MTA administrators about any manual command-line message holds (through, for instance, an **imsimta qm clean** command) they might have recently performed. Application of a Sieve filter **hold** action, whether from a system Sieve filter or from users' personal Sieve filters, may optionally be logged. See the discussion on the LOG_FILTER global MTA option in *Messaging Server Reference* for more information.

Received Message is Encoded

Messages sent by the MTA are received in an encoded format. For example:

```
Date: Wed, 04 Jul 2001 11:59:56 -0700 (PDT)
From: "Desdemona Vilalobos" <Desdemona@example.com>
To: santosh@example.edu
Subject: test message with 8bit data
MIME-Version: 1.0
Content-type: TEXT/PLAIN; CHARSET=ISO-8859-1
Content-transfer-encoding: QUOTED-PRINTABLE

2=00So are the Bo=F6tes Void and the Coal Sack the same?=
```

These messages appear unencoded when read with the MTA decoder command **imsimta decode**. See *Messaging Server Reference* for more information.

The SMTP protocol only allows the transmission of ASCII characters (a seven-bit character set) as set forth by RFC 821. In fact, the unnegotiated transmission of eight-bit characters is illegal through SMTP, and it is known to cause a variety of problems with some SMTP servers. For example, SMTP servers can go into compute bound loops. Messages are sent over and over again. Eight-bit characters can crash SMTP servers. Finally, eight-bit character sets can wreak havoc with browsers and mailboxes that cannot handle eight-bit data.

An SMTP client used to only have three options when handling a message containing eight-bit data: return the message to the sender as undeliverable, encode the message, or send it in direct violation of RFC 821. But with the advent of MIME and the SMTP extensions, standard encodings exist that can encode eight-bit data by using the ASCII character set.

In the previous example, the recipient received an encoded message with a MIME content type of TEXT/PLAIN. The remote SMTP server (to which the MTA SMTP client transferred the message) did not support the transfer of eight-bit data. Because the original message contained eight-bit characters, the MTA had to encode the message.

Server-Side Rules (SSR) Are Not Working

A filter consists of one or more conditional actions to apply to a mail message. Since the filters are stored and evaluated on the server, they are often referred to as server-side rules (SSR).

Testing Your SSR Rules

- To check the MTA's user filters, run the following command:

  ```
  imsimta test -rewrite -debug -filter user@domain
  ```

  In the output, look for the following information:

```
mmc_open_url called to open ssrf: user@ims-ms
URL with quotes stripped: ssrd: user@ims-ms
Determined to be a SSRD URL.
Identifier: user@ims-ms-daemon
Filter successfully obtained.
```

- In addition, you can add the **slave_debug** option to the **tcp_local** channel to see how a filter is applied. The results are displayed in the **tcp_local_slave.log** file. Be sure to set **mm_debug** to 5 by running the **msconfig set mm_debug 5** command to get sufficient debugging information.

Common Syntax Problems

If there is a syntax problem with the filter, look for the following message in the **tcp_local_slave.log-*** file:

Error parsing filter expression:...

- If the filter is good, then filter information is at the end of the output.
- If the filter is bad, then the following error is at the end of the output:

  Address list error - 4.7.1 Filter syntax error: desdaemona@example.com

  Also, if the filter is bad, then the SMTP RCPT TO command returns a temporary error response code:

```
RCPT TO: <user>@<domain>
452 4.7.1 Filter syntax error
```

Slow Response After Users Press Send Email Button

If users are experiencing delays when they send messages, undersized message queue disks could be responsible for reduced disk input/output. When users press the SEND button on their email client, the MTA will not fully accept receipt of the message until the message has been committed to the message queue. See the discussion on disk sizing for MTA message queues in *Messaging Server Installation and Configuration Guide* for more information.

Abnormal Job Controller Terminations Seen in job_controller Logs

The Job Controller is essentially an in-memory database. Unlike other parts of the MTA, it doesn't have queues or transactions with which to contend. It listens for activity coming in on various network connections and updates its database accordingly.

Consequently, if the Job Controller fails, it is most likely a resource allocation failure (resource exhaustion). The only significant resource the Job Controller uses, especially when under stress, is memory. Therefore, allocate the right amount of memory for the machine that contains the Job Controller. See the discussion on planning a messaging server sizing strategy in *Messaging Server Installation and Configuration Guide* for details on memory utilization.

General Error Messages

When the MTA fails to start, general error messages appear at the command line. In this section, common general error messages will be described and diagnosed.

Note: To diagnose your own MTA configuration, use the **imsimta test -rewrite-debug** utility to examine your MTA's address rewriting and channel mapping process. This utility enables you to check the configuration without actually sending a message. See "Check the MTA Configuration" for more information.

MTA subcomponents might also issue other error messages that are described in the MTA command-line utilities and configuration information. See "Configuring POP, IMAP, and HTTP Services" and "MTA Configuration Overview" and *Messaging Server Reference* for more information.

Errors in mm_init

An error in **mm_init** generally indicates an MTA configuration problem. If you run the **imsimta test -rewrite** utility, these errors are displayed. Other utilities such as **imsimta cnbuild**, or a channel, a server, or a browser might also return such an error.

Commonly encountered mm_init errors include:

- bad equivalence for alias...
- cannot open alias include file...
- duplicate aliases found...
- duplicate host in channel table...
- duplicate mapping name found...
- mapping name is too long...
- error initializing ch_facility compiled character set version mismatch
- error initializing ch_facility no room in...
- local host alias or proper name too long for system...
- no equivalence addresses for alias...
- no official host name for channel...
- official host name is too long

bad equivalence for alias...

The right-hand side of an alias file entry is improperly formatted.

cannot open alias include file...

A file included into the alias file cannot be opened.

duplicate aliases found...

Two alias file entries have the same left hand side. You must find and eliminate the duplication. Look for an error message that says **error line #XXX** where **XXX** is a line number. You can fix the duplicated alias on the line.

duplicate host in channel table...

This error message indicates that you have two channel definitions in the MTA configuration that both have the same official host name.

Check your MTA configuration for any channel definitions with duplicate official host names.

duplicate mapping name found...

This message indicates that two mapping tables have the same name, and one of the duplicate mapping tables needs to be removed.

> **Note:** A blank line should precede and follow any line with a mapping table name. However, no blank lines should be interspersed among the entries of a mapping table.

mapping name is too long...

This error means that a mapping table name is too long and needs to be shortened.

error initializing ch_facility compiled character set version mismatch

If you see this message, you must recompile and reinstall your compiled character set tables through the command **imsimta chbuild.** See *Messaging Server Reference* for more information.

error initializing ch_facility no room in...

This error message generally means that you need to resize your MTA character set internal tables and then rebuild the compiled character set tables with the following commands:

```
imsimta chbuild -noimage -maximum -option
imsimta chbuild
```

Verify that nothing else needs to be recompiled or restarted before making this change. See *Messaging Server Reference* for more information.

local host alias or proper name too long for system...

This error indicates that a local host alias or proper name is too long (the optional right hand side in the second or subsequent names in a channel block).

no equivalence addresses for alias...

An entry in the alias file is missing a right hand side (translation value).

no official host name for channel...

This error indicates that a channel definition block is missing the required second line (the official host name line). See the MTA configuration and command-line utilities information in *Messaging Server Reference* for more information on channel definition blocks. A blank line is required before and after each channel definition block, but a blank line must not be present between the channel name and official host name lines of the channel definition.

official host name is too long

The official host name for a channel (second line of the channel definition block) is limited to 128 octets in length. If you are trying to use a longer official host name on a channel, shorten it to a place holder name, and then use a rewrite rule to match the longer name to the short official host name. You might see this scenario if you work with the *l* (local) channel host name. For example:

```
<Original l Channel:>
```

```
!delivery channel to local /var/mail store
l subdirs 20 viaaliasrequired maxjobs 7 pool LOCAL_POOL
walleroo.pocofronitas.thisnameismuchtoolongandreallymakesnosensebutitisan
example.monkey.gorilla.orangutan.antidiexampleblismentarianism.newt.salaman
der.lizard.gecko.komododragon.com

<Create Place Holder:>
!delivery channel to local /var/mail store
l subdirs 20 viaaliasrequired maxjobs 7 pool LOCAL_POOL
newt

<Create Rewrite Rule:>
newt.salamander.lizard.gecko.komododragon.com $U%$D@newt
```

When using the **l** (local) channel, you need to use a REVERSE mapping table. See the MTA configuration information in *Messaging Server Reference* for information on usage and syntax.

Compiled Configuration Version Mismatch

One of the functions of the **imsimta cnbuild** utility is to compile MTA configuration information into an image that can be quickly loaded. The compiled format is quite rigidly defined and often changes substantially between different versions of the MTA. Minor changes might occur as part of patch releases.

When such changes occur, an internal version field is also changed so that incompatible formats can be detected. The MTA components halt with the "Compiled Configuration Version Mismatch" error when an incompatible format is detected. The solution to this problem is to generate a new, compiled configuration with the command **imsimta cnbuild.**

Also, use the **imsimta restart** command to restart any resident MTA server processes, so they can obtain updated configuration information.

Swap Space Errors

To ensure proper operation, it is important to configure enough swap space on your messaging system. The amount of required swap space will vary depending on your configuration. A general tuning recommendation is that the amount of swap space should be at least three times the amount of main memory.

An error message such as the following indicates a lack of swap space:

```
jbc_channels: chan_execute [1]: fork failed: Not enough space
```

You might see this error in the Job Controller log file. Other swap space errors will vary depending on your configuration.

Use the following commands to determine how much swap space you have left and determine how much you have used:

swap -s (at the time MTA processes are busy), **ps -elf**, or **tail /var/adm/messages**

File Open or Create Errors

To send a message, the MTA reads configuration files and creates message files in the MTA message queue directories. Configuration files must be readable by the MTA or any program written against the MTA's SDKs. During installation, proper permissions are assigned to these files. The MTA utilities and procedures which create configuration files also assign permissions. If the files are protected by the system

manager, other privileged user, or through some site-specific procedure, the MTA may not be able to read configuration information. This results in "File open" errors or unpredictable behavior. The **imsimta test -rewrite** utility reports additional information when it encounters problems reading configuration files. See *Messaging Server Reference* for more information.

If the MTA appears to function from privileged accounts but not from unprivileged accounts, then file permissions in the MTA table directory are likely the cause of the problem. Check the permissions on configuration files and their directories. See "Check the Ownership of Critical Files" for more information.

"File create" errors usually indicate a problem while creating a message file in an MTA message queue directory. See "Check the Message Queue Directories" to diagnose file creation problems.

Illegal Host/Domain Errors

You might see this error when an address is provided to the MTA through a browser. Or, the error may be deferred and returned as part of an error return mail message. In both cases, this error message indicates that the MTA is not able to deliver mail to the specified host. To determine why the mail is not being sent to the specified host, follow these troubleshooting procedures:

- Verify that the address in question is not misspelled, is not transcribed incorrectly, or does not use the name of a host or domain that no longer exists.
- Run the address in question through the **imsimta test -rewrite** utility. If this utility also returns an "illegal host/domain" error on the address, then the MTA has no rewrite rules or other configurations to handle the address. Verify that you have configured MTA correctly, that you answered all configuration questions appropriately, and that you have kept your configuration information up to date.
- If **imsimta test -rewrite** does not encounter an error on the address, then MTA is able to determine how to handle the address, but the network transport will not accept it. You can examine the appropriate log files from the delivery attempt for additional details. Transient network routing or name service errors should not result in returned error messages, though it is possible for badly misconfigured domain name servers to cause these problems.
- If you are on the Internet, check that you have properly configured your TCP/IP channel to support MX record lookups. Many domain addresses are not directly accessible on the Internet and require that your mail system correctly resolve MX entries. If you are on the Internet and your TCP/IP is configured to support MX records, you should have configured the MTA to enable MX support. See the discussion on TCP/IP connection and DNS lookup support in *Messaging Server Reference* for more information. If your TCP/IP package is not configured to support MX record lookups, then you will not be able to reach MX-only domains.

Errors in SMTP channels, os_smtp_* errors

Errors such as the following are not necessarily MTA errors: os_smtp_* errors like os_smtp_open, os_smtp_read, and os_smtp_write errors. These errors are generated when the MTA reports a problem encountered at the network layer. For example, an os_smtp_open error means that the network connection to the remote side could not be opened. The MTA may be configured to connect to an invalid system because of addressing errors or channel configuration errors. The os_smtp_* errors are commonly due to DNS or network connectivity problems, particularly if this was a previously working channel or address. os_smtp_read or os_smtp_write errors are usually an

indication that the connection was aborted by the other side or due to network problems.

Network and DNS problems are often transient in nature. The occasional os_smtp_* error is usually nothing to be concerned about. However, if you are consistently seeing these errors, it could be an indication of an underlying network problem.

To obtain more information about a particular os_smtp_* error, enable debugging on the channel in question. Investigate the debug channel log file that will show details of the attempted SMTP dialogue. In particular, look at the timing of when a network problem occurred during the SMTP dialogue. The timing could suggest the type of network or remote side issue. In some cases, you might also want to perform network level debugging (for example, TCP/IP packet tracing) to determine what was sent or received.

39

Troubleshooting the Message Store

This chapter provides guidelines for troubleshooting the Oracle Communications Messaging Server message store as well as recovery procedures for when the message store becomes corrupted or unexpectedly shuts down.

See also:

- Using Message Store Log Messages
- Upgrading the Classic Message Store

Repairing Mailboxes and the Mailboxes Database (reconstruct Command)

If one or more mailboxes become corrupt, use the "reconstruct" utility to rebuild the mailboxes or the mailbox database. You can use this utility to recover from almost any form of data corruption in the mail store. See "Error Messages Signifying reconstruct Is Needed" and "reconstruct" for more details.

Reduced Message Store Performance

Message store problems can occur if the **mboxlist** database cache is too small. Specifically, Message store performance can slow to unacceptable levels and can even dump core. Refer to the discussion on performance tuning considerations in *Messaging Server Installation and Configuration Guide*.

Red Hat Linux - Messaging Server Patch 120230-08 IMAP, POP and HTTP Servers Not Starting Due to Over Sessions Per Process

After installing this patch, when you try to start Messaging Server, the IMAP, POP and HTTP servers do not start and may send the following example error logs:

```
http server - log:
[29/May/2006:17:44:37 +051800] usg197 httpd[6751]: General Critical: Not enough
file
descriptors to support 6000 sessions per process; Recommend ulimit -n 12851 or 87
sessions per process.
pop server - log:
[29/May/2006:17:44:37 +051800] usg197 popd[6749]: General Critical: Not enough
file
descriptors to support 600 sessions per process; Recommend ulimit -n 2651 or 58
sessions per process.
imap server - log:
[29/May/2006:17:44:37 +051800] usg197 imapd[6747]: General Critical: Not enough
file descriptors to support 4000 sessions per process; Recommend ulimit -n 12851
or 58 sessions per process.
```

Set the appropriate number of file descriptors for all three server sessions. Additional file descriptors are available by adding a line similar to the following to the **/etc/sysctl.conf** file and using **sysctl -p** to reread that file:

```
fs.file-max = 65536
```

You must also add a line like the following to the **/etc/security/limits.conf** file:

```
* soft nofile 65536
* hard nofile 65536
```

Convergence Not Loading Mail Page

If users accessing their mail with web clients, like Convergence, cannot load pages, the problem might be that the data is getting corrupted after compression. This can sometimes happen if the system has deployed a outdated proxy server. To solve this problem, try setting the **msconfighttp.gzipstatic** and **http.gzipdynamic** options to **0** to disable data compression. If this solves the problem, you may want to update the proxy server.

Command Using Wildcard Pattern Does Not Work

Some UNIX shells may require quotes around wildcard options and some will not. For example, the C shell tries to expand arguments containing wildcards (*, ?) as files and will fail if no match is found. These pattern matching arguments may need to be enclosed in quotes to be passed to commands like **mboxutil**.

For example:

```
mboxutil -l -p user/usr44*
```

works in the Bourne shell, but fails with tsch and the C shell. These shells would require the following:

```
mboxutil -l -p "user/usr44*"
```

If a command using a wildcard pattern does not work, verify whether you need to use quotes around wildcards for that shell.

Unknown/invalid Partition

A user can get the message "Unknown/invalid partition" in mshttpd if their mailbox was moved to a new partition that was just created and Messaging Server was not refreshed or restarted. This problem only occurs on new partitions. If you now add additional user mailboxes to this new partition, you will not have to do a refresh/restart of Messaging Server.

User Mailbox Directory Problems

A user mailbox problem exists when the damage to the message store is limited to a small number of users and there is no global damage to the system. The following guidelines suggest a process for identifying, analyzing, and resolving a user mailbox directory problem:

1. Review the log files, the error messages, or any unusual behavior that the user observes.

2. To keep debugging information and history, copy the entire *store_root***/mboxlist/** user directory to another location outside the message store.
3. To find the user folder that might be causing the problem, run the command **reconstruct -r -n**. If you are unable to find the folder using **reconstruct**, the folder might not exist in the **folder.db**.

   If you are unable to find the folder using the **reconstruct -r -n** command, use the "hashdir" command to determine the location.
4. Once you find the folder, examine the files, check permissions, and verify the proper file sizes.
5. Use **reconstruct -r** (without the **-n** option) to rebuild the mailbox.
6. If **reconstruct** does not detect a problem that you observe, you can force the reconstruction of your mail folders by using the **reconstruct -r -f** command.
7. If the folder does not exist in the **mboxlist** directory (*store_root***/mboxlist**), but exists in the **partition** directory *store_root***/partition**), there might be a global inconsistency. In this case, you should run the **reconstruct -m** command.
8. If the previous steps do not work, you can remove the **store.idx** file and run the **reconstruct** command again.

   Caution: You should only remove the **store.idx** file if you are sure there is a problem in the file that the **reconstruct** command is unable to find.
9. If the issue is limited to a problematic message, you should copy the message file to another location outside of the message store and run the command **reconstruct -r** on the **mailbox/** directory.
10. If you determine the folder exists on the disk (*store_root***/partition/** directory), but is apparently not in the database (*store_root***/mboxlist/** directory), run the command **reconstruct -m** to ensure message store consistency.

See "Repairing Mailboxes and the Mailboxes Database (reconstruct Command)" for more information on the **reconstruct** command.

Store Daemon Not Starting

If **stored** does not start and returns the following error message:

```
MessagingServer_home/bin/start-msg
MessagingServer_home: Starting STORE daemon ...Fatal error: Cannot find group in
name service
```

This indicates that the UNIX group configured in **local.servergid** cannot be found. **Stored** and others need to set their **gid** to that group. Sometimes the group defined by **local.servergid** gets inadvertently deleted. In this case, create the deleted group, add **mailsrv** to the group, change ownership of the *instance_root* and its files to **mailsrv** and the group.

User Mail Not Delivered Due to Mailbox Overflow

The message store has a hard limit of two gigabytes for a **store.idx** file, which is equivalent to about one million messages in a single mailbox (folder). If a mailbox grows to the point that the **store.idx** file will attempt to exceed two gigabytes, the user

will stop receiving any new email. In addition, other processes that handle that mailbox, such as imapd, popd, mshttpd, could also experience degraded performance.

If this problem arises, you will see errors in **mail.log_current** such as this:

```
05-Oct-2005 16:09:09.63 ims-ms Q 7 ... System I/O error. Administrator, check
server log for details. System I/O error.
```

In addition, the MTA log file will have an errors such as the following:

```
[05/Oct/2005:16:09:09 +0900] jmail ims_master[20745]: Store Error: Unable to
append cache for user/admin: File too large
```

You can determine this problem conclusively by looking at the file in the user's message store directory, or by looking in the **imta** log file to see a more detailed message.

The immediate action is to reduce the size of the file. Either delete some mail, or move some of it to another mailbox. You could also use **mboxutil -r** to rename the folder out of the way, or **mboxutil -d** to delete the folder (see "mboxutil").

Long-term, you will need to inform the user of mailbox size limitations, implement an aging policy (see "Configuring Message Expiration (Tasks)"), a quota policy (see "Message Store Quota Overview"), set a mailbox limit by setting **store.maxmessages** (see *Messaging Server Reference*), set up an archiving system, or make an adjustment to keep the mailbox size under control.

IMAP Events Become Slow

Symptom: After working fine for a short period of time, many IMAP events become unreasonably slow, with some events taking over a second.

Diagnosis: You have the Event Notification Service (ENS) plugin, **libibiff**, configured, but ENS is not running or not reachable. See "Administering Event Notification Service" for ENS details.

Solution: If you want ENS notifications, verify that the ENS is enabled and configured correctly. If you do not want ENS notifications, make sure that **libibiff** is not being loaded. Typical incorrect configuration:

```
notifytarget = /opt/sun/comms/messaging/lib/libibiff
ens.enable = 0
```

Instead, use one of the following configurations:

```
notifytarget =
ens.enable = 0
```

or

```
notifytarget = /opt/sun/comms/messaging/lib/libibiff
ens.enable = 1
```

Part IV

Managing the Message Store and Mailboxes

The message store is the component of Oracle Communications Messaging Server that contains the user mailboxes as well as the servers that provide IMAP, POP, and HTTP access to the mailboxes.

Unless otherwise noted, the message store (specifically, the **stored** process) should be up and running while performing management and maintenance tasks described in "Message Store Command Reference".

The size of the message store increases as the number of users, mailboxes, and log files increase. You can control the size of the message store by specifying limits on the size of mailboxes, by specifying limits on the total number of messages allowed, and by setting aging policies for messages in the store.

Depending on the number of users your server supports, the message store might require one physical disk or multiple physical disks. There are two ways to integrate this additional disk space into your system. The easiest way is to add additional message store partitions (see "Managing Message Store Partitions and Adding Storage"). Likewise, if you are supporting multiple hosted domains, you might want to dedicate a server instance to a single, large domain. With this configuration, you can designate a store administrator for a particular domain. You can also expand the message store by adding more partitions.

Part IV contains the following chapters:

- Message Store Message Type Overview
- Migrating Mailboxes to a New System
- Monitoring Disk Space
- Protecting Mailboxes from Deletion or Renaming
- Reducing Message Store Size Due to Duplicate Storage
- Specifying Administrator Access to the Message Store
- Constructing Valid Message Store UIDs and Folder Names
- Message Store Automatic Failover with Database Replication
- Administering Message Store Database Snapshots (Backups)
- Classic Messaging Server and Tiered Storage Overview
- Message Store Command Reference

Additional information about the Message Store can be found in the following chapters:

- Troubleshooting the Message Store
- Best Practices for Messaging Server and ZFS

40

Managing Mailboxes

This chapter describes how to list, create, remove, rename, move, and view information about mailboxes. It also describes how to find the directory location for a particular mailbox, restore expunged messages, and see how many users other than the mailbox owner have read messages in a shared IMAP folder. Specifically it describes how to use the **mboxutil**, **hashdir**, and **readership** utilities.

To Manage Mailboxes with mboxutil

Use the "mboxutil" command to perform typical maintenance tasks on mailboxes. **mboxutil** tasks include the following:

- List mailboxes
- List and remove orphaned and inactive mailboxes
- Create mailboxes
- Rename mailboxes
- Move mailboxes from one partition to another
- Expunge mailboxes
- Restore expunged messages that have not been purged
- List personal mailbox subscriptions and unsubscribed mailboxes that no longer exist
- You can also use the "mboxutil" command to view information about quotas. See "Managing Message Store Quotas" for more information.

When an end user deletes a mailbox, all messages are expunged and purged according to the value of **store.cleanupage**. However, expunged messages can be restored by using the **mboxutil -R** command, as long as you have enabled the **store.mailboxpurgedelay** option. Expunged messages are moved to the new location when a mailbox is renamed.

> **Caution:** Do not kill the **mboxutil** process in the middle of execution. If it is killed with **SIGKILL** (**kill -9**), it may potentially require that every server get restarted and a recovery be done.

Examples

To list all mailboxes for all users:

```
mboxutil -l
```

To list all mailboxes and also include path and ACL information:

```
mboxutil -l -x
```

To create the default mailbox named **INBOX** for the user **daphne**:

```
mboxutil -c user/daphne/INBOX
```

To delete a mail folder named **projx** for the user **delilah**:

```
mboxutil -d user/delilah/projx
```

To delete the default mailbox named **INBOX** and *all mail folders* for the user **druscilla**:

```
mboxutil -d user/druscilla/INBOX
```

To rename the mail folder **memos** to **memos-april** for the user **desdemona**:

```
mboxutil -r user/desdemona/memos user/desdemona/memos-april
```

To move the mail account for the user **dimitria** to a new partition:

```
mboxutil -r user/dimitria/INBOX user/dimitria/INBOX partition
```

where *partition* specifies the name of the new partition.

To move the mail folder named *personal* for the user **dimitria** to a new partition:

```
mboxutil -r user/dimitria/personal user/dimitria/personal partition
```

To Move Mailboxes to a Different Disk Partition

By default, mailboxes are created in the **primary** partition. If the partition gets full, additional messages cannot be stored. There are several ways to address the problem:

- Reduce the size of user mailboxes
- If you are using volume management software, add additional disks
- Create additional partitions ("To Add a Message Store Partition") and move mailboxes to the new partitions

If possible, add additional disk space to a system using volume management software because this procedure is the most transparent for the user. However, you might also move mailboxes to a different partition.

To move mailboxes to a different partition using the **mboxutil** command:

1. Create an input file of the mailboxes to be moved.

   The input file must consist of pairs of duplicate lines, because the other use of **mboxutil -r** is to rename folders. Thus, the first line of each pair is the old folder name and the second line is the new folder name. In the case of moving a user from one partition to another, the folder names are the same. Here is an example input file, **mvusers**, for three users:

   ```
   user/z4user21@east.example.com/inbox
   user/z4user21@east.example.com/inbox
   user/z4user22@east.example.com/inbox
   user/z4user22@east.example.com/inbox
   user/z4user23@east.example.com/inbox
   user/z4user23@east.example.com/inbox
   ```

2. Move the users to a new partition called **pool4**:

```
mboxutil -r -f /tmp/mvtest pool4
rename user/z4user21@east.sun.com/inbox to user/z4user21@east.sun.com/inbox
rename user/z4user22@east.sun.com/inbox to user/z4user22@east.sun.com/inbox
rename user/z4user23@east.sun.com/inbox to user/z4user23@east.sun.com/inbox
```

Note: If an error occurs with any individual mailbox move, the **mboxutil** command continues processing the rest of the input file. Thus, you must review the output (and possibly the default log file) to determine if any mailbox moves failed. Be sure to save that output.

To Remove Orphan Accounts

To search for orphaned accounts (orphaned accounts are mailboxes that do not have corresponding entries in LDAP) use the following "mboxutil" commands:

```
mboxutil -o
```

Command output follows:

```
mboxutil: Start checking for orphaned mailboxes
user/annie/INBOX
user/oliver/INBOX
mboxutil: Found 2 orphaned mailbox(es)
mboxutil: Done checking for orphaned mailboxes
```

Use the following command to create a file listing orphaned mailboxes that can be turned into a script file that deletes the orphaned mailboxes (example filename is **orphans.cmd**):

```
mboxutil -o -w orphans.cmd
```

The command output is as follows:

```
mboxutil: Start checking for orphaned mailboxes
mboxutil: Found 2 orphaned mailbox(es)
mboxutil: Done checking for orphaned mailboxes
```

Delete the orphan files with the following command:

```
mboxutil -d -f orphans.cmd
```

To Find a Mailbox's Directory Using hashdir

The mailboxes in the message store are stored in a hash structure for fast searching. Consequently, to find the directory that contains a particular user's mailbox, use the "hashdir" utility.

This utility identifies the directory that contains the message store for a particular account. This utility reports the relative path to the message store, such as **d1/a7/**. The path is relative to the directory level just before the one based on the user ID. The utility sends the path information to the standard output.

For example, to find the relative path to the mailbox for user **crowe**:

```
hashdir crowe
```

To Find Out How Many Users Have Read Messages in a Shared Folder

The "readership" utility reports on how many users other than the mailbox owner have read messages in a shared IMAP folder (see "Shared Folders Overview").

An owner of a IMAP folder may grant permission for others to read mail in the folder. A folder that others are allowed to access is called a **shared folder**. Administrators can use the **readership** utility to see how many users other than the owner are accessing a shared folder.

This utility scans all mailboxes and produces one line of output per shared folder, reporting the number of readers followed by a space and the name of the mailbox.

Each reader is a distinct authentication identity that has selected the shared folder within the past specified number of days. Users are not counted as reading their own personal mailboxes. Personal mailboxes are not reported unless there is at least one reader other than the folder's owner.

For example, the following command counts as a reader any identity that has selected the shared IMAP folder within the last 15 days:

```
readership -d 15
```

41

Backing Up and Restoring the Message Store

This chapter describes how to back up and restore Oracle Communications Messaging Server mailboxes. For conceptual information on the message store, see the following topics:

- Administering Message Store Database Snapshots (Backups)
- Message Store Disaster Backup and Recovery and Classic Message Store Directory Layout

Mailbox Backup and Restore Overview

Mailbox backup and restore is one of the most common and important administrative tasks. You must implement a backup and restore policy for your message store to ensure that data is not lost if the following problems occur:

- System crashes
- Hardware failure
- Accidental deletion of messages or mailboxes
- Problems when reinstalling or upgrading a system
- Natural disasters (for example, earthquakes, fire, hurricanes)
- Migrating users

You can back up and restore mailboxes by using the "imsbackup" and "imsrestore" command-line utilities or the integrated backup and restore solution that uses Oracle StorageTek Enterprise Backup Software (EBS).

Messaging Server provides a single-copy backup procedure. Regardless of how many user folders contain a particular message, during backup, the message file is backed up only once using the first message file found. The second message copy is backed up as a link to the name of the first message file, and so on. **imsbackup** maintains a hash table of all messages using the device and inode of the message files as the index. This method does have implications when restoring data, however. See "Considerations for Partial Restore" for more information.

> **Note:** You can also back up and restore the message store by backing up all relevant message files and directories. See "Message Store Disaster Backup and Recovery" for more information.

Backing up mailboxes includes three steps:

1. To Create a Mailbox Backup Policy

2. To Create Backup Groups
3. To Run the imsbackup Utility

To Create a Mailbox Backup Policy

Your backup policy will depend on several factors, such as:

- Peak Business Loads
- Full and Incremental Backups
- Parallel or Serial Backups

Peak Business Loads

Take into account peak business loads when scheduling backups for your system as this can reduce system load during peak hours. For example, backups are probably best scheduled for early morning hours such as 2:00 AM.

Full and Incremental Backups

Incremental backups (see "Incremental Backup") scan the message store for changed data and back up only what has changed. Full backups back up the entire message store. Determine how often the system should perform full as opposed to incremental backups. For example, you probably want to perform incremental backups as a daily maintenance procedure and full backups once a week.

Parallel or Serial Backups

When user data is stored on multiple disks, you can back up user groups in parallel. Depending on system resources, parallel backups can speed up the overall backup procedure. However, you might want to use serial backups to reduce backup impact on the server's performance. Whether to use parallel or serial backups can depend on many factors, including system load, hardware configuration, how many tape drives are available, and so on.

To Create Backup Groups

A backup group is an arbitrary set of user mailboxes defined by regular expressions. By organizing user mailboxes into backup groups, you can define more flexible backup management.

For example, you could create three backup groups, the first containing user IDs starting with the letters A through L, the second with users whose user IDs begin with M through Z, and the third with users whose user IDs begin with a number. Administrators could use these backup groups to back up mailboxes in parallel, or perhaps only certain groups on one day and other groups on another.

Consider the following points about backup groups:

1. They are arbitrary *virtual* groups of mail users that do not precisely map to the "Classic Message Store Directory Layout", although backup groups could resemble the message store directory.
2. Administrators define backup groups by using UNIX regular expressions. The regular expressions are defined in the *MessagingServer_home*/**config/backup-groups.conf** file.

3. When backup groups are referenced in **imsbackup** and **imsrestore**, they use the path format: */partition_name/backup_group*
4. When you run the **imsbackup** command, it evaluates the entire **backup-groups.conf**, and if it finds more than one group that matches a user, it uses the first match. For example, the following **backup-groups.conf** contains these definitions:

```
groupA=a.*
...
groupN=.*n$
```

Because both groups match the user ID **admin**, the **imsbackup** command uses the first match, which is **groupA**. Thus, **groupA** includes the **admin** mailbox. Furthermore, the **groupN** backup does not include the **admin** mailbox.

The format of **backup-groups.conf** is as follows:

```
group_name=definition
group_name=definition
.
.
.
```

Using the example described in the previous paragraph, you would use the following definitions to create the three backup groups:

```
groupA=[a-l].*
groupB=[m,-z].*
groupC=[0-9].*
```

> **Note:** In legacy configuration, you use the **backup-groups.conf** file to create backup groups. In Unified Configuration, you use the **msconfig** command to create backup groups.

In Unified Configuration, you can create the backup group using the **msconfig** command.

You can run the **msconfig** command in interactive mode as follows:

```
% msconfig
msconfig> set backup_group:groupA.re_pattern "[a-jA-J].*"
msconfig# show backup_group
role.backup_group:groupA.re_pattern = [a-jA-J].*
```

You can run the **msconfig** command in non-interactive mode as follows:

```
% msconfig set backup_group:groupA.re_pattern \"[a-jA-J].*\"
% msconfig show backup_group
role.backup_group:groupA.re_pattern = [a-jA-J].*
```

You can now scope **imsbackup** and **imsrestore** at several levels. You can backup the whole message store by using the following backup commands:

```
imsbackup -f <device> /
```

To back up all mailboxes for all users in **groupA** use the following command:

```
imsbackup -f <device> /<partition>/groupA
```

The default partition is called **primary**.

Pre-defined Backup Group

Oracle Communications Messaging Server includes one predefined backup group that is available without creating the **backup-groups** configuration file. This group is called **user** and includes all users. For example, the following command backs up all users on the **primary** partition:

```
imsbackup -f backupfile /primary/user
```

To Run the imsbackup Utility

To back up and restore your mailboxes, Messaging Server provides the "imsbackup" and "imsrestore" utilities. The **imsbackup** and **imsrestore** utilities do not have the advanced features found in general purpose tools like EBS. For example, the utilities have only very limited support for tape auto-changers, and they cannot write a single store to multiple concurrent devices. Comprehensive backup is achieved by using plug-ins to generalized tools like EBS. See "To Use StorageTek Enterprise Backup Software" for more information about using EBS.

Running the imsbackup Utility

With **imsbackup**, you can write selected contents of the message store to any serial device, including magnetic tape, a UNIX pipe, or a plain file. The backup or selected parts of the backup can later be recovered by using the **imsrestore** utility. The output of **imsbackup** can be piped to **imsrestore**.

The following example backs up the entire message store to **/dev/rmt/0**:

```
imsbackup -f /dev/rmt/0 /
```

This example backs up the mailboxes of user ID **joe** to **/dev/rmt/0**:

```
imsbackup -f /dev/rmt/0 /primary/user/joe
```

This example backs up all the mailboxes of all the users defined in the backup group **groupA** to **backupfile** (see "To Create Backup Groups"):

```
imsbackup -f /primary/groupA > backupfile
```

Incremental Backup

The following example backs up messages stored from May 1, 2004 at 1:10 pm to the present. The default is to back up all the messages regardless of their dates:

```
imsbackup -f /dev/rmt/0 -d 20040501:131000 /
```

This command uses the default blocking factor of 20. See "imsbackup" for a complete syntax description.

Regarding date-time stamp:

```
20040501:131000
YYYYMMDD:HHMMSS

2004 05 01 : 13 10 00
YYYY MM DD : HH MM SS
```

Excluding Bulk Mail When You Perform Backups

When you perform a backup operation, you can specify mailboxes that will be excluded from being backed up. By excluding bulk or trash mailboxes that can accrue large numbers of unimportant messages, you can streamline the backup session, reduce the time to complete the operation, and minimize the disk space required to store the backup data.

To exclude mailboxes, specify a value for the **store.backupexclude** option.

You can specify a single mailbox or a list of mailboxes separated by the "%' character. ("%' is an illegal character in a mailbox name.) For example, you could specify the following values:

Trash

Trash%Bulk Mail%Third Class Mail

In the first example, the folder **Trash** is excluded. In the second example, the folders **Trash**, **Bulk Mail**, and **Third Class Mail** are excluded.

Example commands:

```
cd /opt/sun/comms/messaging64/bin
msconfig set store.backupexclude "Trash%Bulk Mail%Third Class Mail"
msconfig show store.backupexclude
role.store.backupexclude = Trash%Bulk Mail%Third Class Mail
```

The backup utility backs up all folders in a user mailbox except those folders specified in the **store.backupexclude** option.

This feature works with the Messaging Server backup utility, StorageTek Enterprise Backup Software, and third-party backup software.

You can override the **store.backupexclude** setting and back up an excluded mailbox by specifying its full logical name during the operation. For example, suppose the Trash folder has been excluded. You can still back up Trash by specifying the following:

```
/primary/user/user1/trash
```

However, if you specify

```
/primary/user/user1
```

the Trash folder is excluded.

To Restore Mailboxes and Messages

To restore messages from the backup device, use the "imsrestore" command. For example, the following command restores messages for **user1** from the file **backupfile**.

```
imsrestore -f backupfile /primary/user1
```

Considerations for Partial Restore

A partial restore is when only a part of the message store is restored. A full restore is when the entire message store is restored. The message store uses a single-copy message system. That is, only a single copy of any message is saved in the store as a single file. Any other instances of that message (for example, when a message is sent to multiple mailboxes) are stored as links to that copy. Because of this, there are implications when restoring messages. For example:

- **Full Restore.** During a full restore, linked messages still point to the same inode as the message file to which they are linked.
- **Partial Backup/Restore.** During a partial backup and partial restore, however, the single-copy characteristic of the message store might not be preserved.

The following examples demonstrate what happens to a message that is used by multiple users when a partial restore is performed. Assume there are three messages, all the same, belonging to three users A, B, and C, as follows:

```
A/INBOX/1
B/INBOX/1
C/INBOX/1
```

Example 1. In the first example, the system performs a partial backup and full restore procedure as follows:

1. Back up mailboxes for users B and C.
2. Delete mailboxes of users B and C.
3. Restore the backup data from step 1.

In this example, **B/INBOX/1** and **C/INBOX/1** are assigned a new inode number and the message data is written to a new place on the disk. Only one message is restored. The second message is a hard link to the first message.

Example 2. In this example, the system performs a full backup and a partial restore as follows:

1. Perform full backup.
2. Delete mailboxes for user A.
3. Restore mailboxes for user A.

A/INBOX/1 is assigned a new inode number.

Example 3. In this example, partial restore might require more than one attempt:

1. Perform full backup.

   B/INBOX/1 and **C/INBOX/1** are backed up as links to **A/INBOX/1**.
2. Delete mailboxes for users A and B.
3. Restore mailboxes for user B.

   The restore utilities ask the administrator to restore **A/INBOX** first.
4. Restore mailboxes for users A and B.
5. Delete mailboxes for user A (optional).

Note: If you want to ensure that all messages are restored for a partial restore, you can run the **imsbackup** command with the **-i** option. The **-i** option backs up every message multiple times if necessary.

If the backup device is seekable (for example, a drive or tape), **imsrestore** seeks to the position containing **A/INBOX/1** and restores it as **B/INBOX/1**. If the backup device is non-seekable (for example, a UNIX pipe), **imsrestore** logs the object ID and the ID of the depending (linked) object to a file, and the administrator must invoke **imsrestore** again with the **-r** option to restore the missing message references.

To Restore Messages from a Mailbox that Has Been Incrementally Backed-up

If you are restoring messages from a mailbox that has been incrementally backed-up, and if that mailbox exists on the server on which you want to restore the messages, then restoring the messages requires a straightforward **imesrestore**. However, if you want to restore messages from a mailbox that has been incrementally backed-up, and if that mailbox no longer exists, you must follow different restore procedures.

Use one of the following procedures to restore messages to a mailbox that does not exist on the message store server:

- During the restore operation, disable delivery of messages to the user. Do this by setting the LDAP attribute **mailDeliveryOption** to **hold**.
- Before you use **imesrestore**, create the mailbox with the **mboxutil -c** command.

The reason why these instructions must be followed for restoring an incremental backup is as follows: When a mailbox has been deleted or is being migrated, the **imsrestore** utility recreates the mailbox with the mailbox unique identification validity and message unique identifications (UIDs) stored in the backup archive.

In the past, when **imsrestore** would recreate a deleted or migrated mailbox, it would assign a new UID validity to the mailbox and new UIDs to the messages. In that situation, a client with cached messages would have to resynchronize the mailbox UID validity and message UIDs. The client would have to download the new data again, increasing the workload on the server.

With the new **imsrestore** behavior, the client cache remains synchronized, and the restore process operates transparently with no negative impact on performance.

If a mailbox exists, **imsrestore** assigns new UIDs to the restored messages so that the new UIDs remain consistent with the UIDs already assigned to existing messages. To ensure UID consistency, **imsrestore** locks the mailbox during the restore operation. However, because **imsrestore** now uses the mailbox UID validity and message UIDs from the backup archive instead of assigning new UID values, UIDs could become inconsistent if you perform incremental backups and restores.

If you perform incremental backups with the **-d** date option of the **imsbackup** utility, you might have to invoke **imsrestore** multiple times to complete the restore operation. If incremental backups were performed, you must restore the latest full backup and all subsequent incremental backups.

New messages can be delivered to the mailbox between the restore operations, but in this case, the message UIDs can become inconsistent. To prevent inconsistency in the UIDs, you need to take one of the actions previously described on this page.

To Use StorageTek Enterprise Backup Software

Messaging Server includes a backup API that provides an interface with third-party backup tools, such as EBS. The physical message store structure and data format are encapsulated within the backup API. The backup API interacts directly with the message store. It presents a logical view of the message store to the backup service. The backup service uses the conceptual representation of the message store to store and retrieve the backup objects.

Messaging Server provides an Application Specific Module (ASM) that can be invoked by the EBS's **save** and **recover** commands to back up and restore the message store data. The ASM then invokes the Messaging Server **imsbackup** and **imsrestore** utilities.

Note: This section provides information about how to use EBS with the Messaging Server message store. To understand the EBS interface, see your StorageTek Enterprise Backup Software documentation.

To Back Up Data By Using StorageTek Enterprise Backup Software

1. Create a symbolic link from **/usr/lib/nsr/imsasm** to *MessagingServer_home*/**lib/msg/imsasm**.
2. From Oracle or EMC, obtain a copy of the **nsrfile** binary and copy it to the following directory:**/usr/bin/nsr**

   This is required only if you are using an older version of Networker (5.x). With Networker 6.0 and above, **nsrfile** is automatically installed under **/usr/bin/nsr**.
3. If you want to back up users by groups, perform the following steps:
   a. Create a backup group file. See "To Create Backup Groups" for more information.
   b. To verify your configuration, run **mkbackupdir.sh**. Look at the directory structure created by **mkbackupdir.sh**. The structure should look similar to that shown in "Classic Message Store Directory Layout". If you do not specify a **backup-groups.conf** file, the backup process uses the default backup group **ALL** for all users.
4. In the directory **/nsr/res/**, create a **res** file for your save group to invoke the **mkbackupdir.sh** script before the backup. See "Classic Message Store Directory Layout" for an example.

   Note: Earlier versions of Networker have a limitation of 64 characters for the save set name. If the name of this directory plus the logical name of the mailbox (for example, **/primary/groupA/fred**) is greater than 64 characters, then you must run **mkbackupdir.sh -p**. Therefore, you should use a short path name for the **-p** option of **mkbackupdir.sh**. For example the following command will create the backup image under the directory **/backup**:

   ```
   mkbackupdir.sh -p /backup
   ```

   Important: The backup directory must be writable by the message store owner (example: **mailsrv**).

   The following is a sample backup groups directory structure.

   ```
   /backup/primary/groupA/amy
   /bob
   /carly
   /groupB/mary
   /nancy
   /zelda
   /groupC/123go
   /1bill
   /354hut
   ```

   The following example shows a sample **res** file named **IMS.res** in the **/nsr/res** directory:

```
type: savepnpc;
precmd: "echo mkbackupdir started",
"/usr/example/server5/msg-example/bin/mkbackupdir.sh -p /backup";
pstcmd: "echo imsbackup Completed";
timeout: "12:00 pm";
```

You are now ready to run the EBS interface as follows:

5. Create the Messaging Server save group if necessary.
   a. Run **nwadmin**.
   b. Select Customize | Group | Create.
6. Create a backup client using **savepnpc** as the backup command:
   1. Set the save set to the directory created by **mkbackupdir**. For a single session backup, use **/backup**.

      For parallel backups, use **/backup/***server/group*. Be sure you have already created *group* as defined in "To Create Backup Groups". You must also set the parallelism to the number of backup sessions. See "To Back Up Data By Using StorageTek Enterprise Backup Software" for more information.
7. Select Group Control | Start to test your backup configuration. Example. Creating A Backup Client in EBS: To create a backup client in EBS. From **nwadmin**, select Client | Client Setup | Create

```
Name: example
Group: IMS
Savesets:/backup/primary/groupA
/backup/secondary/groupB
/backup/tertiary/groupC
.
.
Backup Command:savepnpc
Parallelism: 4
```

Restoring Data Using StorageTek Enterprise Backup Software

To recover data, you can use the EBS **nwrecover** interface or the **recover** command-line utility. The following example recovers user **a1**'s INBOX:

```
recover -a -f -s example /backup/example/groupA/a1/INBOX
```

The next example recovers the entire message store:

```
recover -a -f -s example /backup/example
```

To Use a Third Party Backup Software (Besides StorageTek Enterprise Backup Software)

Messaging Server provides two message store backup solutions, the command line **imsbackup** and the StorageTek Enterprise Backup Software. A large message store running a single **imbackup** to back up the entire message store can take a significant amount of time. The EBS solution supports concurrent backup sessions on multiple backup devices. Concurrent backup can shorten backup time dramatically (backups of 25GB of data per hour have been achieved).

If you are using another third party concurrent backup software (for example, Netbackup), you can use the following method to integrate your backup software with the Messaging Server.

1. Divide your users into groups (see "To Create Backup Groups") and create a **backup-groups.conf** file under the directory *MessagingServer_home***/config/**.

   Note: This backup solution requires additional disk space. To backup all the groups concurrently, the disk space requirement is two times the message store size. If you do not have that much disk space, divide your users into smaller groups, and then backup a set of groups at a time. For example group1 - group5, group6 - group10. Remove the group data files after backup.

2. Run **imsbackup** to back up each group into files under a staging area. The command is **imsbackup -f <device>/<instance>/<group>** You can run multiple **imsbackup** processes simultaneously. For example:

   ```
   imsbackup -f- /primary/groupA > /bkdata/groupA &
   imsbackup -f- /primary/groupB > /bkdata/groupB &
   . . .
   ```

3. Use your third party backup software to back up the group data files in the staging area (in our example that is **/bkdata**).

4. To restore a user, identify the group filename of the user, restore that file from tape, and then use **imsrestore** to restore the user from the data file.

Troubleshooting Backup and Restore Problems

This section describes common backup and restore problems and their solutions.

- **Problem**: **msprobe** restarts everything during a long **imsrestore** during message store migration. This can also happen with **imsbackup**, **imsimport**, or any processing intensive utility.
- **Solution**: When **imsrestore** or any processing intensive operation takes significantly more system resources than normal, and continues doing so longer than the **msprobe** interval, there might be a temporary backlog of DB transaction log files to be cleared. If there are more files than specified in **store.maxlog**, then **msprobe** may erroneously restart all the processes during a restore. To prevent this from happening, disable **msprobe** during the **imsrestore**.
- **Problem**: When I do an restore of a folder or INBOX using **imsrestore** or **imsasm**, it appends all the messages in that folder onto the current folder. This results in multiple copies of the messages in that folder.
- **Solution**: Make sure the **-i** flag of **imsrestore** is not set in the **imsasm** script.
- **Problem**: I want to do an incremental backup of just new messages added in a mail folder, but when I try, the entire folder gets backed up. How do I just back up the new messages?
- **Solution**: Set the **-d***datetime* flag on **imsbackup**. This will backup messages stored from the specified date and time to the present. The default is to back up all messages regardless of their dates.

Message Store Disaster Backup and Recovery

A disaster refers to a catastrophic failure of the entire message store as opposed to a mailbox or set of mailboxes. That is, a situation where all data on the message store servers are lost. A complete message store disaster restore will consist of restoring the following lost data:

- All message store data. These can be backed up using the procedures described above. If file system backup method is used, be sure to back up the following data:
  - All message store partitions
  - The message store database files at *DataRoot***/store/mboxlist**.
  - The message store database snapshots at *DataRoot***/store/dbdata/snapshots** (Note that the location of message store database snapshot files can be configured with the **store.snapshotpath** option.)
- Configuration data. Including the local configuration file at *DataRoot***/config**. See also "Classic Message Store Directory Layout".

If you want to back up your message store for disaster recovery, you can use file system snapshot tools to take a snapshot of the file system. The snapshot **must** be a *point-in-time* file system snapshot.

It is best to capture all the data (message store partitions, database files and so on) at the same point-in-time, however, if this cannot be done, then you must backup the data in this order:

1. Database snapshots
2. Database files
3. Message Store indexes if separated from the messages using the **partition:***partition-name***.messagepath** option (**store.partition.***.**messagepath** in legacy configuration). For additional information, see the discussion about the **messagepath** option in *Messaging Server Reference*.
4. Messages
5. Configuration data

If database files and Message Store indexes are not backed up with the same point-in-time snapshot (or database files, Message Store indexes, and Messages, if indexes are not separate) then **reconstruct -m** is required after restore.

42

Administering Very Large Mailboxes

This chapter describes how Oracle Communications Messaging Server enables the use of "very large" mailboxes.

Very Large Mailboxes Overview

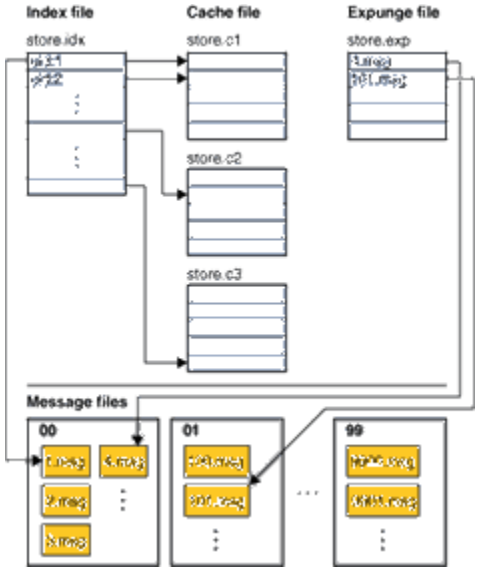
To increase the mailbox size limit and improve **expunge** performance, the index records and cache records have been split into separate files with support added for multiple cache files. An index file contains a mailbox header and a 128-byte fixed-length record for every message in the mailbox. The index record contains the location and size of the corresponding cache record. The cache files contain frequently used data in variable length records. The cache file size is configurable (**store.maxcachefilesize**). The default cache size is 500 Mbs, with a maximum size 2 Gbs. New cache records are appended to the newest cache file. As a cache file fills up, a new cache file is created, enabling a mailbox to continue to grow.

To optimize expunge performance, only the index file **store.idx** is purged when a mailbox is expunged. Expunge removes the obsoleted index records from the index file. Cache record and message file removal is deferred until the size of the expunged data exceeds a configurable threshold (**store.purge.count** or **store.purge.percentage** for cache records; **store.cleanupsize** for the *number* of messages). When the expunged size exceeds the threshold, expunge enqueues a purge request to the "Message Store Maintenance Queue". Then impurge dequeues the request, removes the unused message files and purge the cache files when the expunged size exceeds the purge threshold.

The Structure of a Mailbox

The following figure is an example of a large mailbox:

Figure 42–1 Large Mailbox Example



Mailbox Size Limit

IMAP defines the UID as a 32-bit value. Therefore, the maximum number of messages in a mailbox is limited to 4,294,967,295. For a 64 bit messaging server, the maximum number of messages in a folder is 4,294,967,295. For a 32-bit messaging server, the maximum number of messages in a folder is 16,777,215. The **msconfig** option **store.maxmessages** can be used to limit the size of a folder. The default limit is 16,000,000.

Note, the 4 billion limit is a hard limit. Effective limit is much smaller (normally less than 1 million). Mailbox expunge has to rewrite the **store.idx file**. The larger the mailbox, the longer it takes to expunge.

Mailbox Migration

Mailboxes are migrated to the new format automatically when they are opened. The operation is transparent to the end users.

Pre-Deployment Preparations

The high-level steps for preparing to use very large mailboxes include:

1. Configuring the maximum cache file size
2. Configuring the mailbox expunge size
3. Configuring **store.maxmessages**, quotas or expire rules to prevent mailboxes from getting too big

Table 42–1 describes the new **msconfig** options for managing large mailboxes.

Table 42–1 msconfig Options for Managing Large Mailboxes

msconfig Option	Description
store.maxcachefilesize	Sets the maximum cache file size in bytes, maximum of 2 Gbs.

Table 42–1 (Cont.) msconfig Options for Managing Large Mailboxes

msconfig Option	Description
store.cleanupsize	Cleans the mailbox when the number of expunged messages exceeds this value. Default is 100.

Checking Mailbox Data

The **imcheck** command line utility can be used to dump the data of mailboxes in readable format. For example, **imcheck -m** mailbox dumps the content of a **store.idx** file:

```
imcheck -m user/dumbo/INBOX
--------------------------------------------------------------------------------
Name: user/dumbo
Version: 103
Exists: 4
Flags: 0
Largest Msg: 1521 bytes
Last Append: 20080131104858
Last Repair: -
Last UID: 4
Oldest Msg: 20080131104843
Oldest Uid: 1
Quota Used: 2394
Bytes Expunged: 0
UID Validity: 1201805323
Last CacheId: 1
Start Offset: 256
Append CacheId: 1
ACL: dumbo lrswipcdan
Subscribed: 0
Partition: primary
Path: /var/opt/SUNWmsgsr/store/partition/primary/=user/94/60/=dumbo
Msg Path: /var/opt/SUNWmsgsr/store/partition/primary/=user/94/60/=dumbo

MsgNo Uid Internal-Date Sent-Date Size HSize Cache-Id C-Offset C-Len Last-Updated
Save-Date MT SFlags UFlags Original-Uid Message-id
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
1 1 20080131104843 20080131104843 257 244 1 16 864 20080131104843 20080131104843 2
R 0.0.0 1201805323-1 -
2 2 20080131104851 20020301191039 338 229 1 880 816 20080131104851 20080131104851
2 R 0.0.0 1201805323-2 <001@red.iplanet.com>
3 3 20080131104855 20020301191039 1521 241 1 1696 840 20080131104855
20080131104855 2 R 0.0.0 1201805323-3 <002@red.iplanet.com>
4 4 20080131104858 20050919110532 278 252 1 2536 860 20080131104858 20080131104858
1 R 0.0.0 1201805323-4 <003@red.iplanet.com>
```

imcheck -m mailbox -c *msgno* displays the cache records of a message:

```
imcheck -m user/dumbo/INBOX -c 1
Cache items of user/dumbo message number 1:

ENVELOPE {300}
("Thu, 31 Jan 2008 10:48:43 -0800" "welcome" (("Mail Administrator" NIL
"Postmaster" "puzzle.red.iplanet.com")) (("Mail Administrator" NIL "Postmaster"
"puzzle.red.iplanet.com")) (("Mail Administrator" NIL "Postmaster"
"puzzle.red.iplanet.com")) ((NIL NIL "dumbo" "red.iplanet.com")) NIL NIL NIL NIL)
```

```
BODYSTRUCTURE {75}
("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "7BIT" 13 1 NIL NIL NIL NIL)

BODY {59}
("TEXT" "PLAIN" ("CHARSET" "us-ascii") NIL NIL "7BIT" 13 1)

SECTION {48}
Header offset: 0 len: 244
Body offset: 244 len: 13
1 subparts
(1) offset: 244 len: 13 charset: 0 encoding: 0

CACHEHEADERS {244}
Subject: welcome
To: dumbo@red.iplanet.com
Date: Thu, 31 Jan 2008 10:48:43 -0800
From: Mail Administrator <Postmaster@puzzle.red.iplanet.com>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit



FROM {53}
mailadministrator <postmaster@puzzle.red.iplanet.com>

TO {23}
<dumbo@red.iplanet.com>

CC {0}


BCC {0}


SUBJECT {9}
"welcome"

XSENDER {0}
```

43

Message Store Message Expiration

This chapter describes message expiration concepts. See "Configuring Message Expiration (Tasks)" for information on message removal tasks.

Message expiration automatically removes messages from the message store based on criteria that you set. For example, you can remove old messages, overly large messages, seen or deleted messages, messages with specific Subject: lines, messages of a certain type, and so on.

> **Note:** Oracle Communications Messaging Server removes messages without giving a warning, so it is important to inform users about message expiration policies. Unexpected message removal can be a source of consternation for users and administrators.

imexpire Overview

Message expiration is performed by the **imexpire** utility, which performs a specific action to the expired messages. (See "Deleting, Expunging, Purging, and Cleaning Up Messages" for details on the message removal process.) You can launch the **imexpire** utility from the command-line or schedule it to launch through the **imsched** daemon. You specify a set of expiration rules in the **store.expirerule** file. You can have multiple rules files, each located the directory that pertains to the scope of the rules. That is, rules that apply globally to the entire message store are put in one directory, rules that apply to a partition in another, rules that apply to users in yet another, and so on.

> **Note:** Although global expiration rules can be specified with **store.expire.***attribute* **configutil** options, use **store.expirerule** files to specify these rules. If too many rules are created by using **configutil**, performance problems can result.

imexpire loads all of the expire rules at start up. By default, **imexpire** creates one thread per partition. Each thread goes through the list of user folders under its assigned partition and loads the local expire rule files as it goes. The expire function checks each folder against the expire rules applicable to this folder and expunges messages as needed.

It is also possible to exclude specified users from the expire rules by adding their user ID, one per line, in a file called **expire_exclude_list** in the *MessagingServer_home***/config/** directory.

To Deploy the Message Expiration Feature

Message expiration requires the following three steps:

1. Define message expiration policy: Which messages will be expired? What users, folders, domains, and partitions will have messages expired? What size, message age, and headers will define the removal criteria? Define the scope of messages to be removed. See "To Define Message Expiration Policy" for more information.
2. Specify the **imexpire** rules to implement this policy. See "To Set Rules Implementing Message Expiration Policy" for more information.
3. Specify the **imexpire** scheduling. See "To Schedule Message Expiration and Logging Level" for more information.

To Define Message Expiration Policy

You can define message expiration based on criteria such as:

- **Age of Message** – Expire messages older than X days. Attribute: **messagedays**.
- **Message Count** – Expire messages in a folder exceeding X messages. Attribute: **messagecount**.
- **Age of Oversized Message** – Expire messages that exceed X bytes after Y days grace period. Attributes: **messagesize** and **messagesizedays**.
- **Seen and Deleted Message Flag** – Expire messages with the Seen or Deleted flag set. These criteria can be set to "**and**" or "**or**." If set to **or**, the message's Seen/Delete flag will cause expiration regardless of other criteria. If set to and, the message's Seen/Delete flag must be set along with passing all other specified criteria. Attributes: **seen** and **deleted**.
- **Header Field of Message** – Allows you to specify a header and string as criteria for expiring a message, for example, removing all messages with the header "Subject: Work from Home." Note that this feature also allows you to use message type as a criteria too. See "Expiring Messages by Message Type" for more information.
- **Folder of Messages** – Allows you to specify the folder on which to expire messages. Attribute: **folderpattern**. Note that this attribute only uses the modified UTF-7 character set.

> **Note:** **imexpire** does not allow you to delete or preserve messages based on how long it has been since that message was read. For example, you cannot specify that messages that have not been read for 200 days will be removed.

Examples of Message Expiration Policy

Example 1: Remove all messages 365 days old in a folder exceeding 1,000 messages.

Example 2: Remove messages in domain **example.org** that are older than 180 days.

Example 3: Remove all messages that have been marked as deleted.

Example 4: Remove messages in example.com that have been marked as seen, are older than 30 days, are larger than 100 kilobytes, from folders exceeding 1,000 messages, with the header **X-spam**.

To Set Rules Implementing Message Expiration Policy

Rules are set by putting them into a **store.expirerule** file. An example of two global **store.expirerule** rules is shown below:

```
Rule1.regexp: 1
Rule1.folderpattern: user/.*/trash
Rule1.messagedays: 2
Rule2.regexp: 1
Rule2.folderpattern: user/.*
Rule2.messagedays: 14
```

In this example, Rule 1 specifies that all messages in all users' trash folders are removed after two days. Rule 2 specifies that all messages in any folder in the message store are removed after 14 days.

Expiration Rules Guidelines

This section sets the guidelines for the **store.expirerule** file rules.

> **Note:** In earlier Messaging Server releases, expiration rules could be set with **configutil** options **store.expirerule**.*attribute* (see *Messaging Server Reference*.) This is still true, but expire rules using header constraints (example: expiring a message with a specific subject line) are not supported. Also, regular expressions in the expire rules created with **configutil** need to be POSIX compliant rules. If you want to use UNIX compliant regular expressions you must use the store.expire file. In addition, using both **configutil** options and the global store.expirerule configuration file is not supported. If the configuration file is present, **configutil** options are not used. In any case, it is best to use **store.expirerule** to specify all expiration rules.

- Rules are specified in a file called **store.expirerule**.
- Multiple expiration criteria can be specified with the same rule. (See preceding example.)
- Rules can apply to the entire message store (global rules), a partition, a user, or a folder.
  - The global rules are stored in the *MessagingServer_home*/**config/store.expirerule** file.

    > **Note:** Each global rule will be checked against every mailbox, which can cause some processing overhead depending on the number of global rules you specify. For this reason, you should not specify partition, mailbox or user rules in the global rules file. In general, you should try not to put any more expiration rules than necessary in this file.

  - Partition rules are stored in *store_root*/**partition/***partition_name*/**store.expirerule** (more accurately, the location specified by the **store.partition.*.path configutil** option).
  - User rules are specified in *store_root*/**partition/***partition_name/userid*/**store.expirerule** or by specifying the **folderpattern** rule to be **user/***userid*/**.***.

  - Folder rules are specified in *store_root*/**partition/***partition_name/userid/folder*/**store.expirerule** or by specifying the **folderpattern** rule to be **user/***userid/folder*.

    Multiple non-global rules (user, folder, partition) using *rule_name* was only implemented starting in Messaging Server 6.2p4.

- Multiple expire rules can be applied to a mailbox at the same time. An expire policy for a mailbox consists of global rules and local rules. Local rules apply to the mailbox under the same directory and all of its sub-folders.
- **imexpire** unifies all of the expiration rules applying to a mailbox, unless there is an exclusive rule specified for this mailbox (see Table 43–1). The resulting rule set represents the most restrictive expiration policy based on all applicable rules. For example, if rule X expires messages such that the maximum message life is 10 days, and rule Y specifies 5 days, the union will be 5 days.

Note: When **join: and** attribute is specified in a rule, all expiration criteria specified in a rule must be satisfied for the message to expire (see Table 43–1). This option is valid if the rules are configured as exclusive rule. Example:

```
rule.regexp: 1
rule.folderpattern: user/*
rule.messagedays: 60
rule.messageheader.X-MESSAGE-TYPE: email
rule.exclusive: 1
rule.join: and
```

In this example, the rule specifies to remove the messages that are older than 30 days AND contains message header X-MESSAGE-TYPE: Email.

Table 43–1 imexpire Attributes

Attribute	Description (Attribute Value)
action	Specifies an action to perform on the messages caught by the expire rules. The possible values are: ■ **discard** – discards the message. This is the default. ■ **report** – prints the mailbox name, uid-validity and uid to stdout. ■ **archive** – archives the message with the Compliance and Content Management System and then discards the message. ■ **fileinto:** *folder* – files the message into the specified folder. The shared folder prefix can be used to file messages to folders owned by another user. If the specified folder does not exist, **imexpire** creates it.
exclusive	Specifies whether or not this is an exclusive rule. If specified as **exclusive**, only this rule applies to the specified mailbox(es) and all other rules are ignored. If more than one exclusive rule exists, the last exclusive rule loaded will be used. For example, if a global and a local exclusive rule are specified, the local rule will be used. If there is more than one global exclusive rule, the last global rule listed by **configutil** is used. **(1/0)**
expires	**imexpire** will select the message if the date value specified with these header fields is older than the expiration date based on the messagedays attribute. If multiple expiration header fields are specified, the earliest expiration date will be used. (string)

Table 43–1 (Cont.) imexpire Attributes

Attribute	Description (Attribute Value)
expiry-date	**imexpire** will select the message if the date value specified with these header fields is older than the expiration date based on the messagedays attribute. If multiple expiration header fields are specified, the earliest expiration date will be used. (string)
foldpattern	Specifies the folders affected by this rule. The format must start with a **user/**, which represents the directory *store_root*/**partition/*/**. See Table 43–2, " imexpire Folder Patterns Using Regular Expressions". (POSIX regular expression)
messagecount	Maximum number of messages in a folder. Oldest messages are expunged as additional messages are delivered. (integer)
foldersize	Maximum size of folder before the oldest messages are expunged when additional messages are delivered. (integer in bytes)
messagedays	Number of days in the message store before being expunged. (integer)
messagesize	Maximum size of message in bytes before it is marked to be expunged. (integer)
messagesizedays	Grace period. Days an over-sized message should remain in a folder. (integer)
messageheader.*header*	Specifies a header field and string. Values are not case-sensitive and regular expressions are not recognized. Example: `Rule1.messageheader.Subject: Get Rich Now!` Headers other than Subject: can be used.
regexp	Enable UNIX regular expressions in rules creation. (1 or 0). If not specified, IMAP expressions will be used.
savedays	Number of days the messages are saved in a folder until they are expunged.
seen	**seen** is a message status flag set by the system when the user opens a message. If the attribute **seen** is set to and, then the message must be seen and other criteria must be met before the rule is fulfilled. If the attribute **seen** is set to **or**, then the message only needs to be seen or another criteria be met before the rule is fulfilled. **(and/or)**
sieve	A Sieve test specifying message selection criteria. Example: `Rule17.sieve: header :contains "Subject" "Vigara"`
deleted	**deleted** is a message status flag set by the system when the user deletes a message. If the attribute **deleted** is set to **and**, then the message must be deleted and another criteria must be met before the rule is fulfilled. If the attribute **deleted** is set to **or**, then the message only needs to be deleted or another criteria be met before the rule is fulfilled. **(and/or)**
join	**join** may only be specified in an exclusive rule. The default value is **or**. Specifying **join: and** means all criteria must be met before the rule is fulfilled. (**and /or**)

userflag.*flag-name*	Valid values are **and/or**. Specify user IMAP flags in expiration rules. The following example shows a rule expires those messages with the junk flag set, and which are older than 30 days:
	`messagedays:30`
	`userflag.junk: and`

channel	Specify the name of an MTA channel as which to run for purposes of spam/virus filtering. In order for a channel attribute value to take effect, the **expiresieve** Message Store option must be enabled. (string)

Table 43–1 (Cont.) imexpire Attributes

Attribute	Description (Attribute Value)
rescanhours	When using **imexpire** to perform post-delivery spam/virus filtering, the **rescanhours** tells **imexpire** to rescan those message that have not been scanned for the specified number of hours. In order for any **rescanhours** attribute value to take effect, the **expiresieve** Message Store option must be enabled. (integer)

Localized Mailbox Names in imexpire

The IMAP protocol specifies that mailbox names use modified UTF-7 encoding. Messaging Server supports localized character sets on external interfaces so that mailbox names can be localized. Internally, however, the system converts the localized name to MUTF-7. Thus, a folder that has a localized mailbox name on a client will have a corresponding mailbox file name in MUTF-7. (Note that IMAP error messages will output mailbox names in MUTF-7 and not the localized character set.)

In general, most message store utilities that require mailbox names expect the names in the localized character set, although they may have an option flag that allows a different character set to be used. These utilities include **reconstruct**, **mboxutil**, **imsbackup**, **imsrestore**, and **hashdir**. However, **imexpire** requires that the mailbox name, specified as the attribute **folderpattern**, be in MUTF-7. Using a localized name will not work.

To obtain the appropriate **folderpattern** for **imexpire** it may be necessary to convert a localized mailbox name to the modified UTF-7 equivalent. This can be done using the **mboxutil -E** command as follows:

```
mboxutil -l -p user/han/*

  msgs  Kbytes last msg         partition   quotaroot mailbox

    57     100 2010/04/29 11:18 primary       5242880 user/han/INBOX
     1       1 2010/04/30 12:56 primary             - user/han/<multibyte_mailbox>

mboxutil -l -E MUTF-7 -p user/han/*

    57     100 2010/04/29 11:18 primary       5242880 user/han/INBOX
     1       1 2010/04/30 12:56 primary             -
user/han/&kAFP4W4IMH8wojCkMMYw4A-
```

The first **mboxutil** shows the localized mailbox name. The second **mboxutil** shows the mailbox name in MUTF-7. The MUTF-7 mailbox name is identical to the IMAP list command:

```
x list "" *
* LIST (\NoInferiors) "/" INBOX
* LIST (\HasNoChildren) "/" &kAFP4W4IMH8wojCkMMYw4A-
```

To convert the local **charset** to modified UTF-7 encoding, use the **mboxutil** command with the **-E** option:

```
mboxutil -l -E MUTF-7 -P user/han/<multibyte_mailbox>

  msgs  Kbytes last msg         partition   quotaroot mailbox

     1       1 2010/04/30 12:56 primary             -
user/han/&kAFP4W4IMH8wojCkMMYw4A-
```

Note that **mboxutil -E** can be used for any command that requires the use of a MUTF-7 mailbox name including **imexpire**.

Setting imexpire Rules Textually

Message expiration rules are set by specifying expire criteria in a **store.expirerule** file. The **store.expirerule** file contains one expire criteria per line. An expire criteria of the global rule configuration file (*DataRoot*/**store/store.expirerule**) has the following format:

rule_name.attribute: value

"Example imexpire Rules" shows a set of global expiration rules in *MessagingServer_home*/**config/store.expirerule**.

Rule 1 sets the global expiration policy (that is, policy that applies to all messages), as follows:

- Enable UNIX regular expressions in rules creation.
- Removes messages larger than 100,000 bytes after 3 days.
- Removes messages deleted by the user.
- Removes any message with the strings "Viagra Now!" or "XXX Porn!" in the Subject: header.
- Limits all folders to 1,000 messages, the system removes the oldest messages on a folder to keep the total to 1.000.
- Removes all messages older than 365 days.

Rule 2 sets the message expiration policy for users at the hosted domain **example.org**. It limits mailbox sizes to 1 megabyte, removes messages that have been deleted, and removes messages older than 14 days.

Rule 3 sets the message expiration policy for messages in the inbox folder of user **f.dostoevski**. It removes messages with a subject line having the expression "On-line Casino."

Example imexpire Rules

```
Rule1.regexp: 1
Rule1.folderpattern: user/.*
Rule1.messagesize: 100000
Rule1.messagesizedays: 3
Rule1.deleted: or
Rule1.Subject: Vigara Now!
Rule1.Subject: XXX Porn!
Rule1.messagecount: 1000
Rule1.messagedays: 365
Rule2.regexp: 1
Rule2.folderpattern: user/.*@example.org/.*
Rule2.exclusive: 1
Rule2.deleted: or
Rule2.messagedays: 14
Rule2.messagecount: 1000
Rule3.folderpattern: user/f.dostoevski/inbox
Rule3.Subject: On-line Casino
```

Setting imexpire Folder Patterns

Folder patterns can be specified using POSIX regular expressions by setting the **imexpire** attribute **regex** to 1. If not specified, IMAP expressions will be used. The format must start with a **user/** followed by a pattern. Table 43–2 shows the folder pattern for various folders.

Table 43–2 imexpire Folder Patterns Using Regular Expressions

Scope	Folder Patterns (regex=0)	Folder Pattern (regex=1)
Apply rule to all messages in all folders of *userid*.	**user/***userid*/*	**user/***userid*/.*
Apply rule to messages of *userid* in folder **Sent:**.	**user/***userid*/**Sent**	**user/***userid*/**Sent**
Apply rule to entire message store.	**user/***	**user/.***
Apply rule to any folder called **Trash** anywhere in any user's hierarchy.	**user/*/Trash**	**user/.*/Trash**
Apply rule to folders in hosted domain **example.org**.	**user/*@example.org/***	**user/.*@example.org/.***
Apply rule to folders in default domain.	Not applicable	**user/[^@]*/.***

44

Configuring Message Expiration (Tasks)

This chapter describes the tasks you use to expire messages. See "Message Store Message Expiration" for overview and conceptual information.

To Set imexpire Rules Textually

> **Note:** In Unified Configuration, you can continue to configure messaging expiration as explained in this information, or use the **msconfig** command to individually set the appropriate configuration options. Modifying the **store.expirerule** file enables access to more functionality than using the **msconfig** command.

You expire messages by specifying rules in a **store.expirerule** file. The **store.expirerule** file contains one expire criteria per line. An expire criteria of the global rule configuration file *MessagingServer_home***/config/store.expirerule** has the following format:

```
rule_name.attribute: value
```

An expiration rule for a user or mailbox rule configuration file has the following format:

```
attribute: value
```

The following example shows a set of global expiration rules in the *MessagingServer_home***/config/store.expirerule** file.

Example imexpire Rules

```
Rule1.regexp: 1
Rule1.folderpattern: user/.*
Rule1.messagesize: 100000
Rule1.messagesizedays: 3
Rule1.deleted: or
Rule1.messageheader.Subject: Vigara Now!
Rule1.messageheader.Subject: XXX Porn!
Rule1.messagecount: 1000
Rule1.messagedays: 365
Rule2.regexp: 1
Rule2.folderpattern: user/.*@example.org/.*
Rule2.exclusive: 1
Rule2.deleted: or
Rule2.messagedays: 14
Rule2.messagecount: 1000
```

```
Rule3.folderpattern: user/f.dostoevski/inbox
Rule3.messageheader.subject: On-line Casino
```

Rule 1 sets the global expiration policy (that is, policy that applies to all messages) to:

- Enable UNIX regular expressions in rules creation.
- Remove messages larger than 100,000 bytes after 3 days.
- Remove messages deleted by the user.
- Remove any message with the strings "Vigara Now!" or "XXX Porn!" in the Subject: header.
- Limit all folders to 1,000 messages. After 1,000 messages, the system removes the oldest messages on a folder to keep the total to 1,000.
- Remove all messages older than 365 days.

Rule 2 sets the message expiration policy for users at the hosted domain **example.org**. It limits mailbox sizes to 1 megabyte, removes messages that have been deleted, and removes messages older than 14 days.

Rule 3 sets the message expiration policy for messages in the **inbox** folder of user **f.dostoevski**. It removes messages with a subject line having the expression "On-line Casino."

Note that headers other than Subject: can be used.

To Set Expiration Rules by Using the msconfig Command

In Unified Configuration, you can set expiration rules by using the following **msconfig** command:

```
msconfig set expirerule:name.option value
```

For example:

```
msconfig set expirerule:Rule1.folderpattern user/.*
msconfig set expirerule:Rule1.messagesize 100000
msconfig set expirerule:Rule1.messagesizedays 3
```

You can set the following options in this way:

- **deleted**
- **exclusive**
- **folderpattern**
- **foldersizebytes**
- **messagecount**
- **messagedays**
- **messagesize**
- **messagesizedays**
- **seen**

To Set imexpire Folder Patterns

Folder patterns can be specified by using POSIX regular expressions by setting the **imexpire** attribute **regex** to 1. If not specified, IMAP expressions are used. The format

must start with a **user/** followed by a pattern. Table 44–1 shows the folder pattern for various folders.

Table 44–1 imexpire Folder Patterns Using Regular Expressions

Scope	Folder Pattern (regex=0)	Folder Pattern (regex=1)
Apply rule to all messages in all folders of *userid*.	**user/***userid*/*	**user/***userid*/.*
Apply rule to messages of *userid* in folder **Sent:**.	**user/***userid*/**Sent**	**user/***userid*/**Sent**
Apply rule to entire message store.	**user/***	**user/.***
Apply rule to any folder called **Trash** anywhere in any user's hierarchy.	**user/*/Trash**	**user/.*/Trash**
Apply rule to folders in hosted domain **example.org**.	**user/*@example.org/***	**user/.*@example.org/.***
Apply rule to folders in default domain.	Not applicable	**user/[Configuring Message Expiration in Unified Configuration (Tasks)^@]*/.***

To Schedule Message Expiration and Logging Level

You activate message expiration by using the **imsched** scheduling daemon. By default, **imsched** invokes **imexpire** at 23:00 every day. Use the "impurge" command for the purge function. The **imexpire** schedule can be customized by setting the **schedule.task:expire.crontab** option. See "Expire and Purge Log and Scheduling Options" for more information.

Expire and purge can take a long time to complete on a large message store. You should experiment and decide how often to run these processes. For example, if an expire and purge cycle takes 10 hours, you might not want the default schedule of running expire and purge once a day. Schedule expire and purge by using the **imexpire** command and the automatic task scheduling option (see "Scheduling Automatic Tasks"). For example:

```
msconfig
msconfig> set schedule.task:expire.crontab "0 1 * * 6 bin/imexpire -e"
msconfig# write
msconfig> exit
```

In this example, messages are expired at 1 AM Saturdays and purged every night at 11 PM. If no purge schedule is set, **imexpire** performs purge after an expire.

Expire and Purge Log and Scheduling Options

See "Maintenance Queue Configuration Options" for information about the Purge options. Clicking on the specific option will often provide both the legacy (**configutil**) parameter and the Unified Configuration (**msconfig**) option.

Table 44–2 Expire and Purge Log and Scheduling Options

Option	Description
schedule.task:expire.enable	Whether the expire task should be scheduled. Default: 1

Table 44–2 (Cont.) Expire and Purge Log and Scheduling Options

Option	Description
schedule.task:expire.crontab	Interval for running **imexpire**. Uses UNIX **crontab** format: *minute hour day-of-month month-of-year day-of-week* The values are separated by a space or tab and can be 0-59, 0-23, 1-31, 1-12 and 0-6 (with 0=Sunday) respectively. Each time field can be either an asterisk (meaning all legal values), a list of comma-separated values, or a range of two values separated by a hyphen. Note that days can be specified by both day of the month and day of the week, however, it is not typical to use them both since the number of such occurrences are very small. If they are both specified, then both will be required. For example, setting the 17th day of the month and Tuesday will require both values to be true. You can also use the **-e** and **-c** flags with **imexpire** to and expire only or purge only respectively. See "imexpire". **Interval Examples:** 1. Run **imexpire** at 12:30am, 8:30am, and 4:30pm:**30 0,8,16 * * * bin/imexpire** 2. Run **imexpire** at weekday morning at 3:15 am:**15 3 * * 1-5 bin/imexpire** 3. Run **imexpire** only on Mondays:**0 0 * * 1 bin/imexpire** Default: **0 23 * * * bin/imexpire** To disable: Run **msconfig set schedule.task:expire.enable 0**
store.expire.exploglevel	Specify a log level: 0 = No log. 1 = Log summary for the entire expire session. 2 = Log one message per mailbox expired. 3 = Log one message per message expired. Default: 0

To Set imexpire Logging Levels

imexpire logs a summary to the default log file upon completion. If expire is invoked from the command line, the *-v* (verbose) and **-d** (debug) options can be used to instruct **imexpire** to log detail status and debug messages to **stderr**. If **imexpire** is invoked by **imsched**, the configuration option **store.expire.exploglevel** can be set to 0, 1, 2, or 3 for different levels of logging. Loglevel 0 is the default, and no logging is performed. Loglevel 1 logs a summary for the entire expire session. Loglevel 2 logs one message per mailbox expired. Loglevel 3 logs one message per message expired.

The following example invokes expire from the command line to set verbose logging and shows the resulting messages in the default log file, *DataRoot***/log/default**.

```
cd /opt/sun/comms/messaging64/bin
imexpire -n -v 1
tail ../log/default
[25/Nov/2010:11:51:51 +1100] server imexpire[20730]: General Notice: imexpire
started
[25/Nov/2010:11:51:51 +1100] server imexpire[20730]: General Notice: iBiff plugin
loaded: ms-internal
[25/Nov/2010:11:51:51 +1100] server imexpire[20730]: General Notice: primary
partition: expired 0 messages
[25/Nov/2010:11:51:51 +1100] server imexpire[20730]: General Notice: Expired 0
```

```
messages
[25/Nov/2010:11:51:51 +1100] server imexpire[20730]: General Notice: Expire
finished
```

To Exclude Specified Users from Message Expiration

Exclude specified users from the expire rules by adding their user ID, one per line, in a file called **expire_exclude_list** in the *MessagingServer_home*/**config** directory. Or, configure a "dummy" exclusive expire rule under the user's mailbox.

45

Configuring POP, IMAP, and HTTP Services

This chapter describes how to configure your server to support one or more of the POP, IMAP, or HTTP services by using command-line utilities.

Oracle Communications Messaging Server supports the Post Office Protocol 3 (POP3), the Internet Mail Access Protocol 4 (IMAP4), and the HyperText Transfer Protocol (HTTP) for client access to mailboxes. IMAP and POP are both Internet-standard mailbox protocols. Convergence, a web-enabled electronic mail program, enables end users to access their mailboxes by using a browser running on an Internet-connected computer system using HTTP.

General Configuration

Configuring the general features of the Messaging Server POP, IMAP, and HTTP services includes enabling or disabling the services, assigning port numbers, and optionally modifying service banners sent to connecting clients. This section provides background information about configuring these services.

Enabling and Disabling Services

You can control whether any particular instance of Messaging Server makes its POP, IMAP, or HTTP service available for use. This is not the same as starting and stopping services (see "Stopping and Starting Messaging Server"). To function, POP, IMAP, or HTTP must be both enabled and started.

Enabling a service is a more "global" process than starting or stopping a service. For example, the Enable setting persists across system reboots, whereas you must restart a previously "stopped" service after a reboot.

There is no need to enable services that you do not plan to use. For example, if a Messaging Server instance is used only as a Mail Transfer Agent (MTA), you should disable POP, IMAP, and HTTP services. If a Messaging Server instance is used only for POP services, you should disable IMAP and HTTP. If a Messaging Server instance is used only for web-based email, you should disable both POP and IMAP.

You can enable or disable services at the server level, described later in this information. "To Specify What Services Can Be Started" also describes this process. In addition, you can enable or disable services at the user level by setting the LDAP attribute **mailAllowedServiceAccess** seen in *Schema Reference*.

Specifying Port Numbers

For each service, you can specify the port number that the server is to use for service connections:

- If you enable the POP service, you can specify the port number that the server is to use for POP connections. The default is 110.
- If you enable the IMAP service, you can specify the port number that the server is to use for IMAP connections. The default is 143.
- If you enable the HTTP service, you can specify the port number that the server is to use for HTTP connections. The default is 8990.

You might need to specify a port number other than the default if you have, for example, two or more IMAP server instances on a single host machine, or if you are using the same host machine as both an IMAP server and a Messaging Multiplexor server. See "Configuring and Administering Multiplexor Services" for information about the Multiplexor.

Keep the following in mind when you specify a port:

- Port numbers can be any number from 1 to 65535.
- Make sure the port you choose isn't already in use or reserved for another service.

Ports for Encrypted Communications

Messaging Server supports encrypted communications with IMAP, POP, and HTTP clients by using the Secure Sockets Layer (SSL) protocol. For general information on support for SSL in Messaging Server, see the discussion on configuring encryption and certificate-based authentication in *Messaging Server Security Guide*.

IMAP Over SSL

You can accept the default (recommended) IMAP over SSL port number (993) or you can specify a different port for IMAP over SSL.

Messaging Server provides the option of using separate ports for IMAP and IMAP over SSL because most current IMAP clients require separate ports for them. Same-port communication with both IMAP and IMAP over SSL is an emerging standard. As long as your Messaging Server has an installed SSL certificate (see the discussion on obtaining certificates in *Messaging Server Security Guide*), it can support same-port IMAP over SSL.

POP Over SSL

The default separate SSL port for POP is 995. You can also initiate SSL over normal POP port with the command "STLS" (see "To Configure POP Services").

HTTP Over SSL

You can accept the default HTTP over SSL port number (8991) or you can specify a different port for HTTPS.

Service Banner

When a client first connects to the Messaging Server POP or IMAP port, the server sends an identifying text string to the client. This service banner (not normally displayed to the client's user) identifies the server as Messaging Server, and gives the server's version number. The banner is most typically used for client debugging or problem-isolation purposes.

You can replace the default banner for the POP or IMAP service if you want a different message sent to connecting clients.

Use the **msconfig** utility and the (**pop.banner**) option to set service banners.

Login Requirements

You can control how users are permitted to log in to the POP, IMAP, or HTTP service to retrieve mail. You can allow password-based login (for all services), and certificate-based login (for IMAP or HTTP services). This section provides background information. See the following sections for information about configuring these settings:

- To Configure POP Services
- To Configure IMAP Services
- To Configure the mshttpd Process for Use by Convergence

In addition, you can specify the valid login separator for POP logins.

To Set the Separator for POP Clients

Some older mail clients do no accept @ as the login separator (that is, the @ in an address like **uid@domain**). If you are using one of these older mail clients, the workaround is as follows:

1. Make **+** a valid separator with the following command:

   ```
   msconfig set base.loginseparator "@+"
   ```

2. Inform POP client users that they should log in with **+** as the login separator, not **@**.

To Allow Log In without Using the Domain Name

A typical login involves the user entering a user ID followed by a separator and the domain name and then the password. Users in the default domain specified during installation, however, can log in without entering a domain name or separator.

To allow users of other domains to log in with just the user ID (that is, without having to use the domain name and separator) set the **auth.searchfordomain** option to ***0***. The user ID must be unique to the entire directory tree. If it is not unique, logging in without the domain name will not work.

You might want to modify the attribute that user must enter to log in. For example, to allow the user to log in with a phone number (**telephoneNumber**) or employee number (**employeeID**), change the LDAP search defined by the **auth.searchfilter** option. This option is a global default setting for the **inetDomainSearchFilter** per-domain attribute and follows the same syntax.

Refer to *Messaging Server Reference* for further information on these options.

Password-Based Login

In typical messaging installations, users access their mailboxes by entering a password into their POP, IMAP, or HTTP mail client. The client sends the password to the server, which uses it to authenticate the user. If the user is authenticated, the server decides, based on access-control rules, whether or not to grant the user access to certain mailboxes stored on that server.

If you allow password login, users can access POP, IMAP, or HTTP by entering a password. (Password- or SSL-based login is the only authentication method for POP

services.) Passwords are stored in an LDAP directory. Directory policies determine what password policies, such as minimum length, are in effect.

If you disallow password login for IMAP or HTTP services, password-based authentication is not permitted. Users are then required to use certificate-based login, as described in the next section.

To increase the security of password transmission for IMAP and HTTP services, you can require that passwords be encrypted before they are sent to your server. You do this by selecting a minimum cipher-length requirement for login.

- If you choose 0, you do not require encryption. Passwords are sent in the clear or they are encrypted, depending on client policy.
- If you choose a nonzero value, the client must establish an SSL session with the server by using a cipher whose key length is at least the value you specify, thus encrypting any IMAP or HTTP user passwords the client sends.

If the client is configured to require encryption with key lengths greater than the maximum your server supports, or if your server is configured to require encryption with key lengths greater than what the client supports, password-based login cannot occur. For information on setting up your server to support various ciphers and key lengths, see the discussion on enabling SSL and selecting ciphers in *Messaging Server Security Guide*.

Certificate-Based Login

In addition to password-based authentication, Oracle servers support the authentication of users through examination of their digital certificates. Instead of presenting a password, the client presents the user's certificate when it establishes an SSL session with the server. If the certificate is validated, the user is considered authenticated.

For instructions on setting up Messaging Server to accept certificate-based user login to the IMAP or HTTP service, see the discussion on setting up certificate-based login in *Messaging Server Security Guide*.

If you have performed the tasks required to set up certificate-based login, both password-based and certificate-based login are supported. Then, if the client establishes an SSL session and supplies a certificate, certificate-based login is used. If the client does not use SSL or does not present a client certificate, it sends a password instead.

Performance Options

You can set some of the basic performance options for the POP, IMAP, and HTTP services of Messaging Server. Based on your hardware capacity and your user base, you can adjust these options for maximum efficiency of service. This section provides background information. See the following sections for the steps you follow to make these settings:

- To Configure POP Services
- To Configure IMAP Services
- To Configure the mshttpd Process for Use by Convergence

Number of Processes

Messaging Server can divide its work among several executing processes, which in some cases can increase efficiency. This capability is especially useful with multiprocessor server machines, in which adjusting the number of server processes can allow more efficient distribution of multiple tasks among the hardware processors.

There is a performance overhead, however, in allocating tasks among multiple processes and in switching from one process to another. The advantage of having multiple processes diminishes with each new one added. A simple rule of thumb for most configurations is to have one **IMAPD** and one **POPD** process per hardware processor on your server machine, up to a maximum of perhaps four processes. Your optimum configuration might be different. This rule of thumb is meant only as a starting point for your own analysis.

Note: On some platforms you might also want to increase the number of processes to get around certain per-process limits (such as the maximum number of file descriptors), specific to that platform, that might affect performance. The default number of processes is one each for the POP, IMAP, or HTTP service.

Number of Connections per Process

The more simultaneous client connections your POP, IMAP, or HTTP service can maintain, the better it is for clients. If clients are denied service because no connections are available, they must then wait until another client disconnects.

On the other hand, each open connection consumes memory resources and makes demands on the I/O subsystem of your server machine, so there is a practical limit to the number of simultaneous sessions you can expect the server to support. (You might be able to increase that limit by increasing server memory or I/O capacity.)

IMAP, HTTP, and POP have different needs in this regard:

- IMAP connections are generally long-lived compared to POP and HTTP connections. When a user connects to IMAP to download messages, the connection is usually maintained until the user quits or the connection times out. In contrast, a POP or HTTP connection is usually closed as soon as the POP or HTTP request has been serviced.
- IMAP and HTTP connections are generally very efficient compared to POP connections. Each POP re-connection requires re-authentication of the user. In contrast, an IMAP connection requires only a single authentication because the connection remains open for the duration of the IMAP session (login to logout). An HTTP connection is short, but the user need not re-authenticate for each connection because multiple connections are allowed for each HTTP session (login to logout). POP connections, therefore, involve much greater performance overhead than IMAP or HTTP connections. Messaging Server, in particular, has been designed to require very low overhead by open but idle IMAP connections and by multiple HTTP connections.

Note: For more information about HTTP session security, see the discussion about HTTP security in *Messaging Server Security Guide*.

Thus, at a given moment for a given user demand, Messaging Server may be able to support many more open IMAP or HTTP connections than POP connections.

The default value for IMAP is 4000. The default value for HTTP is 6000 connections per process. The default value for POP is 600. These values represent roughly equivalent demands that can be handled by a typically configured server machine. Your optimum configuration might be different. These defaults are meant only as general guidelines.

Typically, active POP connections are much more demanding on server resources and bandwidth than active IMAP connections since IMAP connections are idle most of the time while POP connections are constantly downloading messages. Having a lower number of sessions for POP is correct. Conversely, POP connections only last as long as it takes to download email, so an active POP user is only connected a small percentage of the time, while IMAP connections stay connected between successive mail checks.

Number of Threads per Process

Besides supporting multiple processes, Messaging Server further improves performance by subdividing its work among multiple threads. The server's use of threads greatly increases execution efficiency, because commands in progress are not holding up the execution of other commands. Threads are created and destroyed, as needed during execution, up to the maximum number you have set.

Having more simultaneously executing threads means that more client requests can be handled without delay, so that a greater number of clients can be serviced quickly. However, there is a performance overhead to dispatching among threads, so there is a practical limit to the number of threads the server can use.

For POP, IMAP, and HTTP, the default maximum value is 250 threads per process. The numbers are equal despite the fact that the default number of connections for IMAP and HTTP is greater than for POP. It is assumed that the more numerous IMAP and HTTP connections can be handled efficiently with the same maximum number of threads as the fewer, but busier, POP connections. Your optimum configuration might be different, but these defaults are high enough that it is unlikely you would ever need to increase them; the defaults should provide reasonable performance for most installations.

Dropping Idle Connections

To reclaim system resources used by connections from unresponsive clients, the IMAP4, POP3, and HTTP protocols permit the server to unilaterally drop connections that have been idle for a certain amount of time.

The respective protocol specifications require the server to keep an idle connection open for a minimum amount of time. The default times are 10 minutes for POP, 30 minutes for IMAP, 3 minutes for HTTP. You can increase the idle times beyond the default values, but you cannot make them less.

If a POP or IMAP connection is dropped, the user must re-authenticate to establish a new connection. In contrast, if an HTTP connection is dropped, the user need not re-authenticate because the HTTP session remains open. For more information about HTTP session security, see the discussion about HTTP security in *Messaging Server Security Guide*.

Idle POP connections are usually caused by some problem (such as a crash or hang) that makes the client unresponsive. Idle IMAP connections, on the other hand, are a normal occurrence. To keep IMAP users from being disconnected unilaterally, IMAP clients typically send a command to the IMAP server at some regular interval that is less than 30 minutes.

Logging Out HTTP Clients

An HTTP session can persist across multiple connections. HTTP clients are not logged out when a connection is dropped. However, if an HTTP session remains idle for a specified time period, the server will automatically drop the HTTP session and the client is logged out (the default time period is 2 hours). When the session is dropped, the client's session ID becomes invalid and the client must re-authenticate to establish another session. For more information about HTTP security and session ID's, see the discussion about HTTP security in *Messaging Server Security Guide*.

Client Access Controls

Messaging Server includes access-control features that enable you to determine which clients can gain access to its POP, IMAP, or HTTP messaging services (and SMTP as well). You can create flexible access filters that allow or deny access to clients based on a variety of criteria.

Client access control is an important security feature of Messaging Server. For information on creating client access-control filters and examples of their use, see the discussion on configuring client access to POP, IMAP, and HTTP services in *Messaging Server Security Guide*.

To Configure POP Services

You configure the Messaging Server POP service by using the **msconfig** command. This section lists the more common POP services options. The *Messaging Server Reference* provides a complete listing of options.

> **Note:** For the POP service, password-based login is automatically enabled.

For more information, see also:

- Enabling and Disabling Services
- To Set the Separator for POP Clients
- Specifying Port Numbers
- Number of Connections per Process
- Dropping Idle Connections
- Number of Threads per Process
- Number of Processes

- To enable the POP service:

```
msconfig set pop.enable 1
```

- To disable the POP service:

```
msconfig set pop.enable 0
```

- To specify the port number:

```
msconfig set pop.port port_number
```

- To set the maximum number of network connections per process (see "Number of Connections per Process" for details):

  ```
  msconfig set pop.maxsessions number
  ```

- To set the maximum idle time for connections (see "Dropping Idle Connections" for details):

  ```
  msconfig set pop.idletimeout number
  ```

- To set the maximum number of threads per process (see "Number of Threads per Process" for more information):

  ```
  msconfig set pop.maxthreads number
  ```

- To set the maximum number of processes (see "Number of Processes" for additional information):

  ```
  msconfig set pop.numprocesses number
  ```

- To enable POP over SSL on port 995:

  ```
  msconfig
  msconfig> set pop.enablesslport 1
  msconfig# set pop.sslusessl 1
  msconfig# set pop.sslport 995
  msconfig# write
  msconfig> exit
  stop-msg pop
  start-msg pop
  ```

  TLS is also supported if SSL is configured correctly.

- To specify a protocol welcome banner:

  ```
  msconfig set pop.banner banner
  ```

To Configure IMAP Services

You configure the Messaging Server IMAP service by using the **msconfig** command. This section lists the common IMAP services options. *Messaging Server Reference* provides a complete listing of options. For more information, see also:

- Enabling and Disabling Services
- Specifying Port Numbers
- Password-Based Login
- Number of Connections per Process
- Dropping Idle Connections
- Number of Threads per Process
- Number of Processes
- Configuring IMAP IDLE

- To enable the IMAP service:

  ```
  msconfig set imap.enable 1
  ```

- To disable the IMAP service:

  ```
  msconfig set imap.enable 0
  ```

- To specify the port number:

  ```
  msconfig set imap.port number
  ```

- To enable a separate port for IMAP over SSL:

  ```
  msconfig set imap.enablesslport 1
  ```

- To specify a port number for IMAP over SSL:

  ```
  msconfig set imap.sslport number
  ```

- To enable or disable password login to the IMAP service:

  ```
  msconfig set imap.plaintextmincipher value
  ```

  If *value* is greater than 0, disable use of plaintext passwords unless a security layer (SSL or TLS) is activated. This forces users to enable SSL or TLS on their client to log in, which prevents exposure of their passwords on the network. Default is 0.

- To set the maximum number of network connections per process (see "Number of Connections per Process" for additional information):

  ```
  msconfig set imap.maxsessions number
  ```

- To set the maximum idle time for connections (see "Dropping Idle Connections" for additional information):

  ```
  msconfig set imap.idletimeout number
  ```

- To set the maximum number of threads per process (see "Number of Threads per Process"):

  ```
  msconfig set imap.maxthreads number
  ```

- To set the maximum number of processes (see "Number of Processes"):

  ```
  msconfig set imap.numprocesses number
  ```

- To specify a protocol welcome banner:

  ```
  msconfig set imap.banner banner
  ```

- To enable IMAP over SSL on port 993:

  ```
  msconfig
  msconfig> set imap.enablesslport 1
  msconfig# set imap.sslusessl 1
  msconfig# set imap.sslport 993
  msconfig# write
  msconfig> exit
  stop-msg imap
  start-msg imap
  ```

Configuring IMAP IDLE

The IMAP IDLE extension to the IMAP specification, defined in RFC 2177, enables an IMAP server to notify the mail client when new messages arrive and other updates take place in a user's mailbox. See "Configuring IMAP IDLE" for conceptual and task information on enabling IMAP IDLE in Messaging Server.

To Configure the mshttpd Process for Use by Convergence

Messaging Server supports the mail client Convergence.

While POP and IMAP clients send mail directly to a Messaging Server MTA for routing or delivery, HTTP clients send mail to a specialized web server called the Webmail Server (also called **mshttpd** or Messaging Server HTTP daemon). Depending on where the message is addressed, the Webmail Server directs the mail to an outbound MTA for routing or to one of the back-end message stores using IMAP. Convergence simply routes requests to and from the Webmail Server.

The Webmail Server accesses the message store through the IMAP server. This provides several advantages:

- Convergence clients are able to access shared folders that are located on different back-end message stores.
- The Webmail Server does not need to be installed on each back-end server.
- The Webmail Server can serve as a front-end server performing multiplexing capabilities.
- Users can access shared folders that are not on their message store.

The Webmail Server operates as a front-end server receiving HTTP client email requests. It translates these requests to SMTP or IMAP calls and forwards the calls to either the MTA or the appropriate IMAP server on the back-end message store. If Messaging Server is used only for web-based email, make sure that IMAP is enabled.

Configuring Your HTTP Service

Many of the HTTP configuration options are similar to the options available for the POP and IMAP services, including options for connection settings and process settings. This section lists common HTTP service options. *Messaging Server Reference* provides a complete listing of options. For more information, see also:

- Enabling and Disabling Services
- Specifying Port Numbers
- Password-Based Login
- Number of Connections per Process
- Dropping Idle Connections
- Logging Out HTTP Clients
- Number of Threads per Process
- Number of Processes

For each IMAP server that users access, the Webmail Server needs to know the IMAP port, whether to use SSL, and the administrative credentials for user log-in. The configuration options to do this are as follows:

- **base.proxyimapport**: IMAP port on which to connect (default 143).
- **base.proxyimapssl**: Enable SSL (default no).
- **base.proxyadmin**: Specifies the store Admin ID.
- **base.proxyadminpass**: Specifies the store Admin password.

You can set these options globally in Unified Configuration to apply to every IMAP back-end server by using **base.proxyadmin**. Alternatively, you can set these options

for each individual IMAP back-end server by using **proxy:***storeaffinitygroup***.imapadmin**.

To use IMAP over SSL, you must configure **mshttpd** as an SSL HTTP server, and the **mshttpd** certificate database must trust the IMAP back end's CA. You must enable **http.sslusessl**. If the back-end message store running IMAP is using a self-signed certificate (for example, as created by **generate-certDB**), then this certificate needs to be added to the front-end **mshttpd** daemon server.

If **base.proxyadmin** and **base.proxyadminpass** are not configured, logins are rejected. The system provides the error message, "**Mail server unavailable. Administrator, check server log for details**" and the HTTP log lists the missing configuration options.

Additional values for HTTP attributes can be set at the command line as follows:

- To enable the HTTP service:

  ```
  msconfig set http.enable 1
  ```

- To disable the HTTP service:

  ```
  msconfig set http.enable 0
  ```

By default, the HTTP service sends outgoing web mail to the local MTA for routing or delivery. You might want to configure the HTTP service to send mail to a remote MTA, for example, if your site is a hosting service and most recipients are not in the same domain as the local host machine. To send web mail to a remote MTA, you need to specify the remote host name and the SMTP port number for the remote host.

- To specify the port number:

  ```
  msconfig set http.port number
  ```

- To enable a separate port for HTTP over SSL:

  ```
  msconfig set http.enablesslport 1
  ```

- To specify a port number for HTTP over SSL:

  ```
  msconfig set http.sslport number
  ```

- To enable or disable password login:

  ```
  msconfig set http.plaintextmincipher value
  ```

  If *value* is greater than 0, then disable use of plaintext passwords unless a security layer (SSL or TLS) is activated. This forces users to enable SSL or TLS on their client to log in, which prevents exposure of their passwords on the network. Default is 0.

- To set the maximum number of network connections per process (for more information, see "Number of Connections per Process"):

  ```
  msconfig set http.maxsessions number
  ```

  See "Dropping Idle Connections" for more information.

- To set the maximum idle time for client sessions (for more information, see "Logging Out HTTP Clients"):

  ```
  msconfig set http.sessiontimeout number
  ```

- To set the maximum number of threads per process:

  ```
  msconfig set http.maxthreads number
  ```

- To set the maximum number of processes:

```
msconfig set http.numprocesses number
```

When an HTTP client constructs a message with attachments, the attachments are uploaded to the server and stored in a file. The HTTP service retrieves the attachments and constructs the message before sending the message to an MTA for routing or delivery. You can accept the default attachment spool directory or specify an alternate directory. You can also specify a maximum size allowed for attachments. To specify the attachment spool directory for client outgoing mail use the following command. This includes all the attachments encoded in base64, and that base64 encoding requires an extra 33 percent more space. Thus, a 5 Mb limit in the option results in the maximum size of one message and attachments being about 3.75 Mb.

- To set the spool directory:

```
msconfig set http.spooldir dirpath
```

- To specify the maximum message size:

```
msconfig set http.maxmessagesize size
```

where *size* is a number in bytes. This includes all the attachments encoded in base64, and that base64 encoding requires an extra 33 percent more space. Thus, a 5 Mb limit in the option results in the maximum size of one message and attachments being about 3.75 Mbs.

- To specify an alternate MTA host name:

```
msconfig set http.smtphost hostname
```

- To specify the port number for the alternate MTA host name:

```
msconfig set http.smtpport portnum
```

To enable HTTP access over SSL on port 8991:

```
msconfig
msconfig> set http.enablesslport -1
msconfig# set http.sslusessl 1
msconfig# set http.sslport 8991
msconfig# write
msconfig> exit
stop-msg http
start-msg http
```

46

Handling Message Store Overload

This chapter describes how to handle message store overload in Oracle Communications Messaging Server.

Overview of Managing Message Store Load

An overloaded message store can suffer from degraded performance. The **mboxlist** database is particularly sensitive to overload conditions. When the database detects deadlocks, all database operations that cannot acquire the locks they need must abort the transactions and retry, thereby decreasing the throughput. If this situation continues, the message store can become very inefficient. In extreme cases, you need to restart the message store to recover.

Therefore, having the ability to control the message store load is crucial to prevent performance degradation. The message store uses transaction checkpoint time as the stress indicator. The **stored** daemon measures the transaction checkpoint duration (the time it takes to sync the database pages from the memory pool to disks). When the transaction checkpoint exceeds one minute, it raises an alarm.

Message Store Load Throttling

Message store throttling is used to regulate short spikes of activities. When the **ims_master** program detects the stressed status from the message store, it informs the Job Controller. The Job Controller responds by temporarily decreasing the number of **ims_master** processes for the **ims-ms** channel. Similarly, when the LMTP server detects the stressed status, it tells the LMTP client, which informs the Job Controller, to back off. By decreasing the number of delivery threads, the Job Controller enables the message store to recover before performance begins to degrade.

Job Controller Stress Handling

Channel programs can now tell the Job Controller if they are being overwhelmed. If this occurs, then the job controller sees if it has happened recently. The job controller ignores stressed channel messages that are received within **job_controller.stressblackout** seconds of a previous stressed message for the same channel. If the message is processed, then the job controller multiplies the effective **threaddepth** option for the channel by **job_controller.stressfactor**, and subtracts **job_controller.stressjob** from the job limit for the channel. **threaddepth** never goes over 134,217,727, and job limit never goes below 1. In addition, then the Job Controller asks all current master programs for the channel to exit, and, if the queue is not empty, starts an appropriate number of processes.

When **job_controller.stresstime** seconds has passed after the last stress change, the Job Controller divides **threaddepth** by **job_controller.unstressfactor** (never allowing thread depth to drop below the original configured **threaddepth**), and adds **UnstressJob** to the job limit (never allowing the job limit to rise above the original configured limit. a "stress change" is either an increase in stress or a decrease in stress.

The **unstresscount** job controller option adds an additional criteria for lowering the stress level for a channel. The level is also lowered when *unstresscount* messages have been processed by the channel and *stresstime* time has elapsed without any indication of stress.

Default Job Controller Configuration

These configuration options have the following default values:

job_controller.stressblackout=60

job_controller.stresstime=120

job_controller.stressfactor=5

job_controller.stressjobs=2

job_controller.unstressfactor=stressfactor

job_controller.unstressjobs=stressjobs

job_controller.unstresscount=10000

47

Managing Message Store Partitions and Adding Storage

This chapter describes Oracle Communications Messaging Server classic message store partitions and adding storage.

For more information, see also:

- To Move Mailboxes to a Different Disk Partition

> **Note:** Cassandra message store does not use partitions. To add more space to the message store, see *Messaging Server Installation and Configuration Guide for Cassandra Message Store*.

Message Store Partition Overview

Mailboxes are stored in message store partitions, an area on a disk partition specifically devoted to storing the message store. Message store partitions are not the same as disk partitions, though for ease of maintenance, it is recommended that you have one disk partition and one file system for each message store partition. Message store partitions are directories specifically designated as a message store.

User mailboxes are stored by default in the *store_root*/**partition/** directory (see "Classic Message Store Directory Layout"). The **partition** directory is a logical directory that might contain a single partition or multiple partitions. At start-up time, the **partition** directory contains one subpartition called the **primary** partition.

You can add partitions to the **partition** directory as necessary. For example, you might want to partition a single disk to organize your users as follows:

```
store_root/partition/mkting/
store_root/partition/eng/
store_root/partition/sales/
```

As disk storage requirements increase, you might want to map these partitions to different physical disk drives.

You should limit the number of mailboxes on any one disk. Distributing mailboxes across disks improves message delivery time (although it does not necessarily change the SMTP accept rate). The number of mailboxes you allocate per disk depends on the disk capacity and the amount of disk space allocated to each user. For example, you can allocate more mailboxes per disk if you allocate less disk space per user.

If your message store requires multiple disks, you can use RAID (Redundant Array of Inexpensive Disks) technology to ease management of multiple disks. With RAID technology, you can spread data across a series of disks but the disks appear as one

logical volume so disk management is simplified. You might also want to use RAID technology for redundancy purposes; that is, to duplicate the store for failure recovery purposes.

> **Note:** To improve disk access, the message store and the message queue should reside on separate disks.

To Add a Message Store Partition

When adding a partition, you specify both an absolute physical path where the partition is stored on disk, and a logical name (called the partition nickname).

The partition nickname enables you to map users to a logical partition name regardless of the physical path. When setting up user accounts and specifying the message store for a user, you can use the partition nickname. The name you enter must be an alphanumeric name and must use lowercase letters.

To create and manage the partition, the user ID used to run the server must have permission to write to the location specified in the physical path.

> **Note:** After adding a partition, you must stop then restart Messaging Server to refresh the configuration information.

- **Command Line,** To add a partition to the store at the command line: The location of message files is controlled by setting the **partition:***partition_name***.messagepath** option, where *partition_name* is the logical name of the partition.

  ```
  msconfig set partition:partition_name.messagepath path
  ```

  path indicates the absolute path name where the partition is stored. To specify the path to the primary partition:

  ```
  msconfig set partition:primary.path path
  ```

To Change the Default Message Store Partition

The default partition is the partition used when a user is created and the **mailMessageStore** LDAP attribute is not specified in the user entry. The **mailMessageStore** LDAP attribute, which specifies a user's message store partition, should be specified in all user entries so that a default partition is not necessary. In addition, the default partition should **not** be changed for load balancing or any other reason. It is invalid and dangerous to change the default partition while there are still users depending on the default partition definition.

If it is absolutely necessary to change the default partition, make sure that all users on the old default partition (the one being left behind) have their **mailMessageStore** attribute set to their current partition (which will no longer be the default), before changing the definition of default with the **store.defaultpartition** option.

Adding More Physical Disks to the Message Store

The Messaging Server message store contains the user mailboxes for a particular Messaging Server instance. The size of the message store increases as the number of mailboxes, folders, and log files increase.

As you add more users to your system, your disk storage requirements increase. Depending on the number of users your server supports, the message store might require one physical disk or multiple physical disks. Messaging Server enables you an add more stores as needed.

48

Managing Message Store Quotas

This chapter describes managing Oracle Communications Messaging Server message store quotas and the quota tasks.

For more information, see also "Monitoring the Message Store".

Message Store Quota Overview

Message store quotas limit or reduce message store usage. They enable you to set *quotas* for how much disk space or how many messages can be used by a user or domain. See "Managing Message Store Quotas" for information on how to use quotas in your system.

Quota Overview

Quotas can be set, in terms of number of messages or number of bytes or both, for specific users or domains. Quotas can also be set for specific folders and message types. For example, you can set different quotas based on whether a message is a voice mail or an email. Folder quotas set limits to the size of a user's folder in bytes or number of messages. For example, a quota can be set on the Trash folder. Messaging Server enables you to set default quotas for domains and users as well as customized quotas.

You can also configure how the system responds to users or domains that are either over quota or approaching the quota. One response is to send users an *over quota notification*. Another response is to halt delivery of messages into the message store when quota is exceeded. This is called *quota enforcement* and usually occurs after a specified *grace period*. A grace period is how long the mailbox can be over the quota before enforcement occurs. If message delivery is halted due to over quota, incoming messages can either remain in the MTA queue until one of the following occurs or be rejected by my the MTA immediately, if **local.store.overquotastatus** is enabled:

- The size or number of the user's messages no longer exceeds the quota, at which time the MTA delivers the messages.
- The undelivered message remains in the MTA queue longer than the specified *grace period*, at which time messages are returned to sender. See "To Set a Grace Period" for more information.
- The message has remained in the message queue longer than the maximum message queue time. This is controlled by the **notices** MTA channel keyword (see the discussion on setting notification message delivery intervals in *Messaging Server Reference*).

For example, if your grace period is set for two days, and you exceed quota for one day, new messages continue to be received and held in the message queue, and delivery attempts continue. After the second day, the messages bounce back to the sender.

Disk space becomes available when a user deletes and expunges messages or when the server deletes messages according to expiration policies established. See "Message Store Message Expiration" for more information.

Exceptions for Telephony Application Servers

To support unified messaging requirements, Messaging Server provides the ability to override quota limitations imposed by the message store. This guarantees the delivery of messages that have been accepted by certain agents, namely telephony application servers (TAS). Messages accepted by a TAS can be routed through a special MTA channel that ensures the message is delivered to the store regardless of quota limits. This is a fairly esoteric usage, but can be useful to telephony applications. For more information about configuring a TAS channel, contact your Oracle messaging representative.

Quota by message type is useful for telephony applications that use unified messaging. For example, if a mix of messages, say text and voice mail, is stored in a user's mailbox, then the administrator can set different quotas for different types of messages. One quota can be set for email and another can be set for voice mail.

Quota Theory of Operations

Customized user and domain quotas are specified by adding quota attributes to LDAP user and domain entries. Quota defaults, notification policy, enforcement, and grace period are specified in **msconfig** options or by using the "imquotacheck" command.

To determine if a user is over quota, Messaging Server first determines if a quota has been set for the individual user. If no quota has been set, Messaging Server looks at the default quota set for all users. For a user, the quota is for all the cumulative bytes or messages in all of the user's folders. For a domain, the quota is for all the cumulative bytes or messages of all the users in a particular domain. For a message type, the quota is for all the cumulative bytes or messages for that message type. For a folder, the quota is for all the cumulative bytes or messages for user's folder.

You can specify the following quota values for a user's mailbox tree:

- Quota values for specific folders in the user's mailbox.
- Quota values for specific message types such as voice mail or text messages. (A message type quota applies to messages of that type in all folders in the user's mailbox.)
- A default quota value that applies to all folders and message types in the user's mailbox that are not explicitly assigned quotas.

The following guidelines apply when you assign multiple quota values for a user:

- Quotas do not overlap. For example, when there is a quota for a particular message type or folder, messages of that type or messages in that folder are not counted toward the default quota. Each message counts toward one and only one quota.
- The total quota for the whole user mailbox equals the sum of the values of all the quotas specified by default, type, and folder.

- Message type quotas take precedence over folder quotas. For example, suppose one quota is specified for a user's **memos** folder and another quota is specified for voice messages. Now suppose the user stores eight voice messages in the **memos** folder. The eight messages are counted toward the voice mail quota and excluded from the **memos** folder quota.

Changes made to the quota attributes and **msconfig** options will take effect automatically, but not immediately as information is stored in caches and it may take a little time before the changes fully take effect. Messaging Server provides a command, "iminitquota" that updates the changes immediately.

The "imquotacheck" utility enables you to check message store usage against assigned quotas.

Methods of Notification

If **store.quotanotification** is enabled, when users approach or exceed their quota limit (depending on **store.quotawarn**), the message defined by **store.quotaexceededmsg** notifies them immediately. Otherwise, you must run **imquotacheck -n** to notify the users. Dynamic user notification by enabling **store.quotanotification** and running **imquotacheck -n** are mutually exclusive. If **store.quotanotification** is enabled, you should not use **imquotacheck -n**. The preferred method is dynamic notification.

Domain level quota enforcement and reporting is done by running **imquotacheck -f**.

For more information on **imquotacheck** options, see Table 64–14, " imquotacheck Options".

Message Store Quota Attributes and Options

This section lists the major the message store quota attributes and **msconfig** options. The intention is to provide you with an overview of the functionality interface. For detailed information on these attributes and options, refer to the appropriate reference documentation.

Table 48–1 lists the quota attributes. Refer to *Schema Reference* for more information about the following attributes.

Table 48–1 Message Store Quota Attributes

Attribute	Description
mailQuota	Bytes of disk space allowed for the user's mailbox.
mailMsgQuota	Maximum number of messages permitted for a user. This is a cumulative count for all folders in the store.
mailUserStatus	Status of the mail user. Some of the possible values are **active**, **inactive**, **deleted**, **hold**, and **overquota**.
mailDomainDiskQuota	Bytes of disk space allowed for the cumulative count of all the mailboxes in a domain.
mailDomainMsgQuota	Maximum number of messages permitted for a domain, that is, the total count for all mailboxes in the store.
mailDomainStatus	Status of the mail domain. Values and default are the same as **mailUserStatus**.

To have the preceding attributes take effect, run "iminitquota" to make quota and usage up-to-date. The changes would take effect without running this, but not immediately, as information is stored in caches and it takes a little time before the changes take effect.

Table 48–2 lists the quota options. Refer to "Overview of Messaging Server Unified Configuration" for the latest and most detailed information.

Table 48–2 Message Store* msconfig *Options

Option	Description
store.quotaenforcement	Enable quota enforcement When off, the quota database is still updated, but messages are always delivered. Default: 1 (bool).
store.quotanotification	Enable quota notification. Default: 0 (bool).
store.defaultmailboxquota	Store default quota by number of bytes. Default: "-1" (string) (unlimited).
store.defaultmessagequota	Store default quota by number of messages. Numeric. Default: "-1" (string) (unlimited).
store.quotaexceededmsg	Message to be sent to user when quota exceeds **store.quotawarn**. If none, notification is not sent. Default: No default (non-empty string). The message must contain a header (with at least a subject line), followed by $$, then the message body. The $ represents a new line. There is support for the following variables: [ID] - userid, [DISKUSAGE] - disk usage, [NUMMSG] - number of messages, [PERCENT] - **store.quotawarn** percentage, [QUOTA] - mailquota attribute, [MSGQUOTA] - mailmsgquota attribute.
store.quotaexceededmsginterval	Interval, in days, for sending overquota notification. Default: 7 (int32).
store.quotagraceperiod	Time, in hours, a mailbox has been overquota before messages to the mailbox will bounce back to the sender. Number of hours. Default: 120 (uint32).
store.quotawarn	Quota warning threshold. Percentage of quota exceeded before clients are sent an over quota warning. Default: 90 (int32).
store.quotaoverdraft	Used to provide compatibility with systems that migrated from the Netscape Messaging Server. When ON, allow delivery of one message that puts disk usage over quota. After the user is over quota, messages are deferred or bounced, the quota warning message is sent, and the quota grace period timer starts. (The default is that the quota warning messages are sent when the message store reaches the threshold.) Default: 0 (bool), but is treated as on if **store.overquotastatus** is set, otherwise the user can never go over quota and the **overquotastatus** is never used.
store.overquotastatus	Enable quota enforcement before messages are enqueued in the MTA. This prevents the MTA queues from filling up. When set, and a user is not yet over quota, but an incoming message pushes the user over quota, then the message is delivered, but the **mailuserstatus** LDAP attribute is set to overquota so no more messages are accepted by the MTA. Default: 0 (bool).

To have the preceding **msconfig** options take effect, restart Messaging Server.

The "imquotacheck" utility enables you to check message store usage against assigned quotas.

Also see "iminitquota" to update the information in the quota database if a user's quota-related LDAP attributes (or the system defaults) have been changed recently and the changes have not yet been propagated automatically to the quota database.

To Specify a Default User Quota

A default quota applies to users who do not have individual quotas set in their LDAP entries. The process consists of the following steps:

1. Specifying a user default quota
2. Specifying which users are bound to the default quota

The following examples show how to set default user quotas. Refer to *Messaging Server Reference* for detailed option information.

- To specify a default user disk quota for message size in bytes:

  ```
  msconfig set store.defaultmailboxquota [ -1 | number ]
  ```

  where **-1** indicates no quota (unlimited message usage) and **number** indicates a number of bytes.

- To specify a default user quota for total number of messages:

  ```
  msconfig set store.defaultmessagequota [ -1 | number ]
  ```

  where **-1** indicates no quota (unlimited messages) and **number** indicates the number of messages.

- To specify the default quota for specific users: Set the **mailQuota** attribute to **-2** in the user entries that use the default message store quota. Note that if **mailQuota** is not specified, the system default quota is used.

To Specify Individual User Quotas

Each user can have individualized quotas. To set user-specific quotas, set the **mailQuota** or **mailMsgQuota** attributes in the user's LDAP entry. For more information on the previous attributes, see *Schema Reference*.

In addition, configuration options of the form **store.quota*** can be used to implement more finely grained quota policies. See *Messaging Server Reference*. The following examples show how to set user quotas.

- To specify the system default quota, do not add **mailQuota** to the LDAP entry, or set it to **-2**.
- To set the quota to 1,000 messages, set **mailMsgQuota** to **1000**.
- To set the quota to two megabytes, set **mailQuota** to **2M**.
- To set the quota to two gigabytes, set **mailQuota** to **2G** or **2000M**.

The following LDAP entry specifies a 2 Gigabyte quota; a 20 Megabyte voice mail quota; and a 100 Megabyte quota for the Archive folder:

```
mailQuota: 2G;#voice%20M;Archive%100M
```

The two gigabyte quota represents all folders in the user's mailbox that are not explicitly assigned quotas. In this example, that excludes messages in the **Archive** folder, and messages of type **voice**. The 100 megabyte quota includes messages in any folders within the **Archive** folder.

Refer to the *Schema Reference* for more information about **mailQuota** values. Also see "Managing Message Types in the Message Store".

To Specify Domain Quotas

You can set disk space or message quotas for domains. These quotas are for the cumulative bytes or messages of all users in a particular domain. To specify domain quotas, set the **mailDomainDiskQuota** or **mailDomainMsgQuota** attributes in the desired LDAP domain entry. For more information about the previous attributes see *Schema Reference*.

- To set the quota to 1,000 messages, set **mailDomainMsgQuota** to **1000**.
- To set the quota to two megabytes, set **mailDomainDiskQuota** to **2M** or **2000000**.
- To set the quota to two gigabytes, set **mailDomainDiskQuota** to **2G** or **2000000000** or **2000M**.

To Set Up Quota Notification

Quota notification is the process of sending users a warning message when they are getting close to their quota.

1. To enable quota notification, run the following command:

   ```
   msconfig set store.quotanotification 1
   ```

   If the message is not set, no quota warning message is sent to the user.

2. To define a quota warning message, run the following command:

   ```
   msconfig set store.quotaexceededmsg 'message'
   ```

   For example:

   ```
   msconfig set store.quotaexceededmsg 'Subject: WARNING: User quota
   exceeded$$User quota threshold exceeded - reduce space used.'
   ```

   The Warning Message is the message that is sent to users who are close to exceeding their disk quota. The warning message must:

   - Be in RFC 822 format
   - Contain a header with at least a subject line, followed by $$, then the message body
   - Use "$" to represent a new line Depending on the shell that you are using, it might be necessary to append a \ before $ to escape the special meaning of $. ($ is often the escape character for the shell.) You can also use the following variables in the message:
   - **[ID]** - userid
   - **[DISKUSAGE]** - disk usage
   - **[NUMMSG]** - number of messages
   - **[PERCENT]** - **store.quotawarn** percentage
   - **[QUOTA]** - **mailquota** attribute
   - **[MSGQUOTA]** - **mailmsgquota** attribute The following example shows a warning message that uses these variables:

```
msconfig set store.quotaexceededmsg 'Subject: Overquota
Warning$$[ID],$$Your mailbox size has exceeded [PERCENT] of its allotted
quota.$Disk Usage: [DISKUSAGE]$Number of Messages: [NUMMSG]$Mailquota:
[QUOTA]$Message Quota: [MSGQUOTA]$$-Postmaster'
```

3. To specify how often the warning message is sent, run the following command:

   ```
   msconfig set store.quotaexceededmsginterval number
   ```

   where **number** indicates a number of days. For example, 3 would mean the message is sent every 3 days.

4. To specify a quota threshold, run the following command:

   ```
   msconfig set store.quotawarn number
   ```

   where **number** indicates a percentage of the allowed quota. A quota threshold is a percentage of a quota that is exceeded before clients are sent a warning. When a user's disk usage exceeds the specified threshold, the server sends a warning message to the user.

   Note: When the **store.quotaoverdraft** is enabled, email notifications are not triggered until the user's disk usage exceeds 100 percent of the quota, regardless of the threshold set with **store.quotawarn**.

   For IMAP users whose clients support the IMAP ALERT mechanism, the message is displayed on the user's screen each time the user selects a mailbox and a message is also written to the IMAP log.

To Disable Quota Notification

- To disable quota notification, run the following command:

  ```
  msconfig set store.quotanotification 0
  ```

To Enable or Disable Quota Enforcement

By default, users or domains can exceed their quotas with no effect except for receiving an over quota notification (if set). Quota enforcement locks the mailboxes from receiving further messages until the disk usage is reduced below the quota level.

To Enable Quota Enforcement at the User level

- To enable quota enforcement at the user level, run the following command:

  ```
  msconfig set store.quotaenforcement 1
  ```

  The MTA saves over-quota messages in its queues and notifies users that their messages were not delivered but that a redelivery attempt is to be made later. Delivery retries continue until either the grace period expires and all messages are sent back to the senders, or the disk usage falls below the quota and messages can be dequeued from the MTA and delivered to the message store. If you want to return messages that are over quota before they get to the message queues, use the following command:

  ```
  msconfig set store.overquotastatus 1
  ```

To Perform Quota Enforcement at the Domain Level

Unlike user-level quotas, domain-level quotas are not maintained dynamically.

- To enforce quotas for a particular domain, use the following command:

  ```
  imquotacheck -f -d domain
  ```

  To enforce quotas for all domains, exclude the **-d** option.

When **imquotacheck -f** finds a domain with **maildomainstatus=active** that has exceeded its quota, the **maildomainstatus** attribute is set to **overquota**, which halts all delivery to this domain. When **imquotacheck -f** is run again and the domain is back under quota, the value is set to **active**.

Disabling Quota Enforcement

If it appears that user quotas are being enforced, even when you have disabled them, check that the following options are disabled or not set:

- **store.quotaenforcement**
- **store.overquotastatus**
- **store.quotaoverdraft**

When **store.overquotastatus** is enabled (set to 1), it always treats **store.quotaoverdraft** as enabled, otherwise users never go over quota to trigger the rejection. Also, when **store.quotaoverdraft** is enabled, users are allowed one message that is smaller than the quota only. That is, it never accepts a message that is greater than the user's quota.

After changing these options, restart Messaging Server.

These Message Store attributes should be active:

- **maildomainstatus**
- **mailuserstatus**

Messages bounce if they are larger than the mailbox quota, regardless of quota enforcement configuration.

To Set a Grace Period

The grace period specifies how long the mailbox can be over the quota (disk space or number of messages) before messages are bounced back to sender. The grace period is not how long the message is held in the message queue, it's how long the mailbox is over quota before all incoming messages, including those in the message queue, are bounced. (see "Message Store Quota Overview" for more details.) The grace period starts when the user has reached the quota threshold and been warned. See "To Set Up Quota Notification" for more information.

- To specify a quota grace period at the command line:

  ```
  msconfig set store.quotagraceperiod number
  ```

where **number** indicates number of hours.

Netscape Messaging Server Quota Compatibility Mode

After disk usage exceeded the quota in the Netscape Messaging Server, the server deferred or bounced message delivery, sent an over quota notification, and started the

grace period. Messaging Server provides an option, **store.quotaoverdraft**, which retains this behavior.

When **store.quotaoverdraft** is enabled (set to 1), messages are delivered until disk usage is over quota. At that time, messages are deferred (messages stay in the MTA message queue but are not delivered to the message store), an over quota warning message is sent to the user, and a grace period starts. The grace period determines how long a mailbox is overquota before the overquota messages bounce. (The default is that the quota warning messages are sent when the message store reaches the threshold.) The default for this option is disabled (set to 0).

49

Managing Message Types in the Message Store

This chapter describes how to work with Oracle Communications Messaging Server message types. See "Message Store Message Type Overview" for conceptual information.

To Configure Message Types

To configure a message type, use the **msconfig** utility to set the **store.messagetype** option values that define and identify the message type.

1. Enable message types by setting the **store.messagetype.enable** option to 1. This option enables the message store to identify and manipulate message types. You must set this option before you can configure an individual message type. For example, type the following command:

   ```
   msconfig set store.messagetype.enable 1
   ```

2. Define and identify the message type by setting **store.messagetype.mtindex***x* options. The variable *x* identifies this particular message type in the message store. The variable *x* must be an integer greater than zero and less than 64. You can define up to 63 message types by iteratively configuring this option with unique integers. You define the value of the message type with a text string that describes the type.

   - For example, to define a text message type, type the following command:

     ```
     msconfig set store.messagetype.mtindex:1.contenttype text/plain
     ```

   - To define a voice message type, type the following command:

     ```
     msconfig set store.messagetype.mtindex:2.contenttype
     multipart/voice-message
     ```

3. Provide a flag name for the message type by setting the **store.messagetype.mtindex:***x***.flagname** option. This option creates a unique flag that identifies the message type. The flag is automatically set whenever a message of this type first arrives in the message store and remains associated with the message until it is purged. The flag name value is a text string that describes the message type. It does not have to be the same as the value set with the **store.messagetype.mtindex:***x* option. The variable *x* is the integer ID of the message type defined with the **store.messagetype.mtindex:***x* option. For example, to define flag names for the message types configured in the preceding step, type the following commands:

   ```
   msconfig set store.messagetype.mtindex:1.flagname text
   msconfig set store.messagetype.mtindex:2.flagname voice_message
   ```

4. Configure a quota root name for the message type by setting the **store.messagetype.mtindex:*x*.quotaroot** option. This option enables the quota function to identify and manage a quota root for this message type. The option value is a name-a text string that describes the message type. It does not have to be the same as the value set with the **store.messagetype.***x* option. The variable *x* is the integer ID of the message type defined with the **store.messagetype.***x* option. When this option is configured, you can set a quota that applies to the specified message type. See "Administering Quotas by Message Type" for more information. For example, to enable the use of quota roots for the message types configured in the preceding steps, type the following commands:

```
msconfig set store.messagetype.mtindex:1.quotaroot text
msconfig set store.messagetype.mtindex:2.quotaroot voice
```

5. To configure an alternate header field for identifying the message type, set the **store.messagetype.header** option. By default, the message store reads the Content-Type header field to determine the message type. Configure the **store.messagetype.header** option only if you want to use a different header field for identifying the message type. The value of this option is a text string. For example, to use a field called **X-Message-Type**, enter the following command:

```
msconfig set store.messagetype.header X-Message-Type
```

Sending Notification Messages for Message Types

Notifications can deliver status information about messages of different types, such as text messages, voice mail, and image data. Messaging Server uses Message Queue to send notification information for message types.

To enable the JMQ notification plug-in to recognize a particular message type, you must configure the **store.messagetype** options, including the **store.messagetype.mtindex:*x*.flagname** option.

Once the message types have been configured, JMQ notification messages can identify the particular message types. You can write a Message Queue client to interpret notification messages by message type and deliver status information about each type to the mail client.

The JMQ notification function counts the number of messages currently in the mailbox, by message type. Instead of sending one count, an array specifying the count for each message type is sent with the notification message. For example, a **NewMsg** notification message can carry data to users informing them that their inbox has new voice mail and text messages.

Administering Quotas by Message Type

When you set a quota for a message type, you include that value in a *quota root*. A quota root specifies quotas for a user. It can specify different quotas for particular message types and mailbox folders, and it can specify a default quota that applies to all remaining message types, folders, and messages not defined by type.

See "Quota Theory of Operations" for complete information about setting and managing quotas.

Before You Set Message-Type Quotas

Before you can set quotas for message types, you must configure the following options:

- Set the **store.messagetype.mtindex:*x*.quotaroot** option for each message type. See "To Configure Message Types" for details.
- Set the **store.typequota.enable** option to **1**. For example, type the following command:

```
msconfig set store.typequota.enable 1
```

Methods of Setting Message-Type Quotas

Use one of the following methods to set quotas for message types:

- Set message-type quotas for a user with the LDAP attributes **mailQuota** or **mailMsgQuota** (or both).

  For information about how to set quota roots with these attributes, see the **mailQuota** and **mailMsgQuota** entries in *Schema Reference*.

- Set default message-type quotas that apply to all individual users when the **mailQuota** and **mailMsgQuota** attributes are not set.

  To set default quotas, use the **store.defaultmessagequota** or **store.defaultmailboxquota** option (or both).

  See "Managing Message Store Quotas" for information about how to set quota roots with these options.

When you set a quota for the message type with a **msconfig** option or LDAP attribute, you must use the quota root specified with the *store.messagetype.mtindex:x.quotaroot* option.

Example of a Message-Type Quota Root

The example described in this section sets the following quotas for the user **joe**:

- The default mailbox storage quota is 40 MBytes.
- The default mailbox message quota is 5000.
- The storage quota for the Archive folder is 100 MBytes.
- The storage quota for text message types is 10 MBytes.
- The message quota for text message types is 2000.
- The storage quota for voice message types is 10 MBytes.
- The message quota for voice message types is 200.

This quota root permits greater storage in the Archive folder (100 MBytes) than in all the other folders and message types combined (60 MBytes). Also, no message limit is set for the Archive folder. In this example, only storage limits matter for archiving.

The message types have both storage and number-of-message quotas. The message-type quotas apply to the sum of all messages of those types, whether they are stored in the Archive folder or in any other folder.

The default mailbox quotas apply to all messages that are not text or voice message types and are not stored in the Archive folder. That is, the message-type quotas and Archive quota are not counted as part of the default mailbox quotas.

To set the quota root in this example:

1. Configure the **store.messagetype.mtindex*x*.quotaroot** option as follows:

```
msconfig set store.messagetype.mtindex:1.quotaroot text
msconfig set store.messagetype.mtindex:2.quotaroot voice
```

2. Configure the **mailQuota** attribute for the user **joe** as follows:

```
mailQuota: 20M;#text%10M;#voice%10M;Archive%100M
```

3. Configure the **mailMsgQuota** attribute for the user **joe** as follows:

```
mailMsgQuota: 5000;#text%2000;#voice%200
```

When you run the **getquotaroot** IMAP command, the resulting IMAP session displays all quota roots for the user **joe**'s mailbox, as shown here:

```
1 getquotaroot INBOX
* QUOTAROOT INBOX user/joe user/joe/#text user/joe/#voice
* QUOTA user/joe (STORAGE 12340 20480 MESSAGE 148 5000)
* QUOTA user/joe/#text (STORAGE 1966 10240 MESSAGE 92 2000)
* QUOTA user/joe/#voice (STORAGE 7050 10240 MESSAGE 24 200)
2 getquotaroot Archive
* QUOTAROOT user/joe/Archive user/joe/#text user/joe/#voice
* QUOTA user/joe/Archive (STORAGE 35424 102400)
* QUOTA user/joe/#text (STORAGE 1966 10240 MESSAGE 92 2000)
* QUOTA user/joe/#voice (STORAGE 7050 10240 MESSAGE 24 200)
```

Expiring Messages by Message Type

The expire and purge feature enables you to move messages from one folder to another, archive messages, and remove messages from the message store, according to criteria you define in expire rules. You perform these tasks with the **imexpire** utility. Because the **imexpire** utility is run by the administrator, it bypasses quota enforcement. See "Configuring Message Expiration (Tasks)" for information about how to write expire rules and use the **imexpire** utility.

You can write expire rules so that messages of different types are expired according to different criteria. The expire feature is extremely flexible, offering many choices for setting expire criteria. This section describes one example in which text and voice messages are expired according to different criteria.

The following example assumes you have configured text and voice message types as follows:

```
store.messagetype.mtindex:1 text/plain
store.messagetype.mtindex:2 multipart/voice-message
```

Assume also that the message store is configured to read the Content-Type header field to determine the message type.

Example: Sample Rules for Expiring Different Message Types

```
TextInbox.folderpattern: user/%/INBOX
TextInbox.messageheader.Content-Type: text/plain
TextInbox.messagedays: 365
TextInbox.action: fileinto:Archive
VoiceInbox.folderpattern: user/%/INBOX
VoiceInbox.messageheader.Content-Type: multipart/voice-message
VoiceInbox.savedays: 14
```

```
VoiceInbox.action: fileinto:OldMail
VoiceOldMail.folderpattern: user/%/OldMail
VoiceOldMail.messageheader.Content-Type: multipart/voice-message
VoiceOldMail.savedays: 30
VoiceOldMail.action: fileinto:Trash
Trash.folderpattern: user/%/Trash
Trash.savedays: 7
Trash.action: discard
```

In this example, text messages and voice mail are expired in different ways, and they follow different schedules, as follows:

- Text messages are moved from a user's inbox to the user's Archive folder one year after they arrive in the message store.
- Voice mail is moved from the inbox to the OldMail folder after two weeks. If the user saves a voice message, the saved date is reset, and the message is moved two weeks after the new date.
- Voice mail is moved from the OldMail folder to the Trash folder after 30 days. The user also can save a voice message in the OldMail folder, which postpones the removal of the message for another 30 days after the new saved date.
- Messages of all types are discarded seven days after they are moved to the Trash folder. The expire rules move voice mail to Trash automatically. Text messages are moved to Trash when a user deletes them.

Note: The **savedays** rule causes a message to be expired the specified number of days after the message is saved. In a typical voice mail system, a user can save voice mail on the voice mail menu. For text messages, a message is saved when it is moved to a folder. The **messagedays** rule causes a message to be expired the specified number of days after it first arrives in the message store, no matter which folder it is stored in or how often it is moved.

50

Managing Shared Folders

This chapter describes the tasks that you use to administer Oracle Communications Messaging Server shared folders.

Shared Folders Overview

A *shared folder* is like any other mail folder except that users other than its owner can read, delete, or add messages to it, depending on the access rights they are granted. Messages can be added to shared folders by normal drag and drop, by Sieve filters, or by sending messages directly using the form: *uid+folder@domain*.

The following example shows the address for sending email to a *private shared folder* owned by **carol.fanning@example.com** called **crafts_club**:

```
carol.fanning+crafts_club@example.com
```

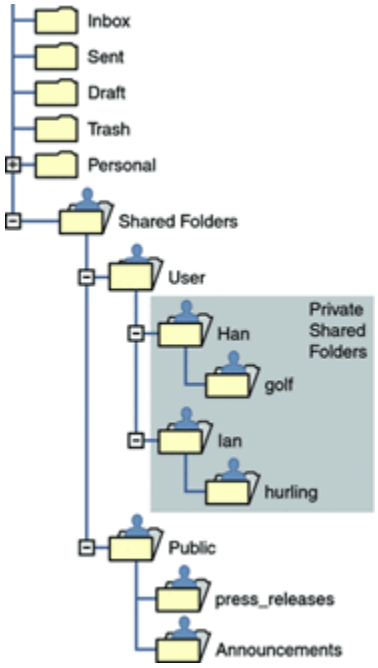
This example shows the address for sending email to a *public shared folder* called **tennis**:

```
public+tennis@example.com
```

Shared folders are useful for starting, sharing, and archiving an ongoing email conversation on a particular topic. For example, a group of software developers can create a shared folder for discussing development of a particular project called **mosaic_voices**. When a message is sent or dropped into the folder **mosaic_voices**, anyone who has permissions to access the shared folder (permissions can granted to individuals or groups) can open this mailbox and read the message.

Shared folders are displayed in user's mailbox tree under a folder called **Shared Folders**, as shown in Figure 50–1.

Figure 50–1 Example of Shared Mail Folder List as Seen from a Mail Client



There are two kinds of shared folders:

- **Private Shared Folder** - A shared folder created and owned by a specific user with access rights granted to other users or groups. The owner can grant access rights using Convergence or other mail clients that support shared folder creation. The mail administrator can also grant access rights using the **readership** command. Private shared folders appear in the **Shared Folders/User** mail folder directory.
- **Public Shared Folder** - A shared folder created by the mail administrator and not owned by a specific user. The mail administrator can grant access rights using the **readership** command. Public shared folders appear in the **Shared Folders/Public** mail folder directory.

For example, you might want a folder, such as **public+software_dev@example.com** for posting information about a special interest group inside the company. Interested employees would be granted access to this public folder.

Messaging Server allows folders to be shared among users of different backend message stores. See "Setting Up Distributed Shared Folders" for details.

See "Managing Shared Folders" for examples of creating shared folders and granting access rights.

Specifying Sharing Attributes for Private Shared Folders

A private shared folder is a normal folder, created by users in the same way that they create other folders. A folder becomes "shared" when its owner grants access rights to other users or groups. Methods to manage folder access include:

- Many IMAP clients
- Convergence web client
- Oracle Communications Messaging Server "readership" command, for mail administrators

Table 50–1 explains the **msconfig** options that pertain to private shared folders.

Table 50–1 Disabling Quota Enforcement

msconfig Option	Description	Default
store.privatesharedfolders.restrictanyone	If enabled (1), disallows regular users from setting the permission on private shared folders to **anyone**.	0
store.privatesharedfolders.restrictdomain	If enabled (1), disallows regular users sharing private folders to users outside of their domain.	0
store.privatesharedfolders.shareflags	If disabled (0), users of a shared folder have their own set of flags (for example, **seen**, **deleted**, and so on) for messages in that folder. If enabled (1), a single set of flags is shared between all users of each shared folder.	0

To Create a Public Shared Folder

Public shared folders must be created by the mail administrator because they require access to the LDAP database as well as the "readership" and "mboxutil" commands.

To create a publich shared folder:

1. Set the userid for Public shared folders. The **store.publicsharedfolders.user** option specifies the userid to act as a container for all public shared folders (see "Shared Folders Overview"). Typically, this is simply *public*. The default is NULL (unset).

   ```
   msconfig set store.publicsharedfolders.user public
   ```

2. Create an LDAP entry for that user. The **uid** must match that specified by **store.publicsharedfolders.user**, for example:

   ```
   dn: cn=public,ou=people,o=example.com,o=ISP
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   objectClass: inetUser
   objectClass: ipUser
   objectClass: inetMailUser
   objectClass: inetLocalMailRecipient
   objectClass: nsManagedPerson
   objectClass: userPresenceProfile
   cn: public
   mail: public@example.com
   mailDeliveryOption: mailbox
   mailHost: manatee.example.org
   uid: public
   inetUserStatus: active
   mailUserStatus: active
   mailQuota: -1
   mailMsgQuota: 100
   ```

3. Create folders within the **public** account by using the "mboxutil" command, for example:

   ```
   mboxutil -c user/public/gardening
   ```

4. Use the "readership" command to grant rights to allow users to access the folder. For example, the following command gives everyone in the **example.com** domain lookup, read, and posting access to the public folder **gardening**:

   ```
   readership -s user/public/gardening anyone@example.com lrp
   ```

The name *anyone@domain* is a special case to designate all users in the specified domain. It does not correspond to any user or group definition in LDAP. The name **anyone** without specifying a domain indicates anyone in any domain. The following command grants the user whose **uid** is **kelly** the same access rights as the owner of the folder:

```
readership -s user/public/gardening kelly@example.com lrswipcdan
```

For individual users, you only need to supply a domain name with hosted domains. Do not use a domain name if the user to whom access is being granted is in the default domain.

See the "readership" command for a list of the " ACL Rights Characters" and their meanings.

To Grant Folder Access Rights Based on Group Membership

In the previous examples, " ACL Rights Characters" have been granted to individual users or to the special case names **anyone** or **anyone@<domain>**. You can also grant rights based on group membership. Members of such a group are identified by having the **aclGroupAddr** attribute. For more information on the **aclGroupAddr**, see *Schema Reference*.

For example, a group called **tennis@example.com** has 25 members and the members have decided that they would like to create a shared folder to store all email going to this group address and to allow members of the group to access that shared folder.

The mail administrator uses the "readership" command to grant group access rights. A group name is distinguished from individual user names by the prefix "**group=**".

To grant folder access rights based on group membership:

1. Create the folder. In this example, the team decided to use a private shared folder. The user **gregk** could have created the folder by using a mail client, or the mail administrator could have created it by using the "mboxutil" command, for example:

   ```
   mboxutil -c user/gregk/gardening
   ```

   If the team were using a public shared folder, the mail administrator would have had to create it:

   ```
   mboxutil -c user/public/gardening
   ```

2. Use the "readership" command to grant **lookup**, **read**, and **posting** access privileges to the group:

   ```
   readership -s user/gregk/gardening group=tennis@example.com lrp
   ```

3. Assign group membership to the individual users. For the purpose of folder access control, group membership is determined by the **aclGroupAddr** attribute on the LDAP entry of the individual users. Add the attribute-value pair **aclGroupAddr=<group-name>** to the user entry of every member of the group, for example:

   ```
   aclGroupAddr: tennis@example.com
   ```

To create group objects in LDAP, you could use the **aclGroupAddr** attribute as the basis for a dynamic group, for example:

```
memberURL:
ldap:///o=example.com??sub?(&(aclGroupAddr=tennis@example.com)(objectclass=inetmai
luser))
```

However, note that the LDAP group object with mail address **tennis@example.com** is not used for determining group membership for the purpose of shared folder access. What matters is that the "**xxx**" value in **group=xxx** on the **readership** command matches the value of the **aclGroupAddr** attribute on the user's LDAP object.

Also note that if you use the **aclGroupAddr** attribute as the criteria for a dynamic group, you should check to make sure that attribute is indexed properly for such lookups.

To Set or Change a Shared Folder's Access Control Rights

Users can set or change the access control for a shared folder by using Convergence. Administrators can set or change the access control for a shared folder using the "readership" command line utility. The command has the following form:

```
readership -s foldername identifier rights_chars
```

where **foldername** is the name of the folder for which you are setting rights, **identifier** is the person or group to whom you are assigning the rights, and **rights_chars** are the rights you are assigning. For the meaning of each character, see Table 50–3, " readership Options" for more information.

Note: **anyone** is a special identifier. The access rights for **anyone** apply to all users. Similarly, the access rights for **anyone@domain** apply to all users in the same domain. For the identifier, only supply a domain name with hosted domains. Do not use a domain name if the folder is in the default domain.

Shared Folder Examples

- To assign everyone in the **example** domain to have lookup, read, and email marking (but not posting) access to the public folder called **golftournament**, type the following command:

  ```
  readership -s user/public/golftournament anyone@example lwr
  ```

- To assign the same access to everyone on the message store type the following command:

  ```
  readership -s user/public/golftournament anyone lwr
  ```

- To assign lookup, read, email marking, and posting rights to a group, type the following command:

  ```
  readership -s user/public/golftournament group=golf@example.com lwrp
  ```

- If you want to assign administrator and posting rights for this folder to an individual, **jdoe**, type the following command:

  ```
  readership -s user/public/golftournament jdoe@example.com lwrpa
  ```

- To deny an individual or group access to a public folder, prefix the **userid** with a dash. For example, to deny lookup, read, and write rights to **jsmith**, type the following command:

```
readership -s user/public/golftournament -jsmith@example.com lwr
```

- To deny an individual or group an access right, prefix the ACL rights character with a dash. For example, to deny posting rights to **jsmith**, type the following command:

```
readership -s user/public/golftournament jsmith@example.com -p
```

- To remove an individual or group access right setting from a folder, set it to an empty set. This is different from an ACL to deny access:

```
readership -s user/public/golftournament jsmith@example.com ""
```

Note: Posting messages to a shared folder by using the *uid+folder@domain* address requires that the **p** (post) access right be used with the readership command. See "To Set or Change a Shared Folder's Access Control Rights" for more information.

Enabling or Disabling Listing of Shared Folders

Use the **store.sharedfolders** option to enable to disable listing of shared folders when responding to an IMAP **LIST** command. Setting the option to **0** disables it. The setting is enabled by default (set to **1**). **SELECT** and **LSUB** commands are not affected by this option. The **LSUB** command returns every subscribed folder, including shared folders. Users can **SELECT** the shared folders they own or are subscribed to.

Setting Up Distributed Shared Folders

Normally, shared folders are only available to users on a particular message store. Messaging Server, however, enables you to create **distributed shared folders** that can be accessed across multiple message stores. That is, access rights to distributed shared folders can be granted to any users within the group of message stores. However, web mail clients do not support remote shared folders access. Users can list and subscribe to the folders, but they cannot view or alter the contents.

Distributed shared folders require the following:

- Every message store **userid** must be unique across the group of message stores.
- The directory data across the deployment must be identical.

The remote message stores (that is the message stores that do not hold the shared folder) must be configured as proxy servers by setting the configuration variables listed in Table 50–2.

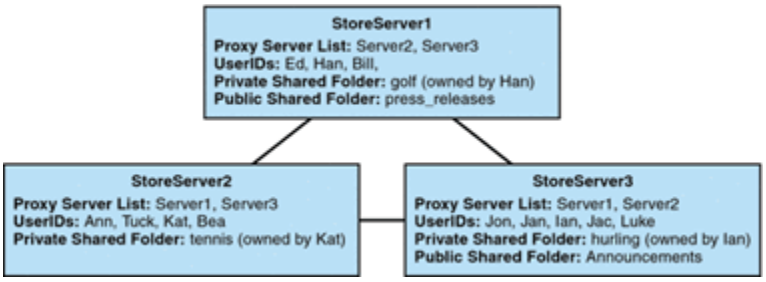
Table 50–2 Variables for Configuring Distributed Shared Folders

Name	Value	Data Format
base.proxyserverlist	Message store server list to list shared folders from	space-separated strings
base.proxyadmin	Default store admin login name	string
base.proxyadminpass	Default store admin password	string
proxy:*hostname***.imapadmin**	Store admin login name for a specific host if different from **base.proxyadmin**	string
proxy:*hostname***.imapadminpass**	Store admin password for a specific host if different from **base.proxyadminpass**	string

Setting Up Distributed Shared Folders-Example

Figure 50–2 shows a distributed folder example of three message store servers called **StoreServer1**, **StoreServer2**, and **StoreServer3**.

Figure 50–2 Distributed Shared Folders-Example



These servers are connected to each other as peer proxy message stores by setting the appropriate **msconfig** options. Each server has a private shared folder: *golf* (owned by Han), *tennis* (owned by Kat), and *hurling* (owned by Luke). In addition, there are two public shared folders called *press_releases* and *Announcements*. Users on any of the three servers can access any of these three shared folders.

The following example shows the ACLs for each server in this configuration.

> **Note:** The **imcheck -d** command is only valid for classic message store.

```
$ StoreServer1 :> imcheck -d lright.db
Ed: user/Han/golf
Ian: user/Han/golf
anyone: user/public/press_releases

$ StoreServer2 :> imcheck -d lright.db
Jan: user/Kat/tennis
Ann: user/Kat/tennis
anyone: user/public+Announcements user/public+press_releases

$ StoreServer3 :> imcheck -d lright.db
Tuck: user/Ian/hurling
Ed: user/Ian/hurling
Jac: user/Ian/hurling
anyone: user/public/Announcements
```

Monitoring and Maintaining Shared Folder Data

The **readership** command-line utility enables you to monitor and maintain shared folder data which is held in the **folder.db**, **peruser.db**, and **lright.db** files. **folder.db** has a record for each folder that holds a copy of the ACLs. The **peruser.db** has an entry per user and mailbox that lists the various flags settings and the last date the user accessed any folders. The **lright.db** has a list of all the users and the shared folders for which they have lookup rights.

Table 50–3 shows the options for the **readership** command-line utility.

Table 50–3 readership Options

Options	Description
-d days	Returns a report, per shared folder, of the number of users who have selected the folder within the specified days.
-p months	Removes data from the **peruser.db** for those users who have not selected their shared folders within the specified months.
-l	List the data in **lright.db.**
-s*folder_identifier_rights*	Sets access rights for the specified folder. This updates the **lright.db** as well as the **folder.db**.

Using the various options, you can perform the following functions:

- To Monitor Shared Folder Usage
- To List Users and Their Shared Folders
- To Remove Inactive Users
- To Set Access Rights

To Monitor Shared Folder Usage

To find out how many users are actively accessing shared folders, use the following command:

```
readership -d days
```

where **days** is the number of days to check. Note that this option returns the number of active users, not a list of the active users.

Example: To find out the number of users who have selected shared folders within the last 30 days:

```
readership -d 30
```

To List Users and Their Shared Folders

To list users and the shared folders to which they have access, use the following command.

> **Note:** The **imcheck -d** command is only valid for classic message store.

```
imcheck -d lright.db
```

Example output:

```
imcheck -d lright.db
group=lee-staff@example.org: user/user2/lee-staff
richb: user/golf user/user10/Drafts user/user2/lee-staff user/user10/Trash
han1: user/public+hurling@example.org user/golf
gregk: user/public+hurling@example.org user/heaving user/tennis
```

To Remove Inactive Users

If you want to remove inactive users (those who have not accessed shared and other folders in a specified time period), use the following commands:

1. This command writes the inactive mailboxes to a file:

   ```
   mboxutil -o [-w file] [-t number_of_days]
   ```

2. This command removes the mailboxes in a given file:

   ```
   mboxutil -d -f file
   ```

Example: Remove users who have not accessed folders for the past six months (180 days) using a file named inactive_users:

```
mboxutil -o -w inactive_users -t 180
mboxutil -d -f inactive_users
```

To Set Access Rights

You can assign access rights to a new public folder, or change access rights on a current public folder.

See "To Set or Change a Shared Folder's Access Control Rights" for an example of how to set access rights with this command

51

Upgrading the Classic Message Store

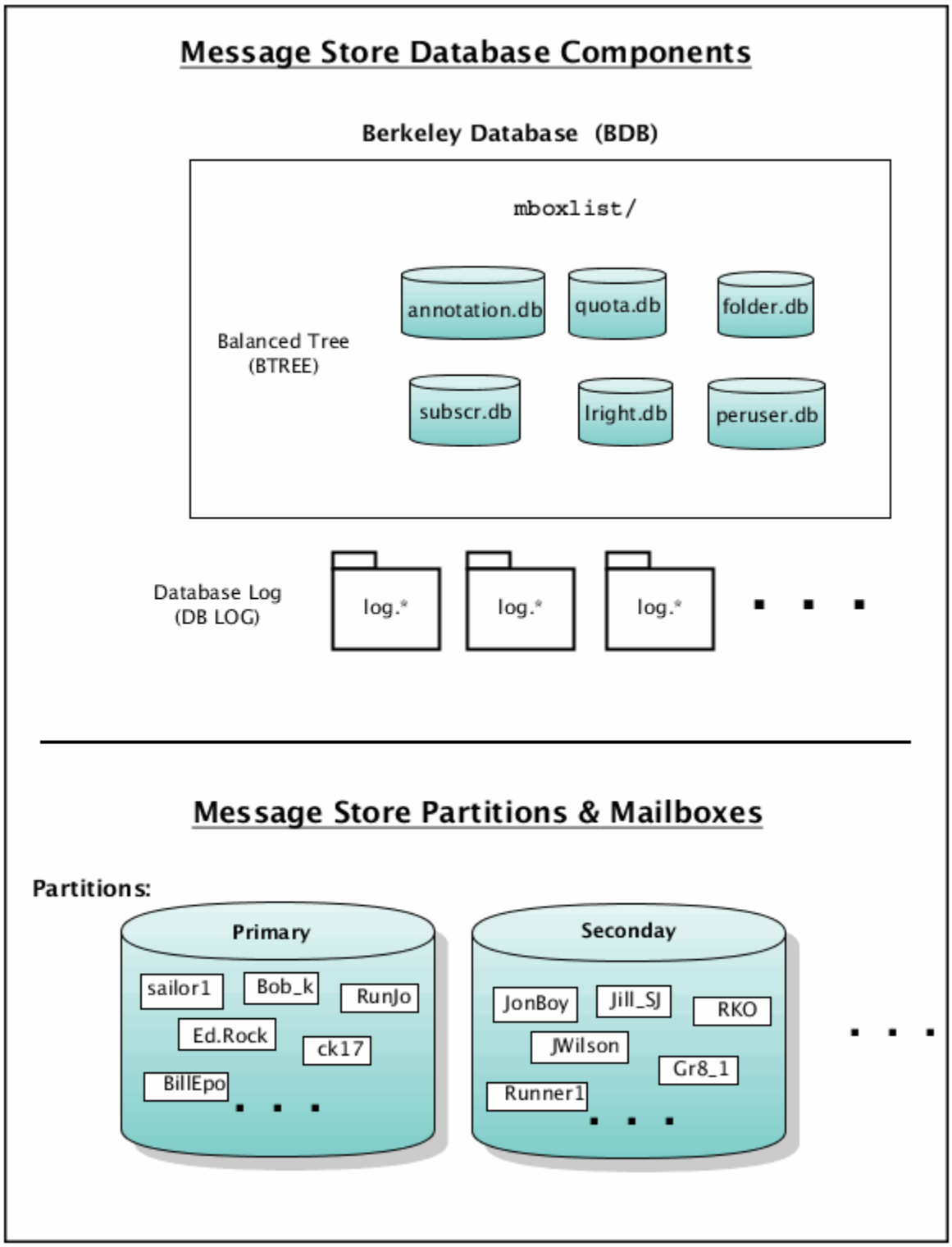
This chapter describes the data components and tools of the Oracle Communications Messaging Server classic message store, and how they affect data upgrades. It provides the technical background for upgrade planning. If you are using the upgrade procedure described in *Messaging Server Installation and Configuration Guide*, you do not need to concern yourself with this level of message store technical detail. However, if you need to customize your message store upgrade process, use this information as a guideline.

See "Significant Changes in the Classic Message Store Between Versions" for specific information about significant design changes in the message store between versions that can impact disk I/O and performance.

Architecture and Components

Figure 51–1 shows the classic message store architecture and its components. See "Message Store Directory Layout" for a view of the message store file structure.

Figure 51–1 Classic Message Store and Components



The classic message store uses the *Berkeley Database* provided by Sleepycat Software and, since 2006, Oracle Corporation. The message store database files are stored in a directory called **mboxlist** (see "Classic Message Store Directory Layout" for more information) and so it is often called the **mboxlist** database.

The classic message store has used several versions of the Berkeley Database over the course of its existence. Thus, when you upgrade your message store, the Berkeley Database may also be upgraded. The database engine upgrade has complexities and implications for the data upgrade. (These complexities and implications are handled in the Message Store upgrade tools and instructions described in *Messaging Server Installation and Configuration Guide*, but these details are described here for custom upgrade plans.)

The **mboxlist** database is stored in the *BTREE database files*. The BTREE database files store information about the message store users and mailboxes (see "Classic Message Store Directory Layout"). The number and types of files, as well as the structure of files themselves have varied over the life of the Berkeley Database.

The Berkeley Database is transactional, so each transaction is logged in a *database log file*. Database log files are used for recovery. All database changes are recorded in the database log file. When the server is crashed and restarted, it uses the log file to bring the database back to a consistent state. Before you do an upgrade, make sure you have a clean shutdown and recovery by running **stored -r**.

Classic Message Store Component Version Compatibilities

Table 51–1 contains the version number of various database components in the classic message store with respect to upgrade.

Table 51–1 Classic Message Store Database Components

Messaging Server	Mailbox	Berkeley Database	Database BTREE	Database Log Files
4.15	1_1	2.6	6	2
5.x	1_2	2.6	6	2
6.0	1_2	3.2.9	8	3
6.1	1_2	4.2	9	8
6.2	1_2	4.2	9	8
6.3	1_2	4.4	9	11
7.0	1_3	4.4	9	11
7 Update 1	1_3	4.7.25	9	14
7 Update 2	1_3	4.7.25	9	14
7 Update 3	1_3	4.7.25	9	14
7 Update 4	1_3	4.7.25	9	14
7 Update 5	1_4	5.3.21	9	19
8.0	1_5	6.1.19	10	22
8.0.1	1_5	6.1.26	10	22
8.0.2	1_5	6.2.32	10	23
8.1.0	1_5	6.2.32	10	23

In general, when you upgrade Messaging Server from one version to another, you also have to upgrade the components that have a different version. Certain message components are not compatible with other components. The following section describes the various compatibilities and incompatibilities.

Upgrading the Mailboxes

The classic message store upgrades mailboxes from version 1_1-1_2 and 1_2-1_4 to the current version automatically. Usually, a mailbox is upgraded when the end users select their mailboxes, or when messages are delivered to the mailboxes after the message store software is upgraded. The message store checks the mailbox version in the mailbox header and upgrades the mailbox if needed.

Mailbox upgrade increases the load on the server. Disable the non-essential tasks during the transition period, such as:.

- Disable peruser.db archive:

  msconfig -o store.seenckpinterval -v 0 (Unified Configuration)

  or

  configutil -o local.store.seenckpinterval -v 0 (legacy configuration)

- Disable last access time tracking:

  msconfig -o base.enablelastaccess -v 0 (Unified Configuration)

  or

  configutil -o local.enablelastaccess -v no (legacy configuration)

- Decrease DB snapshot frequency
- Decrease log level
- Temporarily disable **delivery**
- Temporarily disable **impurge**

You can upgrade the mailboxes manually with **imcheck -H** during off-peak hours. The **imcheck** command opens every mailbox, which triggers the upgrade. You can run this command to make sure all the mailboxes are upgraded.

To upgrade from mail version 1_1 to 1_4, you have to use a migration tool such as **imsbackup** and **imsrestore**.

Downgrading mailbox versions is not automatic. You have to use a migration tool such as **imsbackup** and **imsrestore** to downgrade a mailbox.

Upgrading and Downgrading the Berkeley Database (BDB)

The classic message store uses the Berkeley Database to store various data. The email messages themselves are not part of the data stored in the BDB. If the database is upgraded to a version that is incompatible with the previous version, the database files will become incompatible with older versions of the classic message store. Therefore, make a backup copy of the database before the upgrade in case you want to back out of the upgrade.

The primary database which needs to be backed up is located by default in the **mboxlist/** directory, and consists of **.db**files, and**log.** files. The __* files are temp files for the database and do not need to be copied. A correct copy of the database ensures data is in a consistent state. The message store and utilities must be shut down. Running **stored -r** will make sure the cache files are synced to the database files. Both database and log files are required.

If you need to back out of a patch or update which upgrades the Berkeley Database to a version that is incompatible with previous versions and you did not make a backup, you may be able to rely on a previous database snapshot. Database snapshots are located by default in the **dbdata/** directory. A valid database snapshot directory will

have a **.verified** file. The**.verified** file indicates that the snapshot has been recovered, verified, and is ready to be used.

Normally when the store is brought up, the **stored** process replaces the **mboxlist** directory with any snapshots needed. In the case where you have backed out of a database upgrade, **stored** may not take into account that a downgrade has occurred. It may therefore be necessary to replace the valid snapshot manually. To do this, move the current database files in **mboxlist/** out of the directory and move all the files from the chosen snapshot directory into the **mboxlist/** directory. Be sure to remove the __* tmp files as well. Note that if the store is configured with **store.dbtmpdir**, the tmp files will be in a different location.

If you have no database backup and no valid snapshots, it may be necessary to move the upgraded database out of the way, and rebuild it from scratch. Under normal circumstances, the classic message store rebuilds the database while allowing users to access their mail. Since doing this puts a heavier load on the system, you should create proper database backups instead.

Normal reparation of the database should be done after putting an older version in place by running the **reconstruct -m** and **reconstruct -r** commands.

Some of these manual requirements might be addressed in future releases.

Note that the Berkeley database consist of BTREE files, LOG files, and temporary files (tmp file location is configurable with store.dbtmpdir). To upgrade the database, run "stored -r" before replacing the new libraries and binaries. Note that "stored -r" runs automatically during the proper upgrade process.

In the unlikely event that **stored -r** fails, check the store log files to determine the cause, run **stored** alone to perform a database recovery, and run **stored -r** again. In normal circumstances, this should not be a problem unless there is some underlying system problem which you should resolve before upgrading.

Database BTREE File

BTREE version 8, 9, and 10 are compatible. Upgrade is not needed.

To upgrade the BTREE files from version 6 (BDB 2.6) to a higher version, copy the database files from the old location (example: **/usr/iplanet/server5/msg-store/store/mboxlist** for 5.2) to the new **mboxlist** location (example: **/var/opt/sun/comms/messaging64/store/mboxlist**), then run **ims_db_upgrade**.

Database Log Files

Database log files cannot be upgraded. When the log version changes, run the legacy version of "stored -r" to process the log files (recover the database) before upgrading. Do not remove the old log files.

The following message is logged when the server restarts.

'Skipping log file...historic log version'

The store daemon creates a new log file with the new version.

IMAPD, MSHTTPD and Convergence

The webmail server (**mshttpd**) uses **imapd** to access the message store. **imapd** and **mshttpd** in 6.3 and 7.*x* are fully compatible, so simultaneous upgrade is not required. To prevent memory problems due to redundant IMAP sessions from **mshttpd**, you should run with only one mshttpd process. If you serve a lot of concurrent webmail

users, this might require an upgrade to 64-bit so that you have enough virtual address space for the process.

Convergence requires webmail server. To use Convergence, you must upgrade **mshttpd** to the current version.

Upgrading from Messaging Server 32-bit to 64-bit

Run the legacy version of "stored -r" before upgrading the Messaging Server software from 32-bit to 64-bit.

If you run the 64-bit version, then it is more efficient to run with only one **mshttpd** process.

You might also want to reduce the number of **imapd** process.

Migrating from x86 to SPARC

The classic message store data formats are architecture dependent. You cannot transfer any data components in the matrix from one architecture to another directly. To migrate the message store from an x86 machine to a SPARC machine (or from SPARC to x86), use **imsbackup** and **imsrestore** or **rehostuser**.

stored -r

For cluster upgrade, the administrator need to run **stored -r** on the node running old software before upgrading the node with new software. To run **stored -r**, use the following procedure:

```
# stop-msg store
# stored -r
# stop-msg
```

stored -r performs a final recovery and cleanly removes the database temp files to prepare the database for an upgrade.

For example:

```
/opt/sun/comms/messaging64/lib/stored -r
removing mboxlist environment ... done
removing lock environment ... done
removing session db ... done
```

Make sure **stored -r** completes successfully.

The **stored -r** command requires the watcher process. Make sure that the watcher is running before you perform **stored -r**.

ims_db_upgrade

ims_db_upgrade copies the database files to a backup directory, upgrades the database files to the current version and validates the new database. If upgrade is not successful, the backup files are restored. If the database is validated, the backup files are removed.

Downgrading

Mailbox and BDB downgrade are not supported. To downgrade a classic message store to an older version with incompatible mailboxes or databases, you must restore the mailboxes and mboxlist database from backup.

Significant Changes in the Classic Message Store Between Versions

This section describes the significant design changes in the classic message store that can impact disk I/O and performance. Administrators and deployment designers may need to be aware of these changes for deployments where mboxlist, index, and message data are on different file systems or pools.

Changes from Messaging Server 6.3 to Messaging Server 7.0

The following changes were made between Messaging Server 6.3 and Messaging Server 7.0:

- Changes to store.idx
- Classic Message Store Maintenance Queue and impurge
- Mailbox Self-Healing (Auto-Repair)

Changes to store.idx

The **store.idx** file is separated into two or more files. The **store.idx** file contains the mailbox's meta data and a 128 bytes fixed length record for every message in the folder. The **store.cN** files contain the variable length cache records. The average cache record size is approximately 2 kilobytes.

Benefits

- Mailbox expunge is much faster
- Supports very large mailboxes (up to 33554431 messages)

I/O Impact

- More files (inodes) on the index partitions
- More I/O for very small mailboxes
- Initial mailbox access triggers mailbox upgrade automatically

Classic Message Store Maintenance Queue and impurge

A BDB queue is added to the message store to manage maintenance tasks. Cache file purge and message file cleanup tasks are queued and executed by **impurge** which runs continuously.

Benefits

- Less I/O overall (especially write)
- Event driven message store maintenance
- Higher performance
- Eliminates sudden increase in load during nightly cleanup

I/O Impact

- Less write access
- Higher disk space usage (approximately 10%)

Mailbox Self-Healing (Auto-Repair)

Mailbox corruption is detected and scheduled for repair automatically by **impurge**.

Benefits

- Increases availability
- Reduces mailbox administration cost

I/O Impact

- Minor I/O increase due to error detection and repair.

Changes from Messaging Server 7 to Messaging Server 7 Update 1

The following change was made between Messaging Server 7 and Messaging Server 7 Update 1:

- Berkeley Database Upgrade

Berkeley Database Upgrade

The Berkeley Database was upgrade from version 4.4 to version 4.7.25.

Benefits

- Higher throughput

I/O Impact

- None

Changes from Messaging Server 7 Update 1 to Messaging Server 7 Update 5

The following changes were made between Messaging Server 7 Update 1 and Messaging Server 7 Update 5:

- Changes to the Owner's Seen and Deleted Flags
- Immediate flag update and state sharing
- Change to the service.imap.capability.condstore option
- Changes to the Berkeley Database
- Changes to mboxlist and lockdir BDB environments

Changes to the Owner's Seen and Deleted Flags

The owner's seen and deleted flags have been moved from the Berkeley Database to the **store.idx** file.

Benefits

- Higher throughput by reducing Berkeley Database lock contentions
- Performance bottlenecks can be fixed by adding more spindles

I/O Impact

- Less I/O on the mboxlist file system
- More I/O on the index file system
- Initial mailbox access triggers mailbox upgrade automatically

Immediate flag update and state sharing

Flags and ACL changes are flushed to the disk and shared immediately across IMAP sessions. New message arrival are notified immediately.

Benefits

- IMAP sessions are more up-to-date

I/O Impact

- More I/O on the index file system

Change to the service.imap.capability.condstore option

The **service.imap.capability.condstore** option is enabled by default.

Benefits

- IMAP clients can utilize CONDSTORE
- Improve overall performance (when CONDSTORE is utilized)

I/O Impact

- Minimal (because flags and MODSEQ are updated together)

Changes to the Berkeley Database

The Berkeley Database was upgrade from version 4.7.25 to version 5.3.21. The default cache size increased from 16 MB to 64 MB. Moved the maintenance queue to the mboxlist environment.

Benefits

- Increase performance and scalability
- Simplify store maintenance tasks

I/O Impact

- A small increase in database environment (**tmp** file) size.

Changes to mboxlist and lockdir BDB environments

Mboxlist and lockdir DB environments (store.dbtmpdir and local.lockdir) default to **tmpfs**.

Benefits

- Better performance

I/O Impact

- Less I/O on the data root file system.

52

Message Store Automatic Recovery On Startup

This chapter provides a conceptual overview of the Oracle Communications Messaging Server startup and automatic recovery process of **stored**, and Message Store Database Snapshot. See "Administering Message Store Database Snapshots (Backups)" for task information.

Overview of Automatic Recovery on Startup

Message store data consists of the messages, index data, and the message store database. While this data is fairly robust, on rare occasions there may be message store data problems in the system. These problems are indicated in the default log file, and almost always are fixed transparently. In rare cases an error message in the log file may indicate that you need to run the **reconstruct** utility. In addition, as a last resort, messages are protected by the backup and restore processes. See "Backing Up and Restoring the Message Store" for more information.

The message store automates many recovery operations which were previously the responsibility of the administrator. These operations are performed by message store daemon **stored** during startup and include database snapshots and automatic fast recovery as necessary. **stored** thoroughly checks the message store's database and automatically initiates repairs if it detects a problem.

stored also provides a comprehensive analysis of the state of the database by writing status messages to the default log, reporting on repairs done to the message store and automatic attempts to bring it into operation.

Automatic Startup and Recovery Theory of Operations

The **stored** daemon starts before the other message store processes. It initializes and, if necessary, recovers the message store database. The message store database keeps folder, quota, subscription, and message flag information. The database is logging and transactional, so recovery is already built in. In addition, some database information is copied redundantly in the message index area for each folder.

Although the database is fairly robust, on the rare occasions that it breaks, in most cases **stored** recovers and repairs it transparently. However, whenever **stored** is restarted, you should check the default log files to make sure that additional administrative intervention is not required. Status messages in the log file indicate that **reconstruct** should be run if the database requires further rebuilding.

Before opening the message store database, **stored** analyzes its integrity and sends status messages to the default log under the category of **warning**. Some messages are useful to administrators and some messages consist of coded data to be used for

internal analysis. If **stored** detects any problems, it attempts to fix the database and try starting it again.

When the database is opened, **stored** signals that the rest of the services may start. If the automatic fixes failed, messages in the default log specify what actions to take. See "Error Messages Signifying reconstruct Is Needed" for more information.

After most recoveries, the database is usually be up-to-date and no further action is required. However, some recoveries require a **reconstruct -m** to synchronize redundant data in the message store. Again, this is stated in the default log, so it is important to monitor the default log after a startup. Even though the message store seems to be up and running normally, it is important to run any requested operations such as **reconstruct**.

Another reason for reading the log file is to determine what caused damage to the database in the first place. Although **stored** is designed to bring up the message store regardless of any problem on the system, you should ascertain cause of the database damage as this may be a sign of a larger hidden problem.

Error Messages Signifying reconstruct Is Needed

This section describes the type of error messages that require **reconstruct** to be run.

When the error message indicates mailbox error, run **reconstruct <mailbox>**. Example:

```
Invalid cache data for msg 102 in mailbox user/joe/INBOX. Needs reconstruct

Mailbox corrupted, missing fixed headers: user/joe/INBOX

Mailbox corrupted, start_offset beyond EOF: user/joe/INBOX
```

When the error message indicates a database error, run **reconstruct -m**. Example:

```
Removing extra database logs. Run reconstruct -m soon after startup to resync
redundant data

Recovering database from snapshot. Run reconstruct -m soon after startup to resync
redundant data
```

Message Store Database Snapshot Theory of Operations

This section describes concepts about the message store database snapshot. See "Administering Message Store Database Snapshots (Backups)" for a description of related tasks.

A snapshot is a hot backup of the database and is used by **stored** to restore a broken database transparently. This is much quicker than using **reconstruct** to rebuild the entire database from scratch with the information stored in the message and index partitions.

Snapshots of the database are taken automatically by the scheduler. The default snapshot schedule consists of a full snapshot every day and incremental snapshots every 10 minutes. (Note that older versions of Messaging Server have a more frequent default incremental snapshot schedule).

If the recovery process decides to remove the current database because it is determined to be bad, **stored** will move it into the **removed** directory if it can. This allows the database to be analyzed if desired.

Message Store Database Snapshot Interval and Location

There should be five times as much space for the database and snapshots combined. It is highly recommended that the administrator reconfigure snapshots to run on a separate disk, and that it is tuned to the system's needs.

If **stored** detects a problem with the **mboxlist** database at startup, the most recent verified snapshot is automatically restored. Two snapshot options can be configured: the location of the snapshot file and number of snapshots saved. See "Message Store Database Snapshot Options" for more information about these options.

Having a snapshot interval which is too small results in a frequent burden to the system and a greater chance that a problem in the database is copied as a snapshot. Having a snapshot interval too large can create a situation where the database holds the state it had back when the snapshot was taken.

A snapshot updates all the snapshots with the current data in the **mboxlist** database.

The ultimate role the snapshot plays is to get the system as close to up-to-date and ease the burden of the rest of the system trying to rebuild the data on the fly.

Message Store Database Snapshot Options

Table 52–1 shows the snapshot options that you set with the **msconfig** command.

Table 52–1 Message Store Database Snapshot Options

Option	Description
store.snapshotpath	Location of message store database snapshot files. Either existing absolute path or path relative to the **store** directory. Default: **dbdata/snapshots**
store.snapshotdirs	Number of snapshots to maintain. Do not set this to more than 3. Default: **3**

53

Message Store Maintenance Queue

This chapter describes the Oracle Communications Messaging Server message store maintenance queue.

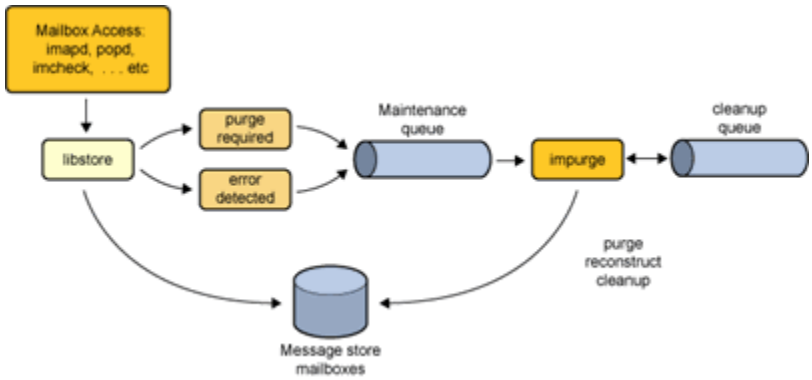
Message Store Maintenance Queue Overview

The message store maintenance queue purges mailboxes of unused cache records and message files. Only the index file is purged when a mailbox is expunged. The purging of cache records is deferred until the amount of expunged data has exceeded a configurable limit. In addition, the maintenance queue is used to schedule mailbox purge and repair tasks. Mailbox corruptions detected by the message store are also queued for repair automatically.

Figure 53–1 summarizes the major components and their interactions with the queue.

Note: Classic message store uses three maintenance queues, whereas Cassandra message store uses only a single maintenance queue.

Figure 53–1 Message Store Maintenance Queue



When a mailbox is expunged, the message store processes update the number of expunged messages and enqueue the mailbox name for purging when the number of expunged messages has exceeded the configured limit (set by the **store.cleanupsize** option). Similarly, when store corruptions are detected by the store processes, they enqueue the mailbox name for repair. A maintenance queue entry contains a mailbox name, a task ID, and a timestamp. The **impurge** process dequeues the entries, checks the timestamp and mailbox size or last repair time and performs the maintenance task if required. The mailbox size and last repair time are stored in the **mboxlist** database. Mailboxes that have already been repaired after the enqueue time are ignored.

Obsoleted cache and message files are enqueued for removal after the cleanup age has expired.

You can set the **impurge** process to execute as a daemon or through the scheduler. When you use the scheduler, you can configure **impurge** to exit when the queue is empty or when the end time expires. In this way, you can configure **impurge** to run during off peak hours.

impurge executes as a daemon by default.

Displaying the Maintenance Queue

Use the **imcheck -q** command to display the contents of the maintenance queue, for example:

```
imcheck -q
RecNo      Mailbox                            Timestamp      Action
---------- ---------------------------------- -------------- ------
2          user/username                      20080225095558 Clean
```

The maintenance queue database is stored under the *DataRoot***store/mboxlist** directory and persists across Messaging Server restarts.

Deleting, Expunging, Purging, and Cleaning Up Messages

Message removal is a four steps process:

1. Delete
   - A user deletes a message. This results in the per-user **\Deleted** flag being set on the message. If there is a second client, the deleted flag may not be recognized immediately by that second client. You can set the **imap.immediateflagupdate** option to enable immediate flag update.
2. Expunge and Expire
   - When the mailbox is expunged, messages with the **\Deleted** flag are removed from the **store.idx** file. The message itself is still on disk, but once messages are expunged, clients can no longer restore them. The number of the expunged messages is also recorded in the "Expunged" mailbox meta-data field. This field can be reviewed using the **./imcheck -m***mailbox* command.
   - When a message matches an expiration rule during an **imexpire** run, the **imexpire** command removes the message from the **store.idx** file and increments the "Expunged" mailbox meta-data field.
3. Cleanup

   Note: The cleanup step only occurs for classic message store. For Cassandra message store, once the value for the **store.cleanupsize** option is reached, the store is immediately in the purge state.

   - Cleanup of a folder is scheduled when the number of expunged in the folder messages exceeds **store.cleanupsize**, default is 100. The **store.exp** file is renamed to **store.exp.timestamp**.
   - Obsolete message files and cache files are removed from the store partitions when the **store.cleanupage** period has elapsed. **store.cleanupage** controls the cleanup grace period.

4. Purge
   - **impurge** purges the cache files when the number of expunged cache records exceeds **store.purge.count** (default is 500) and the percentage of expunged cache records exceeds **store.purge.percentage** (default is 5%). This is different from the **imsimta purge** command, which purges older MTA log files.

Mailbox Self Healing (Auto Repair)

The message store performs sanity checks when a mailbox is opened and when index and cache data are accessed. When the message store encounters a mailbox format error, and error is returned to the client and the mailbox is enqueued for repair. **impurge** dequeues the mailbox and repairs the mailbox automatically. The mailbox size and last repair time are stored in the mboxlist database. Mailboxes that have already been repaired after the enqueue time are ignored.

The following is an example of the auto repair process. In this example, the **store.idx** file for a user's INBOX has been removed. When the user accessed their INBOX the following error was reported in the **imap** log file:

```
[25/Feb/2012:09:55:57 +1100] hostname imapd[7264]: Store Critical: Unable to open
mailbox user/username: Mailbox does not exist
[25/Feb/2012:09:55:57 +1100] hostname imapd[7264]: Store Error: user/username:
Mailbox has an invalid format; repair requested
```

In the **default** log file an impurge record indicates that the requested mailbox repair as been processed and the store.idx restored:

```
[25/Feb/2012:09:55:58 +1100] hostname impurge[7263]: General Notice: repairing
user/username
```

Maintenance Queue Configuration Options

The **store.purge.enable** option enables the purge server daemon process on **start-msg** startup. Various other **store.*** options control the maintenance queue. For more information on these options, see *Messaging Server Reference*.

The impurge Command

You can use the **impurge** command to manually purge unused cache records and message files in mailboxes when the purge server daemon process is not running, that is, when the **store.purge.enable** option is disabled.

Attempting to run the impurge command while the purge server daemon process is running results in the following error message in the Messaging Server **default** log:

```
[24/Feb/2012:14:47:15 +1100] hostname impurge[17986]: General Error: Could not get
purge session lock.
Possibly another impurge is running
```

See "impurge" for details.

54

Message Store Message Type Overview

This chapter provides the conceptual background for using Oracle Communications Messaging Server message types in the message store. See "Managing Message Types in the Message Store" for task information.

About Message Type

A unified messaging application can receive, send, store, and administer messages of many types, including text messages, voice mail, fax mail, image data, and other data formats. The message store allows you to define up to 63 different message types.

One method of managing messages by type is to organize the messages by their types into individual folders.

With the introduction of the message type feature, you do not have to maintain different message types in individual mailbox folders. Once you configure a message type, the message store can identify it, no matter where it is stored. Thus, you can store heterogeneous message types in the same folder. You also can perform the following tasks:

- Track the usage of message types
- Send notifications grouped by message type
- Set and administer different quotas for different message types, whether they are stored in the same folder or different folders
- Move messages from one folder to another, according to criteria configured uniquely for each message type
- Expire messages according to criteria configured for each message type

Planning the Message-Type Configuration

In a unified messaging application, data of heterogeneous formats are given standard internet message headers so that Messaging Server can store and manage the data. For example, when voice mail is sent to an end-user's phone, a telephone front-end system adds a message header to the incoming voice mail and delivers it to the message store.

To recognize and administer messages of different types, all components of the unified messaging system must use the same message-type definitions and the same header fields to identify the messages.

Before you configure the message store to support message types, you must:

- Plan which message types you intend to use

- Decide on the definition for each message type
- Decide which header field to use

For example, if your application includes phone messages, you can define this message type as "multipart/voice-message" and use the Content-Type header field to identify message types.

You would then configure the telephone front-end system to add the following header information to each phone message to be delivered to the message store:

Content-Type: multipart/voice-message

Next, you would configure the message store to recognize the **multipart/voice-message** message type, as described in the sections that follow.

Defining and Using Message Types

You define a message type by giving it a unique definition such as **multipart/voice-message**. By default, the message store reads the Content-Type header field to determine the message-type. If you prefer, you can configure a different header field to identify the message types.

The message store reads the Content-Type (or other specified) header field, ignoring case. That is, the message store accepts the header field as valid even if the header's combination of uppercase and lowercase letters differs from the expected combination.

The message store reads only the message-type name in the header field. It ignores additional arguments or options.

To define a message type, use the **msconfig** command to set values for the **store.messagetype** option. See "To Configure Message Types" for instructions,

Configuring a message type allows the message store to identify and manipulate messages of the specified type. It is the first, essential step in administering message types in a unified messaging application.

To take full advantage of the message-type features provided by the message store, you also should perform some or all of the following tasks:

- Configure a JMQ notification plug-in and write Message Queue clients for retrieving notifications that track the status of the message types
- Configure quota roots that apply to each message type
- Write expire rules and set LDAP attribute values to expire and purge messages according to message type

Message Types in IMAP Commands

When you configure the **store.messagetype.mtindex:*n*.flagname** option for a message type, you create a unique flag that identifies the message type. This flag cannot be modified by end users.

Messaging Server presents the message-type flag as a user flag to IMAP clients. Mapping the message type to a user flag allows mail clients to use simple IMAP commands to manipulate messages by message type.

For example, you can perform the following operations:

- Use the **IMAP FETCH FLAGS** command to display a message-type flag name as a user-defined flag to the client. See "Example 1: IMAP FETCH Session Based on the Message-Type msconfig Configurations" for a sample use of the **IMAP FETCH**

FLAGS command.

- Use a message-type flag as a keyword in an **IMAP SEARCH** command. See "Example 2: IMAP SEARCH Session Based on the Message-Type msconfig Configurations" for a sample use of the **IMAP SEARCH** command.

The message-type user flag is read only. It cannot be modified by IMAP commands.

The following examples assume that you configure the message-type **msconfig** options with the values shown here:

```
store.messagetype.enable 1
store.messagetype.mtindex:1.contenttype text/plain
store.messagetype.mtindex:1.flagname text
store.messagetype.mtindex:1.quotaroot text
store.messagetype.mtindex:2.contenttype multipart/voice-message
store.messagetype.mtindex:2.flagname voice_message
store.messagetype.mtindex:2.quotaroot voice
```

Example 1: IMAP FETCH Session Based on the Message-Type msconfig Configurations

The following IMAP session fetches messages for the currently selected mailbox:

```
2 fetch 1:2 (flags rfc822)
* 1 FETCH (FLAGS (\Seen text) RFC822 {164}
Date: Wed, 8 July 2006 03:39:57 -0700 (PDT)
From: bob.smith@example.com
To: john.doe@example.com
Subject: Hello
Content-Type: TEXT/plain; charset=us-ascii
* 2 FETCH (FLAGS (\Seen voice_message) RFC822 {164}
Date: Wed, 8 July 2006 04:17:22 -0700 (PDT)
From: sally.lee@example.com
To: john.doe@example.com
Subject: Our Meeting
Content-Type: MULTIPART/voice-message; ver=2.0
2 OK COMPLETED
```

In the preceding example, two messages are fetched, one text message and one voice mail.

The message-type flags are displayed in the format configured with the **store.messagetype.mtindex:***n***.flagname** option.

The Content-Type header fields identify the message types. The message-type names are displayed as they were received in the incoming messages. They use mixed uppercase and lowercase letters and include the message-type arguments such as **charset=us-ascii**.

Example 2: IMAP SEARCH Session Based on the Message-Type msconfig Configurations

The following IMAP session searches for voice messages for the currently selected mailbox:

```
3 search keyword voice_message
* SEARCH 2 4 6
3 OK COMPLETED
```

In the preceding example, messages 2, 4, and 6 are voice messages. The keyword used in the search is **voice_message**, the value of the **store.messagetype.mtindex:2.flagname** option.

55

Migrating Mailboxes to a New System

This chapter describes several ways to move or relocate mailboxes from one Oracle Communications Messaging Server host (mailhost) to another.

Tools Summary for Relocating Messaging Server Users to a New Mailhost

The following tools are available for relocating users from one mailhost to another:

- **rehostuser**, Enables you to move a Messaging Server user's mail store from one mailhost to another. The **rehostuser** utility also disconnects any active session, locks the store to ensure atomicity of the move from the user's perspective (no loss of data, flag change, and so on), changes the user's LDAP entry, flushes LDAP caches as necessary, and causes any queued mail to be rerouted to the new store. See "rehostuser" for more information.
- **imsbackup | imsrestore:** Manually backs up the account from one host and restores on another. Can be combined with **ssh** and "piped" across the network. You can also back up to a file and then access that file from the other host through NFS, or move the file in between.
- **imsimport:** Imports messages from a UNIX **/var/mail** format file into the Messaging Server message store.

Notes:

- If both the source and destination mailhosts are installed with at least Messaging Server 7, use the "rehostuser" utility. The **rehostuser** utility solves issues present in the other solutions, such as preventing users from accessing mail while it is being moved, making sure mail is held in the MTA during the move and redirected to the new message store, and so on.
- You can manually run "imsbackup" on one host and "imsrestore" on the other host. You can combine multiple users in one step. You can combine "imsbackup" with **ssh** or NFS, or you could use **gzip** and transfer the backup file from one host to the other by using **ftp**. This method is actually used within the **rehostuser** utility. However, **rehostuser** shields you from having to set the users' status, disconnect them, and so on. If you use **imsbackup** and **imsrestore** manually, you also need to deal with those details manually. But if you have a large amount of data to move and can afford for the MTA and IMAP user access to be down while it is being moved, this method might be more efficient.
- Similar to **imsrestore** and **imsbackup**, you can use "imsimport" to import UNIX **/var/mail** format files into the message store. If you are moving from a non-Oracle server and it does not have IMAP access capability, perhaps you can get it to export the folders in UNIX **/var/mail** format and then import them this way.

Migrating Mailboxes from an x86 Host to a SPARC Host

The message store data formats are architecture dependent. You cannot transfer any data components in "Classic Message Store Component Version Compatibilities" from one architecture to another directly. To migrate the message store from an x86 host to a SPARC host (or from a SPARC host to an x86 host), use **imsbackup** and **imsrestore**, or **rehostuser**.

Moving Mailboxes to Another Messaging Server While Online

You can migrate the message store from an older version of Messaging Server to a newer version, or move mailboxes from one Messaging Server message store to another, while remaining online. This procedure works for iPlanet Messaging Server 5.0 and later. You cannot move messages from prior versions of Messaging Server or a non-Oracle Communications Suite message store. Moving mailboxes while online have the following advantages and disadvantages.

Advantages

- You can move the mailboxes from the old source system to the new destination system without user involvement.
- This process is faster than any of the other processes.
- Re-linking is not required if you are moving an entire partition.
- Both Messaging Server systems remain active and online.
- You can migrate all the mailboxes on a messages store or a subset of those messages. This procedure allows for incremental migrations.

Disadvantages

- This method does not work with non-Oracle Communications Suite messaging servers.
- The users being migrated do not have access to their mailboxes until the migration of their own mailbox is complete.
- This method can be complex and time consuming.

Incremental Mailbox Migration While Online

Incremental migration provides numerous advantages for safely and effectively moving your message store to a different system or upgrading to a new system, incremental migration allows you to build a new back-end message store system alongside the old back-end message store. You can then test the new system, migrate a few friendly users, then test the new system again. Once you are comfortable with the new system and configuration, and you are comfortable with the migration procedure, you can start migrating real commercial users. These users can be split into discrete backup groups so that during migration, only members of this group are offline, and only for a short time.

Another advantage of on-line incremental migration is that you do not have to plan for a system-wide back out in case your upgrade fails. A back out is a procedure for reverting changes you have made to a system to return the system to the original working state. When doing a migration, you have to plan for failure, which means that for every step in the migration requires a plan to return your system back to its previous operational state.

The problem with offline migrations is that you can't be sure your migration is successful until you've completed all the migration steps and switched the service back on. If the system doesn't work and cannot be quickly fixed, you'll need a back out procedure for all the steps performed. This can be stressful and take some time, during which your users will remain offline.

With an online incremental migration you perform the following basic steps:

1. Build the new system alongside the old one so that both can operate independently.
2. Configure the old system for coexistence with the new.
3. Migrate a group of "friendly" users and test the new system and its coexistence with the old system.
4. Divide the users on the old system into groups and migrate group by group to the new one as desired.
5. Disassemble the old system.

Because both systems will coexist, you have time to test and get comfortable with the new system before migrating to it. If you do have to perform a back-out procedure, which should be very unlikely, you only have to plan for steps 2 and 4. Step 2 is easy to revert because you do not ever touch user's data. In step 4, the backout is to revert the user's state to active and their mailhost attribute back to the old host. No system-wide back out is required.

Online Migration Overview

Migrating mailboxes while remaining online is a straightforward process. One solution is to hold messages sent during the migration process in a *held* state and wait for the messages in the various channel queues to be delivered. However, messages can get stuck in queues because of system problems or because a particular user is over quota. In this case, you must address this situation before migrating the mailboxes.

You can take various measures to reduce the likelihood of lost messages and to verify that messages are not stuck in a channel queue, but at a cost of increased complexity of the procedure.

The order and necessity of steps in the procedure vary depending upon your deployment and whether every message addressed to every mailbox must not be lost. This section describes the theory and concepts behind the steps. It is incumbent on you to understand each step and decide which to take and in which order, given your specific deployment. Following is an overview of the process of moving mailboxes. This process might vary depending upon your deployment.

1. Block user access to the mailboxes being moved.
2. Temporarily hold messages addressed to the mailbox being moved.
3. Verify that messages are not stuck in the channel queues.
4. Change the user's mailhost attribute to the new mailbox location.
5. Move the mailboxes to the new location.
6. Release held mail to be delivered to the new mailbox and enable incoming messages to be delivered to the migrated mailboxes.
7. Examine the old message store to see if any messages were delivered after the migration.

8. Unblock user access to mailbox.

To Migrate User Mailboxes from One Messaging Server to Another While Online

The requirements for this type of migration are as follows:

- **stored** should be running on both the source (old) and destination (new) messaging servers.
- The source system and destination system must be able to route messages to each other if both systems will operate in co-existence. This is needed, for instance, so that delivery status notification messages can be generated on the destination system and get delivered to the source system.

> **Note:** Some steps apply only if you are upgrading the messaging server from an earlier version to a later version. These steps might not apply if you are only migrating mailboxes from one message store to another. The steps that apply to migrating entire systems are noted.

1. On the source system, split your user entries to be moved into equal backup groups by using the **backup-groups.conf** file. This step is in preparation for the mailbox migration, Step 8, that occurs later in this procedure. See "To Create Backup Groups" for detailed instructions. You can also place the user names into files and use the -u option in the **imsbackup** command.
2. Notify users to be moved that they will not have access to their mailboxes until the move is completed. Ensure that users to be moved are logged out of their mail systems before the data move occurs. See "Monitoring User Access to the Message Store" for more information.
3. Set the authentication cache timeout to 0 on the back-end message store and MMP systems, and **alias_entry_cache_timeout** option to 0 on the MTAs.

   ```
   msconfig set mta.alias_entry_cache_timeout 0
   ```

   a. On the back-end message stores containing the mailboxes to be moved, set the authentication cache timeout to 0.

      ```
      msconfig set base.authcachettl -0
      ```

      This step and Step 7 (changing **mailUserStatus** to **hold**) immediately prevents users from accessing their mailboxes during migration.

   b. On all MMPs, set the LDAP and authentication cache timeout to 0. For the IMAP proxy and POP proxy, set both **ldapcachettl** and **authcachettl** to 0. For example:

      ```
      msconfig
      msconfig> set imapproxy.authcachettl 0
      msconfig# set imapproxy.ldapcachettl 0
      msconfig# set popproxy.authcachettl 0
      msconfig# set popproxy.ldapcachettl 0
      msconfig# write
      ```

   c. On any Messaging Server host that contains an MTA that inserts messages into mailboxes that are to be migrated, set the **alias_entry_cache_timeout** option to 0. Messaging Server hosts that run an MTA that inserts messages into the migrating mailboxes are typically the back-end message store. However, if the system is using LMTP, then that system is the inbound MTA.

Check your configuration to make sure. Resetting the **alias_entry_cache_timeout** option forces the MTA to bypass the cache and look directly at the LDAP entry so that intermediate channel queues (for example, the **conversion** or **reprocess** channels) see the new **mailUserStatus** (**hold**) of the users being moved rather then the out-of-date cached information.

d. Restart the systems on which the caches were reset. You must restart the system for these changes to take place. See "Stopping and Starting Messaging Server" for instructions.

4. Ensure that both your source Messaging Server and destination Messaging Server are up and running. The source Messaging Server must be able to route incoming messages to the new destination server.

5. Change the LDAP attribute **mailUserStatus** on all user entries whose mailboxes will be moved from **active** to **hold**. Changing the attribute holds incoming messages in the **hold** queue and prevents access to the mailboxes over IMAP, POP, and HTTP. Typically, users are moved in groups of users. If you are moving all the mailboxes of a single domain, you can use the **mailDomainStatus** attribute. For more information on **mailUserStatus**, see the **mailUserStatus attribute** in *Schema Reference*.

6. Make sure that messages addressed to mailboxes being migrated are not stuck in the **ims-ms** or **tcp_lmtp*** channel queues (if LMTP has been deployed). Use the following commands to see if messages exist in the channel queue directory tree and in the *held* state (to see **.HELD** files) addressed to a user to be migrated:

```
imsimta qm directory -to=<user_address_to_be_migrated> -directory_tree
imsimta qm directory -to=<user_address_to_be_migrated> -held -directory_tree
```

   If there are messages in the queue, run these same commands later to see if the MTA has dequeued them. If there are messages that are not being dequeued, then you must address this problem before migrating. This should be a rare occurrence, but possible causes are recipient mailboxes being over quota, mailboxes being locked perhaps because users are logged in and moving messages, the LMTP backend server is not responding, network or name server problems, and so on).

7. Change the LDAP attribute **mailHost** in the user entries to be moved as well as in any mail group entries. Use the **ldapmodify** command to change the entries to the new mail server. Use the **ldapmodify** that comes with Messaging or Directory Server. Do no use the Oracle Solaris **ldapmodify** command.

   - You only need to change the **mailHost** attribute in the mail group entry if the old mail host is being shut down. You can either change this attribute to the new mail host name or just eliminate the attribute altogether. It is optional for mail groups to have a **mailHost.** Having a **mailHost** means that only that host can do the group expansion. Omitting a **mailHost** (which is the more common case) means all MTAs can do the group expansion. Mail group entries do not have mailboxes to be migrated and typically do not even have the **mailhost** attribute. For more information on **mailhost**, see the **mailHost attribute** in *Schema Reference*.

8. Move the mailbox data from the source Messaging Server message store to the destination Messaging Server message store and record the time when started. Back up the mailboxes with the **imsbackup** utility and restore them to the new Messaging Server with the **imsrestore** utility. For example, to migrate mailboxes from a Messaging Server system called **oldmail.example.com** to **newmail.example.com**, run the following command on **oldmail.example.com**:

```
<server-root>/bin/msg/store/bin/imsbackup -f- instance/group | rsh
```

```
newmail.example.com /opt/sun/comms/messaging64/lib/msg/imsrestore.sh -f- -c y
-v 1
```

You can run multiple concurrent "imsbackup" and "imsrestore" sessions (one per group) to maximize the transfer rate into the new message store. See also "Backing Up and Restoring the Message Store" for more information.

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

Note: Record the timestamp of when **imsbackup** is run for later delivery validation.

---

9. (*Conditional Step for System Upgrades*) If your mailbox migration is part of the process of upgrading from an earlier version of Messaging Server to the current version, set this current version of Messaging Server to be the new default Messaging Server for the system. Change the DNS **A** record of **oldmail.example.com** to point to **newmail.example.com** (the server responsible for domain(s) previously hosted on **oldmail.example.com**).
10. Enable user access to the new message store. Set the LDAP attribute **mailUserStatus** or **mailDomainStatus**, if applicable, to whatever value it had been before it was changed to **hold** (for example,**active**).
11. Release the messages in the *held* state on all source Messaging Servers. Any system that may be holding incoming messages needs to run the following command to release all the user messages:

    ```
    imsimta qm release -channel=hold -scope
    ```

    where *scope* can be **all**, which releases all messages; **user**, which is the user ID; or **domain** which is the domain where the user resides.
12. Reset the authentication cache timeout and the **alias_entry_cache_timeout** option to the default or desired values and restart the system. At this point, you've migrated all the user mailboxes that need to be migrated. Before proceeding, make sure that no new entries in LDAP have been created with the old system as the **mailhost**, and if some have, migrate them. Also, make sure that no such entries can be created by modifying the provisioning systems. You also want to change the **preferredmailhost** attribute to the name of the new mail host. For back-end messages stores, set authentication cache timeout to 900 as follows:

    ```
    msconfig set base.authcachettl 900
    ```

    For the IMAP proxy and POP proxy, use the **msconfig** command set both **ldapcachettl** and **authcachettl** to 900. For example:

    ```
    msconfig
    msconfig> set imapproxy.authcachettl 900
    msconfig# set imapproxy.ldapcachettl 900
    msconfig# set popproxy.authcachettl 900
    ```

```
msconfig# set popproxy.ldapcachettl 900
msconfig# write
```

For MTAs, set the **alias_entry_cache_timeout** option to 600.

```
msconfig set mta.alias_entry_cache_timeout 600
```

You must restart the system for these changes to take place. See "Stopping and Starting Messaging Server" for instructions.

13. Ensure that the user clients are pointing to the new mail server. After the upgrade finishes, have the users point to the new server through their mail client program (in this example, users would point to **newmail.example.com** from **oldmail.example.com**). An alternative is to use a Messaging Multiplexor (MMP), which obviates the need to have users point their clients directly to the new mail server. The MMP gets that information from the **mailHost** attribute that is stored in the LDAP user entries and automatically redirects the client to the new server.
14. After everything works, verify that no messages were delivered to the old message store after the migration. Go to the old message store and run **mboxutil -l** to list the mailboxes. Check the last message delivery timestamp. If a message was delivered after the migration timestamp (the date stamp when you ran the **imsbackup** command), then migrate those messages with a backup and restore command. Because of the preparatory steps provided, it would be exceedingly rare to see a message delivered after migration. Theoretically, a message could be stuck in a queue for the number of days or hours specified by the **notices** channel options. See the discussion on setting notification message delivery intervals in *Messaging Server Reference*.
15. Remove duplicate messages on the new message store, run the **relinker** command. This command might free disk space on the new message store.
16. Remove the old messages from the store you migrated from and delete users from the database on the old store. Run the **mboxutil -d** command. See "mboxutil" for more information.

To Move Mailboxes Using an IMAP Client

This procedure can be used anytime messages need to be migrated from one messaging server to a different messaging server. Consider the advantages and disadvantages before moving mailboxes using this method.

Advantages

- This method can be used to migrate from a non-Oracle Communications host to the Messaging Server host. It can also be used to move mailboxes from one physical server to a different physical server.
- After you set up the new mail server or message store, responsibility for moving mailboxes to the new system is left to users.
- The process for moving mailboxes is relatively simple.
- User access to mailboxes does not have to be disabled.

Disadvantages

- Requires that both the old and new systems be simultaneously running and accessible to users.
- Cumulatively, this method takes longer to move mailboxes than other methods.

- Responsibility for moving mailboxes to the new system is left to users.
- The size of the new message store will be significantly larger than the old message store until the re-linking operation is performed.

1. Install and configure the new Messaging Server.
2. Set **store.relinker.enable** to **1**. This reduces the message store size on the new system caused by duplicate storage of identical messages.
3. Provision users on the new Messaging Server. You can use Delegated Administrator to do this. As soon as users are provisioned on the new system, newly arriving mail is delivered to the new INBOX.
4. Have users configure their mail client to view both new and old Messaging Server mailboxes. This may involve setting up a new email account on the client. See mail client documentation for details.
5. Instruct users to drag folders from their old Messaging Server to their new Messaging Server.
6. Verify with users that all mailboxes are migrated to the new system, then shut down the user account on the old system.

To Move Mailboxes by Using the imsimport Command

This procedure is specifically used to move mailboxes from UNIX **/var/mail** format folders into a Messaging Server message store. However, if the Messaging Server host from which you are migrating can convert the IMAP message stores to UNIX **/var/mail** format, then you can use the **imsimport** command to migrate messages to Messaging Server. Consider the advantages and disadvantages before moving mailboxes using this method.

Advantages

- You have complete responsibility for moving mailboxes from the old system to the new system. Users do not have to do anything.

Disadvantages

- This method takes longer to move mailboxes than the other non-IMAP methods.
- Users access to mailboxes must be disabled while mailboxes are being moved.
- The size of the new message store will be significantly larger than the old message store until the re-linking operation is performed.

1. Install and configure the new Messaging Server.
2. Set **store.relinker.enabled** to **1**. This reduces the message store size on the new system caused by duplicate storage of identical messages.
3. Provision users on the new Messaging Server if needed. You can use Delegated Administrator to do this. Do not switch over to the new system yet.
4. Disable user access to both the new and old messaging store. Set the **mailUserStatus** LDAP attribute to **hold**. User's mail is sent to the hold queue and access to the mailbox over IMAP, POP, and HTTP is disallowed. MTA and Message Access Servers on the store server must comply with this requirement. This setting overrides any other **mailDeliveryOption** settings.
5. If the mail store from the existing mail server is not already in the **/var/mail** format, convert the mail store to **/var/mail** files. Refer to the third-party mail server documentation.

6. Run the **imsimport** command. For example:

   ```
   imsimport -s /var/mail/joe -d INBOX -u joe
   ```

   See the "imsimport" for details.

7. Enable user access to the message store. Set the **mailUserStatus** LDAP attribute to **active**.
8. Enable user access to the new messaging store.
9. Shut down the old system.

Migrating Mailboxes from Microsoft Exchange Server to Oracle Communications Messaging Server

See the discussion about migrating mailboxes from Microsoft Exchange Server to Oracle Communications Messaging Server (Doc ID 2632831.1) on the My Oracle Support Web site:

https://support.oracle.com/portal/

56

Monitoring Disk Space

This chapter describes Oracle Communications Messaging Server configuration options for monitoring disk and partition usage and for generating warnings about disk space availability.

Disk Space Overview

Inadequate disk space or inadequate space within a disk partition are among the most common causes of mail server problems and failure. Typical causes of inadequate space are:

- Message store quotas are not enforced and the message store outgrows the disk space available for a partition.
- Over-long MTA message queues.
- Log files that are not adequately monitored and kept within defined limits. (Note that there are several log files such as LDAP, MTA, and Message Access, and that each of these log files can be stored on different disks.)

Symptoms of Insufficient Disk Space

Symptoms of insufficient disk-space are:

- MTA queues overflow and reject SMTP connections.
- Messages remain in the **ims-master** queue and are not delivered to the message store.
- Log files overflow.

If a message store partition fills up, message access daemons can fail and message store data can be corrupted. Message store maintenance utilities such as **imexpire** and **reconstruct** can repair the damage and reduce disk usage. However, these utilities require additional disk space, and repairing a partition that has filled an entire disk can cause down time.

Monitoring Disk Space

Depending upon the system configuration you may need to monitor various disks and partitions. For example, MTA queues may reside on one disk/partition, message stores may reside on another, and log files may reside on yet another. Each of these spaces will require monitoring and the methods to monitor these spaces may differ.

Messaging Server provides specific methods for monitoring message store disk usage and preventing partitions from filling up all available disk space.

You can take the following steps to monitor the message store's use of disk space:

- Set options to monitor message store disk usage
- Lock message store partitions when a disk-usage threshold is reached

Monitoring the Message Store

You can monitor message store disk usage by configuring the following attributes with the **msconfig** utility:

- **alarm.system:***alarmtype***.statinterval** specifies the length of time, in seconds, between disk availability checks. For example, to set the system to monitor disk space every 600 seconds, enter the following command:

  ```
  msconfig set alarm.system:diskavail.statinterval 600
  ```

- **alarm.system:***alarmtype***.threshold** specifies a percentage of disk space that must be available or a warning is generated. For example, it is recommended that disk-space usage should not exceed 75%; correspondingly, the following command generates a warning whenever the amount of disk space available falls below 25%:

  ```
  msconfig set alarm.system:diskavail.threshold 25
  ```

- **alarm.system:***alarmtype***.warninginterval** specifies an interval, in hours, between the repetition of disk availability alarms. For example, the following command sets an interval of one hour between one disk availability warning and another.

  ```
  msconfig set alarm.system:diskavail.warninginterval 1
  ```

- **alarm.system:***alarmtype***.description** specifies a description of the disk availability alarm. For example, the following sets the description of an availability alarm for a message-queue alarm:

  ```
  msconfig set alarm.system:diskavail.description "Percentage message-queue
  partition diskspace available"
  ```

Monitoring Message Store Partitions

By default, partition monitoring is in effect so that when a message-store partition uses more than a specified percentage of available disk space, the partition is locked and any incoming messages are held in the MTA message queue. Two **msconfig** options control partition monitoring:

- **checkdiskusage**

  checkdiskusage enables partition monitoring. It takes a boolean value; the default is 1 (monitoring is enabled).

- **diskusagethreshold**

  diskusagethreshold specifies a disk-usage threshold beyond which the partition is locked. It takes an integer value from 1 to 99; the default value is 99.

As a partition approaches the threshold specified in **diskusagethreshold**, the message-store daemon checks the partition with increasing frequency, ranging from once every 100 minutes to once every minute. If disk usage goes higher than the threshold specified in **diskusagethreshold**, the message-store daemon:

- Locks the partition; incoming messages are held in the MTA message queue and are not delivered to mailboxes in the message store partition until it is unlocked.

- Logs a message to the default log file.
- Sends an email notification to the postmaster. (You can change the recipient of the notification by setting the **msconfigalarm.noticercpt** option.)

In setting the **diskusagethreshold** option, specify a usage percentage that is low enough to allow time for repartitioning or assigning more disk space to the local message store. For example, if a partition fills up disk space at a rate of 2 percent per hour and it takes an hour to allocate additional disk space, set the disk-usage threshold to a value lower than 98 percent.

57

Protecting Mailboxes from Deletion or Renaming

You might want to protect some mailboxes from deletion or modification except by the administrator. The following procedure describes how to do this.

If someone other than an administrator attempts to delete, modify, or rename a protected mailbox, the error message "**mailbox is pinned**" is displayed.

- Set the **store.pin** configuration option by using the following format:

  ```
  msconfig set store.pin mailbox1%mailbox2%mailbox3
  ```

  where *mailbox1*, *mailbox2*, and *mailbox3* are the mailboxes to be protected (you can use spaces in mailbox names), and **@** is the separator between each mailbox.

In this example, the mailboxes specified in *mailbox1@mailbox2@mailbox3* are not per user mailboxes, as in **user/user1/Drafts**. Instead, the mailboxes specified are for all users using a certain directory, such as **Drafts**. An administrator can therefore prevent users from renaming or deleting the **Drafts** or **Backup** folder for all users by running:

```
msconfig set store.pin Drafts%Backup
```

58

Reducing Message Store Size Due to Duplicate Storage

This chapter describes how Oracle Communications Messaging Server uses the relinker feature to reduce message store size due to duplicate storage.

When a message is sent to multiple recipients, that message is placed in each recipient's mailbox. Some messaging systems store separate copies of the same message in each recipient's mailbox. By contrast, Messaging Server strives to retain a single copy of a message regardless of the number of mailboxes in which that message resides. It does this by creating hard links to that message in the mailboxes containing that message.

When other messaging systems are migrated to the Messaging Server, these multiple message copies may be copied over with the migration process. With a large message store, this means that a lot of messages are duplicated unnecessarily. In addition, multiple copies of the same message can be accumulated in normal server operation, for example, from IMAP append operations or other sources.

Messaging Server provides a command called **relinker** that removes the excess message copies and replaces them with hard links to a single copy.

> **Note:** The **relinker** feature is intended to repair the situation where the normal single-copy nature of the message store has become broken for some reason. You should only need to use the relinker if you have done something which could have caused duplicate messages to become individual copies instead of the normal single-copy. This feature is not the normal way the store normally accomplishes its single-copy feature. You should not need to keep the real-time relinker feature enabled for long periods of time. You should not need to use the relinker command on an ongoing basis. You should only need this feature if you have done (or will soon be doing) something which would break the single-copy feature of the store. See "How the Message Store Works" for more information about single-copy.

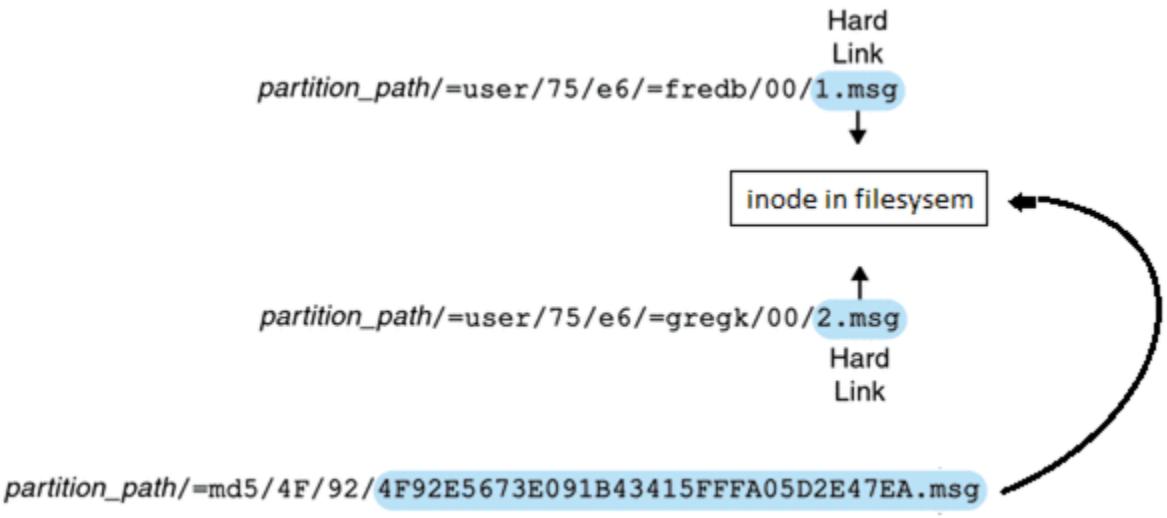
Relinker Overview

The relinking function can be run in the command or realtime mode. When the **relinker** command is run, it scans through the message store partitions, creates or updates the MD5 message digest repository (as hard links), deletes excess message files, and creates the necessary hard links.

The digest repository consists of hard links to the messages in the message store. It is stored in the directory hierarchy **partition_path**/=md5. This directory is parallel to the

user mailbox hierarchy _ partition_path_/=user (see "Classic Message Store Directory Layout"). Messages in the digest repository are uniquely identified by their MD5 digest. For example, if the digest for fredb/00/1.msg is 4F92E5673E091B43415FFFA05D2E47EA, then *partition*/=user/*hashdir*/*hashdir*/=fredb/00/1.msg is linked to *partition*/=md5/_hashdir_/*hashdir*/4F92E5673E091B43415FFFA05D2E47EA.msg. If another mailbox has this same message, for example, *partition_path*/=user/*hashdir*/*hashdir*/gregk/00/17.msg, that message will also be hard linked to *partition_path*/=md5/4F/92/4F92E5673E091B43415FFFA05D2E47EA.msg. This is shown in Figure 58–1.

Figure 58–1 Message Store Digest Repository



For this message, the link count will be three. If both messages are deleted from the mailboxes of fredb and gregk, then the link count will be one and the message can be purged.

The **relinker** process can also be run in the realtime mode for similar functionality. See "Using Relinker in the Realtime Mode" for details.

Using relinker in the Command Line Mode

The **relinker** scans through a message store partition, creates or updates the MD5 message repository (as hard links) and deletes excess message files. After **relinker** scans a store partition, it outputs statistics on the number of unique messages and size of the partition before and after relinking. To run more quickly on an already hashed store, **relinker** only computes the digest of the messages not yet present in =md5. It also has an option to erase the entire digest repository (which doesn't affect the user mailboxes).

The syntax for the command is as follows:

relinker [-*ppartitionname*] [-d]

where *partitionname* specifies the partition to be processed (default: all partitions) and **-d** specifies that the digest repository be deleted. Sample output is shown below:

```
relinker
```

```
Processing partition: primary
Scanning digest repository...
Processing user directories.............................
-------------------------------------------------------
Partition statistics Before After
-------------------------------------------------------
Total messages 4531898 4531898
Unique messages 4327531 3847029
Message digests in repository 0 3847029
Space used 99210Mb 90481Mb
Space savings from single-copy 3911Mb 12640Mb
-------------------------------------------------------
relinker -d
Processing partition: primary
Purging digest repository...
-------------------------------------------------------
Partition statistics Before After
-------------------------------------------------------
Message digests in repository 3847029 0
-------------------------------------------------------
```

relinker can take a long time to run, especially for the first time if there are no messages are in the repository. This is because it has to calculate the digest for every message (if the **relinker** criteria is configured to include all messages-see "Configuring Relinker" for information on configuring **relinker** criteria.) For example, it could take six hours to process a 100 Gigabyte message store. However, see "Using Relinker in the Realtime Mode" if run-time relinking is enabled

If the **relinker** command line mode is used exclusively, and not the run-time option, it is necessary to purge the digest repository (**=md5**), otherwise messages purged in the store (**=user**) will not become available disk space since they still have a link in the digest repository (they become orphaned). If you are just performing a one-time optimization of the store-for example after a migration-you can run **relinker** once, then delete the entire repository with **relinker -d**. For repeated purging during migration, it is sufficient to just run the **relinker** command repeatedly, since each time it runs it also purges the expired or orphaned messages from the repository.

It is safe to run multiple instances of **relinker** in parallel with each processing a different partition (using the **-p** option). Messages are only relinked inside the same partition.

Using Relinker in the Realtime Mode

The **relinker** function can be enabled in the realtime mode by setting the **msconfig** option **store.relinker.enable** to **1**. Using **relinker** in the realtime mode will compute the digest of every message delivered (or restored, IMAP appended, and so forth) which matches the configured **relinker** criteria ("Configuring Relinker"), then look in the repository to see if that digest is already present. If the digest is present, it creates a link to it in the destination mailbox instead of creating a new copy of the message. If there is no digest, it creates the message and adds a link to it in the repository afterwards.

stored scans the digest repositories of each partition and purges the messages having a link count of 1, or which don't match the **relinker** criteria. The scan is done one directory at a time over a configurable time period. This is so that the I/O load is evenly distributed and does not noticeably impact other server operations. By default the purge cycle is 24 hours, which means messages can still be present on the disk for up to 24 hours after they have been deleted from the store or have exceeded the

configured maximum age. This task is enabled when the **relinker** realtime mode is enabled.

Configuring Relinker

Table 58–1 shows the options used to set **relinker** criteria.

Table 58–1 relinker msconfig options

Option	Description
store.relinker.enable	Enables real-time relinking of messages in the append code and **stored** purge. The **relinker** command-line tool may be run even if this option is off. However since **stored** will not purge the repository, **relinker -d** must be used for this task. Turning this option on affects message delivery performance in exchange for the disk space savings. Default: 0
store.relinker.maxage	Maximum age in hours for messages to be kept in the repository, or considered by the **relinker** command-line. -1 means no age limit, that is, only purge orphaned messages from the repository. For **relinker** it means process existing messages regardless of age. Shorter values keep the repository smaller thus allow **relinker** or **stored** purge to run faster and reclaim disk space faster, while longer values allow duplicate message relinking over a longer period of time, for example, when users copy the same message to the store several days apart, or when running a migration over several days or weeks. Default: 24
store.relinker.minsize	Minimum size in kilobytes for messages to be considered by run-time or command-line **relinker**. Setting a non-zero value gives up the relinker benefits for smaller messages in exchange for a smaller repository. Default: 0
store.relinker.purgecycle	Approximate duration in hours of an entire **stored** purge cycle. The actual duration depends on the time it takes to scan each directory in the repository. Smaller values will use more I/O and larger values will not reclaim disk space as fast. 0 means run purge continuously without any pause between directories. -1 means don't run purge in **stored** (then purge must be performed using the **relinker -d** command). Default: 24

59

Specifying Administrator Access to the Message Store

This chapter describes how to grant store privileges to the message store for your Oracle Communications Messaging Server installation. See "Managing Message Store Partitions and Adding Storage" for conceptual information.

Overview of Message Store Administrators

Message store administrators can view and monitor user mailboxes and specify access control for the message store. Store administrators have proxy authentication privileges to any service (POP, IMAP, HTTP, or SMTP), which means they can authenticate to any service using the privileges of any user. These privileges allow store administrators to run certain utilities for managing the store.

> **Note:** Other users might also have administrator privileges to the store. For example, some administrators may have these privileges.

Also, see "Protecting Mailboxes from Deletion or Renaming" for more information.

Adding an Administrator Entry

To add an administrator entry at the command line, enter:

```
msconfig set store.admins adminlist
```

where *adminlist* is a space-separated list of administrator IDs. If you specify more than one administrator, you must enclose the list in quotes. In addition, the administrator must be a member of the Service Administrator Group, in the LDAP user entry: **memberOf: cn=Service Administrators,ou=Groups,o=usergroup**. You must restart **imapd** for the system to recognize the change in **store.admins**.

Modifying or Deleting an Administrator Entry

To modify or delete an existing entry in the message store Administrator UID list at the command line, use the same command:

```
msconfig set store.admins adminlist
```

where *adminlist* is a space-separated list of administrator IDs who should be included in the modified list. If you specify more than one administrator, you must enclose the list in quotes. In addition, the administrator must be a member of the Service

Administrator Group, in the LDAP user entry: **memberOf: cn=Service Administrators,ou=Groups,o=usergroup**. You can delete members from the list, but the modified list must contain at least one administrator ID.

You must restart **imapd** for the system to recognize the change in **store.admins**.

60

Constructing Valid Message Store UIDs and Folder Names

This chapter describes valid constructions for Oracle Communications Messaging Server message store UIDs and folder names. Note that folder and mailbox are used synonymously.

Message Store User ID

The message store user ID is a mail user's unique identifier in the message store. In the default domain, this is the same as the user's LDAP **uid** attribute. In hosted domains, this is *uid@domain* where *uid* is the **uid** LDAP attribute and *domain* is the canonical domain name.

Message Store Mailbox Name for Commands

Some message store commands require that you specify a mailbox name. The required form of the name is **user/***userid/mailbox* where *userid* is the message store user ID (see "Message Store User ID") and **mailbox** is a user's mailbox. Specifying INBOX sometimes implies all the user's mailboxes in the message store. For example, the following command removes the INBOX and all the folders of user **joe**.

```
mboxutil -d user/joe/INBOX
```

Note that in the context of message stores, folders and mailboxes are synonymous.

Valid UIDs

Valid and invalid **UID** characters are controlled separately by the MTA and message store mechanisms. That means **UID** character limitations are specified by the union of MTA and message store limitations. The following characters and strings are invalid as **UIDs** in the message store:

- % ? * & / : \
- ASCII values less than 20 or greater than 7E hexidecimal (see man ascii)
- A leading '-' is prohibited because it is reserved for negative rights
- A leading 'group=' is prohibited because it is reserved for group IDs
- The following UIDs are reserved: 'anonymous' 'anybody' 'anyone' and 'anyone@domain'
- The maximum supported length for a UID is 127 bytes

The following characters are invalid in *UIDs* in the MTA:

```
<space> $ ~ = # * + % ! @ , { } ( ) / \ < > ; : " ` [ ] & ?
```

The list of characters forbidden by the MTA can be modified by setting the **ldap_uid_invalid_chars** option with a string of the forbidden characters using decimal ASCII values, however, you are strongly advised not to change the default constraint. The default setting is as follows and reflect the characters listed above:

```
ldap_uid_invalid_
chars=32,33,34,35,36,37,38,40,41,42,43,44,47,58,59,60,61,62,63,64,91,92,93,96,123,
125,126
```

Valid mail folder names. The following characters are invalid as folder names:

% * ? and ASCII values less than 20 or greater than 7E hexidecimal (see **man ascii**).

In addition, folder names must be valid MUTF-7 sequences. (Note that MUTF-7 names should use Unicode Normalization Form C to comply with RFC 5198.)

61

Message Store Automatic Failover with Database Replication

This chapter describes Oracle Communications Messaging Server database replication and classic message store failover.

Overview of Message Store Database Replication

Berkeley Database provides support for building high availability (HA) applications based on replication. Message store database replication uses Berkeley Database HA facilities and low cost NFS storage devices to build an HA message store.

The **mboxlist** database is a transactional database. Database changes are written to the transaction logs. You can replicate the database by transporting the transaction log records from one site to another. Berkeley Database HA architecture supports single writer (master) and multiple reader replication. You must perform all database updates on the master. Replicas are available for read only activity. When the master fails, an election takes place, and one of the replicas will take over as master.

Either all replicas are on SPARC CPUs or all replicas are on x86 CPUs. Replication between these two types of CPUs is not supported by Messaging Server.

The database replication and message store failover feature requires that replicas be on the same platform. Either all replicas are SPARC or all replicas are x86_64. It is also best practice for all replicas to be homogeneous (same hardware/CPU/OS), unless you are in the middle of a rolling hardware upgrade.

A message store replication group consists of one or more message store nodes. The message store nodes typically run on different physical hosts. The **mboxlist** database is replicated on every node. You can store the database locally. You store the mailbox partitions on remote storage devices running NFS servers. The NFS file systems are always mounted on all of the nodes in a replication group. You can configure the message store nodes with one or more remote hosts. If remote hosts are configured, the message store contacts a remote host to retrieve the replication group data on start up. If a master has not been established in a group, an election is called. A priority value is assigned to a node. When an election is held, the node with the most up-to-date log record and the highest priority becomes the new master. A node with priority 0 cannot be elected.

When the master fails, the replicas will automatically hold an election to select a new master. You are responsible for monitoring the master. The automatic failover facility will redirect incoming connections to a new master. The message store replicas run in read only mode. Any attempt to modify a mailbox on a replica returns an error. You can perform read only operations such as **mboxutil -l** and **imsbackup** on the replicas.

You can install Multiple message store nodes on the same host in different zones. Each message store node must have a unique **base.listenaddr**.

Berkeley Database maintains an internal database to keep track of the replication group data. Starting the first node in a replication group for the first time initializes the database. To start up the first node the first time, run **start-msg -m**. Database transaction commits blocks until it either receives enough acknowledgments, or the acknowledgment timeout expires. You can configure the acknowledgment policy and timeout.

A two-site replication group is particularly vulnerable to duplicate masters when there is a disruption to communications between the sites. Two-site replication is disabled by default. You can enable it with the **store.dbreplicate.twosites** option. When this option is disabled, the message store cannot take over as master if the original master fails in a replication group with only two sites. In the event this happens, the message store cluster will be unavailable for write access.

When a replication-aware client, such as the MMP (POP or IMAP), LMTP client or ENS subscriber connects to a replication group, it has to be aware of the hosts in the replication group so if one is down it can attempt to connect to the next one. This is presently accomplished by setting the **proxy:***repgroup***.storehostlist** option to the same value on all replication-aware clients and back-ends. These clients also remember the last host to which they connected in the replication group so it's not necessary to failover on every connection. In addition, replication-aware clients that perform write operations need to connect to the replication group master. This is accomplished by having the back-end servers refer the client to the replication group master. Details about how each replication-aware client works are as follows:

- **IMAP server**

  Advertises OK login referrals (RFC 2221) that point to the master. Third-party use of such referrals is supported. Note that login referrals are also provided when a user logs in into an IMAP server that does not contain their **INBOX** (as may be necessary to access shared folders). Login referrals of the latter form will point to a host that is not in the **storehostlist** for the replication group.

- **POP server**

  Advertises referrals with a **SYS/REFER/***hostname* extended error on login. This is an Oracle extension to the protocol based on RFC 3206. Third-party use of such referrals is supported.

- **LMTP server**

  The referral is indicated via a private protocol extension in the LMTP greeting. Oracle does not support third-party use of our LMTP server.

- **ENS server**

  No changes have been made.

- **Message Store ENS publisher**

  Publishes to all available hosts in the replication group for the current store by default. The recommended deployment for ENS with store failover is to have **enpd** running on every message store master and replica.

- **ENS and JMQ publishers for Message Store**

  The **hostname** attribute will use the replication group name instead of the local host name.

- **ENS C Client API**

If **storehostlist** is configured, it will perform failover and cache the last successful host in the host list.

- **MMP IMAP client**

  Performs failover, caches the last successful master, and follows referrals.

- **MMP POP client**

  Performs failover, caches the last successful master, and follows referrals.

- **MTA LMTP client**

  When the **affinitylist** channel option is set, this performs failover, caches the last successful master, and follows referrals.

- **MTA BURL IMAP client**

  Performs failover and caches the last successful host. As this is read-only, it does not follow referrals.

- **Shared folder IMAP client in imapd**

  Performs failover, caches the last successful master, and follows referrals.

- **mshttpd IMAP client**

  Performs failover, caches the last successful master, and follows referrals.

- **Glassfish MQ (aka JMQ)**

  No changes other than the **hostname** attribute mentioned above. Support for Glassfish MQ is deprecated.

Configuration Options

Topics in this section include:

Configuration Options

Table 61–1 shows the configuration options (only supported in Unified Configuration), their descriptions, data types, and defaults.

Table 61–1 Configuration Options

Option	Description	Data Type	Default
store.dbreplicate.enable	Enable database replication	boolean	0
store.dbreplicate.port	Replication port number	integer	55000 Note that **storehostlist** does not support non-default port numbers.
store.dbreplicate.dbremotehost	Remote host name list; this option is deprecated in 8.0.1.	*host*[:*port*] [*host*[:*port*]]...	In 8.0.1 and later this defaults the value of the **proxy:***repgroup***.storehost list** with the current host omitted from the list.
store.dbreplicate.dbpriority	Host priority	integer	100

Table 61–1 (Cont.) Configuration Options

Option	Description	Data Type	Default
store.dbreplicate.ackpolicy	Replication acknowledgment policy	0=none, 1=one, 2=one peer, 3=quorum, 4=all peer, 5=all available, 6=all clients	3
store.dbreplicate.acktimeout	Replication acknowledgment timeout	number of seconds	1 second
store.dbreplicate.twosites	Enable two sites replication group	boolean	0
proxy:*repgroup***.storehostlist**	Sets list of hosts in each store replication group for all relevant servers in the deployment. The preferred master should be listed first.	*host* [*host*]	Value of LDAP **mailHost** attribute for a user.
proxy:*repgroup***.imapport**	The port used to connect to IMAP for this replication group.	unsigned 16-bit integer	Value of **base.proxyimapport** option.
proxy:*repgroup***.imapadmin**	The administrative user name used when connecting to this replication group.	non-empty-string	Value of **base.proxyadmin** option.
proxy:*repgroup***.imapadminpass**	The administrative password used to connect to this replication group.	password	Value of **base.proxyadminpass** option.

Command-line Utilities

- The **imcheck** subsystem option prints the database replication statistics. See "imcheck" for additional information.

  To print the database replication statistics, run the following command:

  ```
  imcheck -s rep
  ```

  Note: The **imcheck -s** command is only valid for classic message store.

- The **start-msg -m** option starts the message store as a replication master. See *Messaging Server Refernce Guide* for additional information on the **start-msg** command.

  To start the message store as a replication master, run the following command:

  ```
  start-msg -m
  ```

- The **stored -d** *site* option removes a replication site from the replication database. See "stored" for additional information.

  To delete an **mboxlist** replication site called **grumpy** from the cluster, run the following command:

  ```
  stored -d grumpy
  ```

Configuring Message Store Database Replication

The following configuration examples show you how to configure message store database replication.

To Configure a Three Node Cluster for HA

The following example sets up a cluster with three electable nodes (**huey**, **dewey**, and **louie** at **example.com**). This example assumes LMTP has been configured on these back-ends. The message store partition is on a shared storage mounted at **/zfssa/primary**.

On **huey.example.com**:

```
msconfig set store.dbreplicate.enable 1
msconfig set proxy:cluster1.storehostlist "huey.example.com dewey.example.com
louie.example.com"
msconfig set partition:primary.path /zfssa/primary
msconfig set task:snapshot.enable 0
msconfig set task:snapshotverify.enable 0
start-msg -m
```

On **dewey.example.com**:

```
msconfig set store.dbreplicate.enable 1
msconfig set proxy:cluster1.storehostlist "huey.example.com dewey.example.com
louie.example.com"
msconfig set partition:primary.path /zfssa/primary
msconfig set task:snapshot.enable 0
msconfig set task:snapshotverify.enable 0
start-msg
```

On **louie.example.com**:

```
msconfig set store.dbreplicate.enable 1
msconfig set proxy:cluster1.storehostlist "huey.example.com dewey.example.com
louie.example.com"
msconfig set partition:primary.path /zfssa/primary
msconfig set task:snapshot.enable 0
msconfig set task:snapshotverify.enable 0
start-msg
```

In addition, the **storehostlist** has to be set on all front-end servers as well, and the LMTP client has to be configured on the front-ends with the **affinitylist** channel option. Due to the complexity of setting up LMTP, it is recommended to copy the example recipe file **LMTPBackendFailover.rcp** and modify it with appropriate settings for the back-end stores.

To Change the DB Replication Local Instance Port

The replication group info is maintained by BDB. **store.dbreplicate.port** is for the local site only.

To change the port number on one node with the cluster running:

1. Stop the message store.

   ```
   stop-msg store
   ```

2. Remove the local site from the group.

   ```
   stored -d
   ```

3. Change the port number on the local site in Unified Configuration.

```
msconfig set store.dbreplicate.port <newport>
```

Or in legacy configuration.

```
configutil -o store.dbreplicate.port -v <newport>
```

4. Restart the message store

```
start-msg
```

5. Run **imcheck -s rep** on the other sites. You should see the new port for this site.

```
imcheck -s rep
```

Note: The **imcheck -s** command is only valid for classic message store.

Message Store Automatic Failover

This section describes Messaging Server's message store automatic failover feature and its configuration. It contains the following sections:

- Basic Requirements
- Overview of Message Store Automatic Failover
- Configuring Message Store Automatic Failover

Basic Requirements

In order to use message store automatic failover, the following is necessary.

- Messaging Server must be running in Unified Configuration mode.
- Messaging Server must be deployed using LMTP.

Overview of Message Store Automatic Failover

The message store automatic failover feature is useful for customers who already have 24/7 operators in their machine room.

The basic model is that all Messaging Server hosts in the deployment need to be manually configured with an ordered list of hostnames for each mailStore. Each hostname corresponds to a separate product installation, but in a given mailStore list all the hosts must use a shared disk (for example, NFS, filer) for the product data, but have a configuration that is largely identical except for the "base.hostname" setting. The first host in the list is the primary host for that mailStore. The primary host is running and the secondary hosts are not running (on standby).

The MMP, LMTP client, and imapd shared folder support will now automatically use the secondary host for requests related to the primary mailHost; there is no need to refresh or restart these services for this to happen.

Note that a standby hostname does not mean it is necessary to have unused hardware. If multiple IP addresses are used on the same server, then it can support multiple installations. However, if the primary host is dedicated and the secondary host is shared for a mailStore, then the service response time will be reduced when automatic

failover happens. Sites need to consider how much spare capacity is needed to service users when there is a hardware outage.

To enable safety during rolling version upgrades where a node is deliberately shutdown during upgrade, we recommend having at least 3 hostnames associated with each mailHost.

Configuring Message Store Automatic Failover

For this example, assume the following hosts are in a deployment:

LDAP mailStore "store1.example.com"

store1a.example.com primary

store1b.example.com

store1c.example.com

LDAP mailStore "store2.example.com"

store2a.example.com primary

store2b.example.com

store2c.example.com

mta.example.com - an MTA configured to use LMTP

mmp.example.com - an MMP

We recommend that customers use Unified Configuration recipes to set up automatic failover to avoid typographical errors when configuring multiple machines by running a recipe similar to the following on all machines in the deployment.

```
set_option("proxy:store1\\.example\\.com.storehostlist",
"store1a.example.com store1b.example.com store1c.example.com");
set_option("proxy:store2\\.example\\.com.storehostlist",
"store2a.example.com store2b.example.com store2c.example.com");
```

Save the recipe above to a plain text file, for example, automatic-failover.rcp

Some extra configuration need be added to LMTP server and all the clients communicating to LMTP server.

To Configure the LMTP Server

1. Add the **automatic-failover.rcp** file configuration to the existing LMTP server configuration. Or, if you prefer, there is also a sample recipe **LMTPBackendFailover.rcp** that configures a backend LMTP server for use with failover. If you want to use this, you must copy the recipe script and manually add in to your LMTP client IP addresses and mailstore proxy information. This is available in the Messaging Server installed location *MessagingServer_home*/**lib/recipes/LMTPBackendFailover.rcp**.
2. Run the recipe script on all back-end machines in the deployment by executing the following the command.

   ```
   msconfig run manual-failover.rcp OR LMTPBackendFailover.rcp
   ```

3. If you are running a compiled a configuration, recompile by running:

   ```
   imsimta cnbuild
   ```

4. Start the Messaging Server by running:

```
start-msg
```

To Configure the Client

1. Add the **automatic-failover.rcp** file configuration to all clients' configurations. For the LMTP client, you must add some extra configuration.
2. Run the recipe script on all client machines (MMP, LMTP client, and do on) in the deployment by executing the following command:

   ```
   msconfig run automatic-failover.rcp
   ```

3. For the LMTP client only, you must set the affinity list channel option on the LMTP client channel:

   ```
   msconfig set channel:tcp_lmtpcs.affinitylist
   imsimta cnbuild
   stop-msg
   start-msg
   ```

4. Stop the Messaging Server Client by running:

   ```
   stop-msg mmp or mta
   ```

5. Start the Messaging Server Client by running:

   ```
   start-msg mmp or mta
   ```

62

Administering Message Store Database Snapshots (Backups)

This chapter describes the tasks for administering Oracle Communications Messaging Server database snapshots. See "Message Store Automatic Recovery On Startup" for conceptual information on database snapshots.

The primary message store database is critical to smooth operation of the message store. You must always ensure that a snapshot (or backup) of the database is available. If the active database becomes damaged, restarting services allows the message store **stored** process to swap in the best snapshot and enable services to come back on demand.

To Specify Message Store Database Snapshot Interval and Location

Before doing these tasks, see "Message Store Database Snapshot Interval and Location". Also, descriptions of the **configutil** options described in this page are in *Messaging Server Reference*.

A database snapshot is a hot backup (dynamic backup) of the message store database. The system makes this copy without any locking, and requires that both the database files and transaction logs so the snapshot can be "recovered" into a valid copy of the database.

By default, snapshots are scheduled with the **imdbverify -s** command to run at specific times.

```
local.schedule.snapshot.enable = "1"
local.schedule.snapshot = "0 2 * * * bin/imdbverify -s -m"
```

By default, the **imdbverify -s** command takes a database snapshot at 2 a.m. (This command uses UNIX crontab format: minute hour day-of-month month-of-year day-of-week command arguments.) The **-m** option is used to verify the snapshot. The **-m** option is not required. See "Message Store Database Snapshot Recovery and Verification" for more information.

Database snapshots are located in the following base directory:

```
local.store.snapshotpath = "dbdata/snapshots"
```

Change this directory to a different disk than the disk used by the primary database, for both performance and recovery reasons.

You configure the number of snapshots retained over time with the following option:

```
local.store.snapshotdirs = "3"
```

Each snapshot requires as much disk space as the entire database and transaction logs at any given time.

Take enough snapshots such that you both have a recent copy, and copies that go back a day or two, to be sure you can find a database not affected by any odd system problems that are not immediately discovered.

Message Store Database Snapshot Recovery and Verification

The message store has been enhanced to continuously recover archived log files into an up-to-date backup copy of the message store database. If the actual database becomes unusable, then the message store automatically uses this backup database. Having an up-to-date database backup provides the next level of recovery and stability for the message store.

The system automatically runs **imdbverify -m** as specified for rolling backups, and **imdbverify -s -m** as specified under snapshots.

If the verification process detects any errors, the errors are written to the **default** log. Errors in the **default** log mean not only that the snapshot failed, but they could also be pointing to a problem with the active database. (However, at this time, not all verification errors indicate a live database problem.)

Message Store Database Snapshot Rolling Backup

The message store rolling backup operates as a specially designated snapshot, where each transaction log is added to the snapshot and recovered every few minutes to provide a more up-to-date backup. For this reason, always enable snapshots, and rolling backup will be enabled by default.

Rolling backup requires the following three configuration settings, which are enabled by default:

```
local.store.rollingdbbackup = "yes"

local.schedule.snapshotverify.enable = 1

local.schedule.snapshotverify = "1,3,5,7,9,11,13,15,17,19,
                                 21,23,25,27,29,31,33,35,37,
                                 39,41,43,45,47,49,51,53,55,
                                 57,59 * * * * bin/imdbverify -m"
```

In this configuration:

- **local.store.rollingdbbackup** enables rolling backup. This means the log archive function running under the **stored** daemon copies the database transaction logs to the rolling snapshot instead of removing them every minute.
- **local.schedule.snapshotverify** verifies addition, which is required to continually roll the log files into the snapshot.

> **Note:** Should a rolling backup fail any of its verifies, each side of the process declares the rolling backup invalid and cleans up the logs. Rolling backup then restarts after the next snapshot is put in place. Rolling backup relies on an initial snapshot taken by the normal snapshot process.

Message Store Database Recovery

When the message store services are started, the message store process **stored** decides if the current database is damaged, and if so, replaces it with the best snapshot. The best snapshot is printed in the logs and any recovery actions are also printed in the **default** log.

63

Classic Messaging Server and Tiered Storage Overview

This chapter describes the operation of the Oracle Communications Messaging Server classic message store, its performance characteristics, and how to plan for and allocate store partitions. Additionally, this document describes next generation best practices to meet the storage needs of both ISPs and enterprises.

Overview of Messaging Server Storage

For traditional ISPs that provide web-based email services, the rules of engagement have changed. Thanks to companies such as Google, which can now offer consumers multiple gigabytes of email storage space with unlimited retention, the threat is clear: ISPs, with their much smaller storage allotments (50-100 Mbytes) and automatic purging of messages older than 90 days, need to stay competitive by providing storage and retention capabilities similar to Google, or lose out.

The rules have also changed for enterprises and the corporate messaging market, which are being forced to comply with new regulations requiring email retention for ever longer periods of time. In fact, some companies are faced with the requirement of saving every incoming and outgoing email message forever. The storage requirements for such scenarios can be staggering, to say the least.

As if there weren't enough problems already for ISPs and enterprises, email by its nature is a very I/O intensive application where transaction speed is critical to customer satisfaction. In increasing storage capacity, businesses must ensure that email services maintain acceptable performance levels and do so in a cost-effective manner. To stay competitive, businesses understand that they must purchase large numbers of disk drives to satisfy this new appetite for storage space. Purchasing fiber channel drives is typically cost prohibitive, so ISPs and enterprises will be looking to less expensive alternatives, such as Serial Advanced Technology Attachment (SATA) drives. The challenge with SATA is that in order to reduce cost and provide higher capacity, performance is sacrificed. And there's the dilemma: both ISPs and enterprises need to dramatically increase their email storage capacity while at the same time maintaining acceptable performance levels without "breaking the bank." Customers are under immense legal and financial pressure to find a solution to this problem. The good news is that Oracle can deliver an excellent storage solution for Messaging Server deployments in the form of the Sun StorageTek 6540 Array.

The Messaging Server message store is one of the highest IOPs applications that exists. In the past, customers have kept all portions of the message store on their highest performing, most expensive disks. Fortunately, you can distributed the message store components onto different performing disks to create a cost-effective but high performing application.

Care must be taken to keep the highest IOP portion of the message store (the database and its store partition indexes) on high performance disks. The message store can generate up to 15+ IOPS per message delivered, typically many small, random writes, and is extremely sensitive to response times. If response time diminishes, it can have a cascading effect through the application. Because of the high IOP needs, the message store is ideal for Oracle's StorageTek 6540 controller.

Message Store and ZFS

Oracle's Communications Suite Deployment Engineering group has performed extensive testing with ZFS and measured its impact on the message store. The Messaging Group believes that in the future most Messaging Server customers will be using the ZFS file system. ZFS changes the workload characteristics on the file system, so that there are fewer I/Os, but the I/Os are bigger. ZFS enables read rates to diminish somewhat, whereas it enables write rates to diminish much more. In addition, ZFS can perform snapshotting and compression, which enhances the ability to back up the application. ZFS is also now supported by Oracle Solaris Cluster software.

How the Message Store Works

The message store is a dedicated data store for the delivery, retrieval, and manipulation of Internet mail messages. The message store works with the IMAP4 and POP3 client access servers to provide flexible and easy access to messaging. The message store also works through the HTTP server (mshttpd) to provide messaging capabilities to Convergence clients in a web browser. The message store is organized as a set of folders or user mailboxes. The folder or mailbox is a container for messages. Each user has an INBOX where new mail arrives.

Each IMAP or Webmail user can also have one or more folders where mail can be stored. Folders can contain other folders arranged in a hierarchical tree. Mailboxes owned by an individual user are private folders. Private folders can be shared at the owner's discretion with other users on the same message store. Messaging Server supports sharing folders across multiple stores by using the IMAP protocol. There are two general areas in the message store, one for user files and another for system files. In the user area, the location of each user's INBOX is determined by using a two-level hashing algorithm. Each user mailbox or folder is represented by another directory in its parent folder. Each message is stored as a file. When there are many messages in a folder, the system creates hash directories for that folder. Using hash directories eases the burden on the underlying file system when there are many messages in a folder. In addition to the messages themselves, the message store maintains an index and cache of message header information and other frequently used data to enable clients to rapidly retrieve mailbox information and do common searches without the need to access the individual message files.

A message store can contain many message store partitions for user files. A message store partition is contained by a file system volume. As the file system becomes full, you can create additional file system volumes and message store partitions on those file system volumes to store new users.

Note: If a message store partition fills up, users on the partition are not able to store additional messages. Address this problem by using one or more of the following approaches:

- Reducing the size of user mailboxes
- If you are using volume management software, adding additional disks
- Creating additional partitions and moving mailboxes to the new partitions

The message store maintains only one copy of each message per partition. This is sometimes referred to as a single-copy message store. When the message store receives a message addressed to multiple users or a group or distribution list, it adds a reference to the message in each user's INBOX. Rather than saving a copy of the message in each user's INBOX, the message store avoids the storage of duplicate data. The individual message status flag (seen, read, answered, deleted, and so on) is maintained per folder for each user.

The system area contains information on the entire message store in a database format for faster access and no loss of service. The information in the system area can be reconstructed from the user area. Messaging Server contains a database snapshot function. When needed, you can quickly recover the database to a known state.

Messaging Server also has fast recovery, so that in case of database corruption, you can shut down the message store and bring it back immediately without having to wait for a lengthy database reconstruction.

Messaging Server Disk Throughput

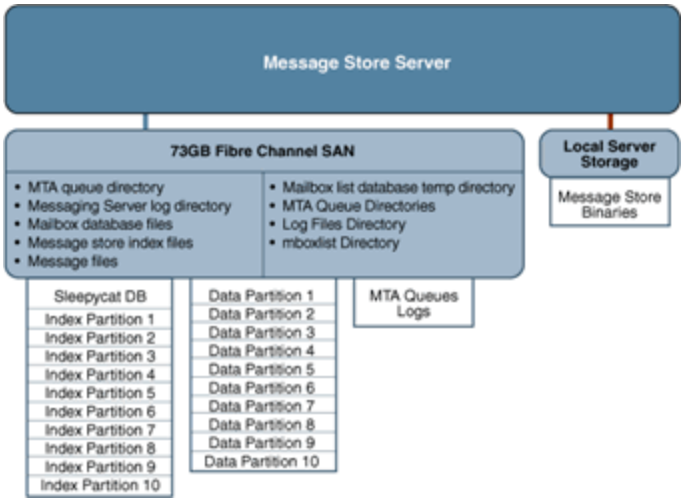
Disk throughput is the amount of data that your system can transfer from memory to disk and from disk to memory. The rate at which this data can be transferred is critical to the performance of Messaging Server. To create efficiencies in your system's disk throughput:

- Consider your maintenance operations, and ensure you have enough bandwidth for backup. Backup can also affect network bandwidth particularly with remote backups. Private backup networks might be a more efficient alternative.
- Carefully partition the store and separate store data items (such as tmp and db) to improve throughput efficiency.
- Ensure the user base is distributed across RAID (Redundant Array of Independent Disks) environments in large deployments
- Stripe data across multiple disk spindles in order to speed up operations that retrieve data from disk.
- Allocate enough CPU resources for RAID support, if RAID does not exist on your hardware.

Note: Measure disk I/O in terms of IOPS (total I/O operations per second) not bandwidth. You need to measure the number of unique disk transactions the system can handle with a very low response time (less than 10 milliseconds).

Typically, most customers deploy their entire message store on the highest performing disk as shown in Figure 63–1.

Figure 63–1 Typical Disk Storage with Message Store



Messaging Server Disk Capacity

When planning server system disk space, be sure to include space for operating environment software, Messaging Server software, and message content and tracking. Be sure to use an external disk array if availability is a requirement. For most systems, external disks are required for performance because the internal system disks supply no more than four spindles. For the message store partitions, the storage requirement is the total size of all messages plus 30 percent overhead. In addition, user disk space needs to be allocated. Typically, this space is determined by your site's policy.

Disk Sizing for MTA Message Queues

The purpose of the Messaging Server MTA Queue is to provide a transient store for messages waiting to be delivered. Messages are written to disk in a persistent manner to maintain guaranteed service delivery. If the MTA is unable to deliver the message, it retries delivery. At some point, if delivery is still unsuccessful, the MTA no longer tries to send the message and returns it to the sender.

MTA Message Queue Performance

Sizing the MTA Message Queue disks is an important step for improving MTA performance. The MTA's performance is directly tied to disk I/O above any other system resource. This means that you should plan on a disk volume that consists of multiple disk spindles which are concatenated and striped by using a disk RAID system. End users are quickly affected by the MTA performance. As users press the SEND button on their email client, the MTA does not fully accept receipt of the message until the message has been committed to the MTA Message Queue. Therefore, improved performance on the MTA Message Queue results in better response times for the end-user experience.

MTA Message Queue Availability

SMTP services are considered a guaranteed message delivery service. This is an assurance to end users that the messaging server will not lose messages that the

service is attempting to deliver. When you architect the design of the MTA Message Queue system, all effort should be made to ensure that messages will not be lost. This guarantee is usually made by implementing redundant disk systems through various RAID technologies.

MTA Message Queue Available Disk Sizing

The MTA Message Queue grows excessively if one of the following conditions occurs:

- The site has excessive network connectivity issues
- The MTA configuration is holding on to messages too long
- Valid problems are occurring with those messages (not covered in this document)
- Message stores are operationally down, for example, for maintenance
- Remote target sites are unavailable or overwhelmed

Performance Considerations for a Message Store Architecture

Message store performance is affected by a variety of factors, including:

- Disk I/O
- Inbound message rate (also known as message insertion rate)
- Message sizes
- Login rate (POP/IMAP/HTTP)
- Transaction rate for IMAP and HTTP
- Concurrent number of connections for the various protocols
- Network I/O
- Use of SSL

The preceding factors list the approximate order of impact to the message store. Most performance issues with the Message Storage arise from insufficient disk I/O capacity. Additionally, the way in which you lay out the store on the physical disks can also have a performance impact. For smaller standalone systems, it is possible to use a simple stripe of disks to provide sufficient I/O. For most larger systems, segregate the file system and provide I/O to the various parts of store.

Note: In a deployment using LMTP, the MTA queue is almost always unused.

Messaging Server Directories (General Recommendations for Storage)

Messaging Server uses six directories that receive a significant amount of input and output activity. Because these directories are accessed very frequently, you can increase performance by providing each of those directories with its own disk, or even better, providing each directory with a Redundant Array of Independent Disks (RAID).

The six directories are:

- MTA Queue Directory
- Messaging Server Log Directory
- Mailbox Database Files

MTA Queue Directory

Recommendation: Can be located on slower disks or on shared NAS

In this directory, many files are created, one for each message that passes through the MTA channels. After the file is sent to the next destination, the file is then deleted. The directory location can be changed by making the queue directory a symlink.

Default location of the MTA Queue directory: **/var/opt/sun/comms/messaging/queue**

Messaging Server Log Directory

Recommendation: Can be located on slower disks

This directory contains log files which are constantly being appended with new logging information. The number of changes depend on the logging level set. The directory location is controlled by the **msconfig** option ***.logfile.logdir** (Unified configuration) or the **configutil** option **logfile.***.logdir** (legacy configuration), where * can be a log-generating component such as admin, default, http, imap, or pop. The MTA log files can be changed by making that directory a symlink.

Default location of the Messaging Server log directory: *DataRoot***/log**

Mailbox Database Files

Recommendation: Keep on a fast disk

These files require constant updates as well as cache synchronization. Put this directory on your fastest disk volume. These files are always located in the **/var/opt/sun/comms/messaging/store/mboxlist** directory.

Message Store Index Files

Recommendation: Keep on fast disk

These files contain meta information about mailboxes, messages, and users. By default, these files are stored with the message files. The msconfig option **partition:***.**path** (Unified configuration) or the **configutil** option **store.partition.***.**path** (legacy configuration), where ***** is the name of the partition, controls the directory location. If you have the resources, put these files on your second fastest disk volume.

Default location of the message store index file:
/var/opt/sun/comms/messaging/store/partition/primary

Message Files

Recommendation: Can be located on slower disks

These files contain the messages, one file per message. Files are frequently created, never modified, and eventually deleted. By default, they are stored in the same directory as the message store index files. The location can be controlled with the **msconfig** option **partition:***partition_name***.messagepath** (Unified configuration) or the **configutil** option **store.partition.***partition_name***.messagepath** (legacy configuration), where *partition_name* is the name of the partition. Some deployments might use a single message store partition called **primary** specified by the **store.partition.primary.path**. Large sites might have additional partitions that can be

specified with **store.partition.***partition_name***.messagepath**, where *partition_name* is the name of the partition.

Default location of the message files:
/var/opt/sun/comms/messaging/store/partition/primary

> **Note:** To separate the message files from the index files (enabling tiered storage), the **store.partition.*** and **.messagepath** options are key. These options must be correctly configured to put the message files on a SATA file system when you create the partition.

Mailbox List Database Temporary Directory

Recommendation: Can be located on fast disk

This is the directory used by the message store for all temporary files. To maximize performance, this directory should be located on the fastest file system. For Solaris OS, use the **msconfig** or the **configutil** options to configure the **store.dbtmpdir** (same in both Unified configuration and legacy configuration) variable to a directory under **tmpfs**, for example, **/tmp/mboxlist**.

Default location of the mailbox list database:
/var/opt/sun/comms/messaging/store/mboxlist

The following sections provide more detail on Messaging Server high-access directories.

Additional Information: MTA Queue Directories

In non-LMTP environments, the MTA queue directories in the message store system are also heavily used. LMTP works such that inbound messages are not put in MTA queues but directly inserted into the store. This message insertion lessens the overall I/O requirements of the message store machines and greatly reduces use of the MTA queue directory on message store machines. If the system is standalone or uses the local MTA for Webmail sends, significant I/O can still result on this directory for outbound mail traffic. In a two-tiered environment using LMTP, this directory is lightly used, if at all. In prior releases of Messaging Server, on large systems this directory set needs to be on its own stripe or volume.

MTA queue directories should usually be on their own file systems, separate from the message files in the message store. The message store has a mechanism to stop delivery and appending of messages if the disk space drops below a defined threshold. However, if both the log and queue directories are on the same file system and keep growing, you will run out of disk space and the message store will stop working.

Additional Information: Log Files Directory

The log files directory requires varying amounts of I/O depending on the level of logging that is enabled. The I/O on the logging directory, unlike all of the other high I/O requirements of the message store, is asynchronous. For typical deployment scenarios, do not dedicate an entire LUN for logging. For very large store deployments, or environments where significant logging is required, a dedicated LUN is in order.

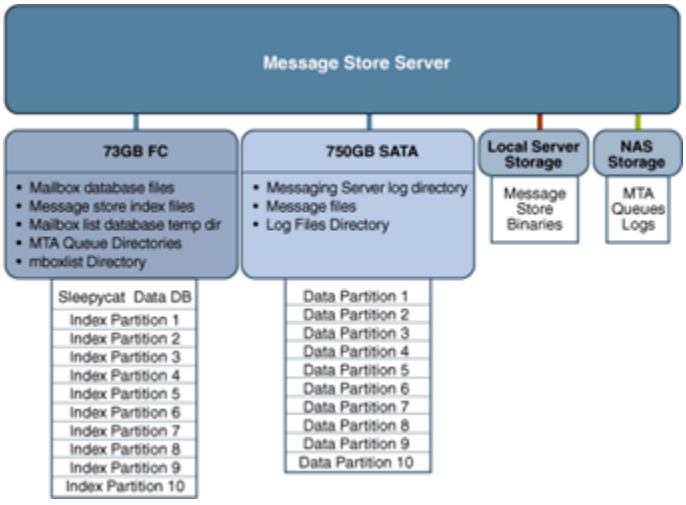
Note: In almost all environments, you need to protect the message store from loss of data. The level of loss and continuous availability that is necessary varies from simple disk protection such as RAID5, to mirroring, to routine backup, to real time replication of data, to a remote data center. Data protection also varies from the need for Automatic System Recovery (ASR) capable machines, to local HA capabilities, to automated remote site failover. These decisions impact the amount of hardware and support staff required to provide service.

Additional Information: mboxlist Directory

The **mboxlist** directory is highly I/O intensive but not very large. The **mboxlist** directory contains the databases that are used by the stores and their transaction logs. Because of its high I/O activity, and due to the fact that the multiple files that constitute the database cannot be split between different file systems, you should place the **mboxlist** directory on its own stripe or volume in large deployments. This is also the most likely cause of a loss of vertical scalability, as many procedures of the message store access the databases. For highly active systems, this can be a bottleneck. Bottlenecks in the I/O performance of the mboxlist directory decrease not only the raw performance and response time of the store but also impact the vertical scalability.

For systems with a requirement for fast recovery from backup, place this directory on Solid State Disks (SSD) or a high performance caching array to accept the high write rate that an ongoing restore with a live service will place on the file system. Figure 63–2 depicts this configuration.

Figure 63–2 Next Generation Storage with Message Store



Multiple Store Partitions

The message store supports multiple store partitions. Place each partition on its own stripe or volume. The number of partitions that should be put on a store is determined by several factors. The obvious factor is the I/O requirements of the peak load on the server. By adding additional file systems as additional store partitions, you increase the available IOPS (total IOs per second) to the server for mail delivery and retrieval. In most environments, you get more IOPS out of a larger number of smaller stripes or LUNs than a small number of larger stripes or LUNs. With some disk arrays, it is possible to configure a set of arrays in two different ways. You can configure each

array as a LUN and mount it as a file system. Or, you can configure each array as a LUN and stripe them on the server. Both are valid configurations.

However, multiple store partitions (one per small array or several partitions on a large array striping sets of LUNs into server volumes) are easier to optimize and administer. Raw performance, however, is usually not the overriding factor in deciding how many store partitions you want or need. In corporate environments, it is likely that you need more space than IOPS. Again, it is possible to software stripe across LUNs and provide a single large store partition. However, multiple smaller partitions are generally easier to manage. The overriding factor of determining the appropriate number of store partitions is usually recovery time.

Recovery times for store partitions fall into several categories:

- First, the **fsck** command can operate on multiple file systems in parallel on a crash recovery caused by power, hardware, or operating system failure. If you are using a journaling file system (highly recommended and required for any HA platform), this factor is small.
- Secondly, backup and recovery procedures can be run in parallel across multiple store partitions. This parallelization is limited by the vertical scalability of the mboxlist directory as the message store uses a single set of databases for all of the store partitions. Store cleanup procedures (**expire** and **purge**) run in parallel with one thread of execution per store partition.
- Lastly, mirror or RAID re-sync procedures are faster with smaller LUNs. There are no hard and fast rules here, but the general recommendation in most cases is that a store partition should not encompass more than 10 spindles. The size of drive to use in a storage array is a question of the IOPS requirements versus the space requirements. For most residential ISP POP environments, use "smaller drives." Corporate deployments with large quotas should use "larger" drives. Again, every deployment is different and needs to examine its own set of requirements.

Setting Disk Stripe Width

When setting disk striping, the stripe width should be about the same size as the average message passing through your system. A stripe width of 128 blocks is usually too large and has a negative performance impact. Instead, use values of 8, 16, or 32 blocks (4, 8, or 16 kilobyte message respectively).

MTA Performance Considerations

MTA performance is affected by many factors including, but not limited to:

- Disk performance
- Use of SSL
- The number of messages/connections inbound and outbound
- The size of messages
- The number of target destinations/messages
- The speed and latency of connections to and from the MTA
- The need to do spam or virus filtering
- The use of Sieve rules and the need to do other message parsing (like use of the conversion channel)

The MTA is both CPU and I/O intensive. The MTA reads from and writes to two different directories: the queue directory and the logging directory. For a small host

(four processors or fewer) functioning as an MTA, you do not need to separate these directories on different file systems. The queue directory is written to synchronously with fairly large writes. The logging directory is a series of smaller asynchronous and sequential writes. On systems that experience high traffic, consider separating these two directories onto two different file systems. In most cases, you will want to plan for redundancy in the MTA in the disk subsystem to avoid permanent loss of mail in the event of a spindle failure. (A spindle failure is by far the single most likely hardware failure.) This implies that either an external disk array or a system with many internal spindles is optimal.

MTA and RAID Trade-offs

There are trade-offs between using external hardware RAID controller devices and using JBOD arrays with software mirroring. The JBOD approach is sometimes less expensive in terms of hardware purchase but always requires more rack space and power. The JBOD approach also marginally decreases server performance, because of the cost of doing the mirroring in software, and usually implies a higher maintenance cost. Software RAID5 has such an impact on performance that it is not a viable alternative. For these reasons, use RAID5 caching controller arrays if RAID5 is preferred.

Background: Communication Services Logical Architectures Overview

You can deploy Communications Services in either a single-tiered or two-tiered logical architecture. Deciding on your logical architecture is crucial, as it determines which machine types you will need, as well as how many. In general, enterprise corporate deployments use a single-tiered architecture while internet service providers (ISPs) and telecommunications deployments use a two-tiered architecture. However, as with all generalities, the exceptions prove the rule. Small ISPs might just as well deploy on a single machine, and larger, centralized enterprises might deploy in a two-tiered architecture for many of the same reasons that ISPs do. As more and more corporations look to offer ease of access to employees working remotely, their deployments will begin to look more and more like an ISP.

Two-tiered Logical Architecture

In a two-tiered logical architecture, the data stores communicate through front-end processes. In the case of Messaging Server, this means MMPs and MTAs are residing on separate machines from the data store processes. A two-tiered architecture enables the mail store to offload important and common tasks and focus on receiving and delivering mail.

There might be some level of cohabitation with other services. For example, you could have the Calendar store and the message store on the same machine. Similarly, you could have the calendar front end on the MMP machine. In a two-tiered logical architecture, Directory Server is usually a complex deployment in its own right, with multi-master and replication to a set of load-balanced consumer directories.

Benefits of a Two-tiered Architecture

All services within the Communications Services offering rely on network capabilities. A two-tiered architecture provides for a network design with two separate networks: the public (user-facing) network, and the private (data center) network. Separating your network into two tiers provides the following benefits:

- **Hides Internal Networks.** By separating the public (user-facing) network and the private (data center) network, you provide security by hiding the data center information. This information includes network information, such as IP addresses and host names, as well as user data, such as mailboxes and calendar information.
- **Provides Redundancy of Network Services.** By provisioning service access across multiple front-end machines, you create redundancy for the system. By adding redundant messaging front-end servers, you improve service uptime by balancing SMTP requests to the available messaging front-end hosts.
- **Limits Available Data on Access Layer Hosts.** Should the access layer hosts be compromised, the attackers cannot get to critical data from the access hosts.
- **Offloads Tasks to the Access Layer.** By enabling the access layer to take complete ownership of several tasks, the number of user mailboxes on a message store increases. This is useful because the costs of both purchase and maintenance are much higher for store servers than for access layer machines (the second tier). Access layer machines are usually smaller, do not require large amounts of disk (see "MTA Performance Considerations") and are rarely backed up. A partial list of features that are offloaded by the second tier includes:
  - Denial of service protection
  - SSL
  - Reverse DNS
  - UBE (spam) and virus scanning
  - Initial authentication - Authentications to the message store should always succeed and the directory servers are more likely to have cached the entry recently.
  - LMTP - With support for LMTP between the MTA relays and the message stores, SMTP processing is offloaded and the need to do an additional write of the message into the MTA queues on the message stores is eliminated.
- **Simplifies End-user Settings in Client Applications.** By using a two-tiered architecture, end users do not have to remember the physical name of hosts that their messaging and calendar applications connect to. The Access-Layer Application hosts provide proxies to connect end users to their assigned messaging or calendar data center host. Services such as IMAP are connected to the back-end service using LDAP information to identify the name of the user's mailbox host. For calendar services, the calendar front-end hosts provide a calendar lookup using the directory server to create a back-end connection to the user's assigned calendar store host.

  This capability enables all end users to share the same host names for their client settings. For example, instead of remembering that their message store is host-a, the user simply uses the setting of mail. The MMP provides the proxy service to the user's assigned message store. You need to provide the DNS and load balancing settings to point all incoming connections for mail to one (or more) MMPs.

  By placing Messaging Server into two tiers, more than one Messaging Server back-end server can be used.

  An additional benefit of this proxy capability provides geographically dispersed users to leverage the same client application settings regardless of their physical location. Should a user from Europe visit California, the user will be able to connect to the immediate access server in California. The user's LDAP information will tell the access server to create a separate connection on the user's behalf to the

user's message store located in Europe. Lastly, this enables you to run a large environment without having to configure user browsers differently, simplifying user support. You can move a user's mailbox from one mail store to another without contacting the user or changing the desktop.

- **Reduces Network HTTP Traffic on the Data Center.** The Messaging Server front-end greatly reduce HTTP traffic to the data center network. HTTP provides a connectionless service. For each HTML element, a separate HTTP request must be sent to the mail or calendar service. These requests can be for static data, such as an image, style sheets, JavaScript files, or HTML files. By placing these elements closer to the end user, you reduce network traffic on the back-end data center.

Horizontal Scalability Strategy

Scalability is critical to organizations needing to make the most cost-effective use of their computing resources, handle peak workloads, and grow their infrastructure as rapidly as their business grows. Keep these points in mind:

- How the system responds to increasing workloads: what performance it provides, and as the workload increases, whether it crashes or enables performance to gracefully degrade.
- How easy it is to add processors, CPUs, storage, and I/O resources to a system or network to serve increasing demands from users.
- Whether the same environment can support applications as they grow from Low-end systems to mid-range servers and mainframe-class systems.

When deployed in a two-tiered architecture, Messaging Server is meant to scale very effectively in a horizontal manner. Each functional element can support increased load by adding additional machines to a given tier.

Scaling Front-end and Back-end Services

In practice, the method for scaling the front-end and back-end services differs slightly. For Tier 1 elements, you start the scaling process when traffic to the front end grows beyond current capacity. You add relatively low cost machines to the tier and load balance across these machines. Thus, load balancers can precede each of the Tier 1 service functions as overall system load, service distribution, and scalability requirements dictate.

For Tier 2 elements, you start the scaling process when the back-end services have exceeded user or data capacity. As a general rule, design the Tier 2 services to accommodate just under double the load capacity of the Tier 1 services. For example, for an architecture designed for 5,000 users, the Tier 1 front-end services are designed to support 5,000 users. The back-end services are then doubled, and designed to accommodate 10,000 users. If the system capacity exceeds 5,000 users, the front-end services can be horizontally scaled. If the overall capacity reaches 5,000 users, then the back-end services can be scaled to accommodate. Such design enables flexibility for growth, whether the growth is in terms of users or throughput.

Implementing Local Message Transfer Protocol (LMTP) for Messaging Server

Best practices say you should implement LMTP to replace SMTP for message insertion. An LMTP architecture is more efficient for delivering to the back-end message store because:

- Reduces the load on the stores. Relays are horizontally scalable while stores are not, thus it is a good practice to make the relays perform as much of the processing as possible.

- It reduces IOPS by as much as 30 percent by removing the MTA queues from the stores.
- It reduces the load on LDAP servers.
- The LDAP infrastructure is often the limiting factor in large messaging deployments.
- It reduces the number of message queues.
- It requires a small amount of shared/NFS storage across all inbound MTAs.

You need a two-tiered architecture to implement LMTP.

Background: "How Email Works" Introduction to Messaging Server

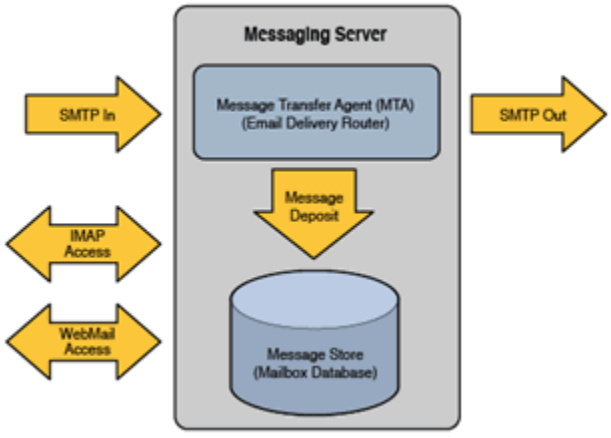
This write up is a basic introduction. It does avoid most of the more complicated configurations and mechanisms which are part of the Messaging Server. The key objective is to understand "how email works" from a basic "black box" model. As this section progresses, you will start getting the more complicated information on the internals of the Messaging Server.

What Does Messaging Server Enable Users to Do?

The Messaging Server enables users to interact with their email message data in three different manners:

1. Users can write an email, and then send it to someone(s).
2. Users can receive emails from others.
3. Users can access their mailbox.

Figure 63–3 How Messaging Server Allows Users to Interact with Message Data



The key components of Messaging Server are:

- Message Transport Agent (MTA)
- Message Store Database
- Message Access Services
- IMAP/POP3

- Webmail Access for Convergence (separate product)

Figure 63–3 simplifies the Messaging Server into the following three components:

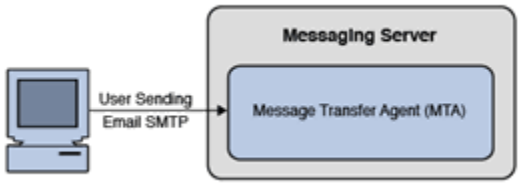
- Messages are delivered into the Messaging Server via the MTA. The MTA is the SMTP Server which accepts the email message and then route the message to it's destination. The destination for the email could be either the message store or to another server somewhere out over the network.
- The message store is an Oracle private database, which stores user's mailboxes and message data.
- Users access their mailboxes via either IMAP, POP3 or through Webmail.

A User Decides to Send an Email

Users write their emails using any of the popular email clients out on the market. These email clients could include Mozilla/Thunderbird, Apple Mail, Outlook, or any other IMAP-based email application. Once the email is ready, the user would click the "SEND" button.

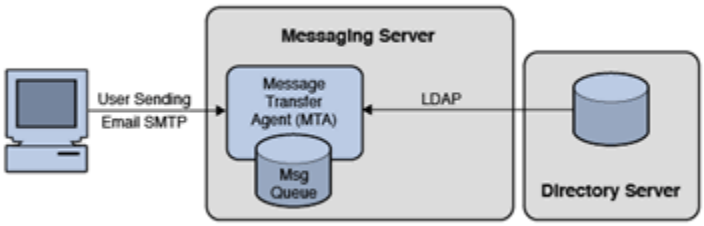
The "SEND" button allows the email application to connect with the MTA. The client connects to the MTA using the SMTP protocol. Figure 63–4 shows the connection of the user to the Messaging Server MTA.

Figure 63–4 Client Connects to the Messaging Server MTA



On the connection to the MTA, the client will send the message over the network to the MTA. The MTA would then store the message in the Message Queue Disk and release the client connection. The message stored on the queue will then be read and the address of the message will be evaluated by using information stored in the Directory Server using LDAP. Figure 63–5 shows the MTA storing the message in the message queue, which is then evaluated by LDAP.

Figure 63–5 MTA Stores Message to Message Queue and Message Evaluated Through LDAP



At this point, the MTA will attempt to deliver the message to one of three places:

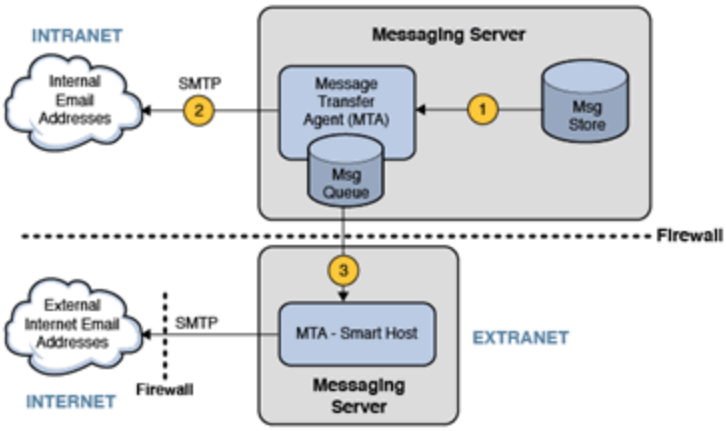
1. Into a User's Mailbox
2. Into another Messaging Server
3. Into another site on the Internet (note for security: it is recommend that the MTA forward the message to a "smarthost" MTA to handle the message delivery over the Internet)

It is important to note that once the message is on the Message Disk Queue, and the MTA has responsibility of the message. We guarantee this responsibility by committing the message to disk. It is this commitment which provides the first clue into a our first performance limitation. We write all MTA data to a disk queue, and we therefore are bounded by the disk i/o performance of that disk system. We also need to ensure that the disk system is highly available through RAID. (Typically RAID 0+1 or RAID 5.)

Postal Service Example: An example of this would be a letter that you drop into a postal mailbox. If you drop a letter of at the Post Office, you are assuming that the Postal Service will mail the message to the destination. You will not expect the message to get lost.

The MTA will not remove the message from the Message Disk Queue until a successful delivery. If a delivery attempt fails, it will keep trying. If it fails a final time, the message will be HELD on the disk queue and a notification sent back to the sender. The MTA logs will record the whole history. Figure 63–6 shows how the MTA delivers a message.

Figure 63–6 How the MTA Delivers a Message



The MTA will attempt to deliver the message in one of three possible ways:

1. Message is Deposited into the User's Mailbox on the message store.
2. Message is Sent to another MTA Server on the Same Local Network.
3. Message is sent to the Internet by first sending to the Smarthost.

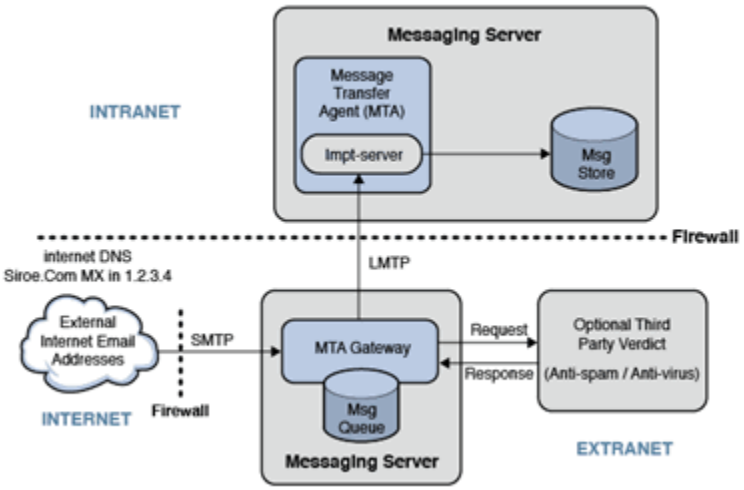
The Smarthost MTA sits between the Messaging Server inside the protected network and the Internet. Typically this would be in an area outside the internal network. The Smarthost would evaluate the message address by domain name. Using DNS, the MTA would attempt to identify the DNS MX Record and A Record for the destination

of the email. It would then attempt to deliver the message to the host of that IP Address using SMTP.

User Receives an Email

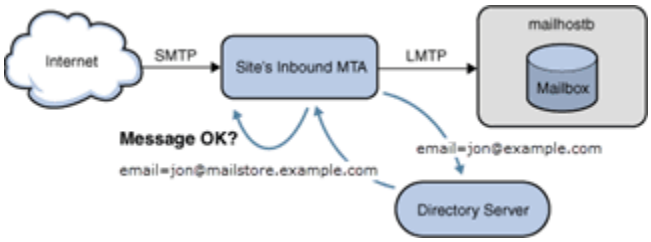
Figure 63–7 shows how a receives email from the MTA.

Figure 63–7 How a User Receives Email from the MTA



An email sent to a user on the message store. In the case of the message arriving from the Internet, the email sender would address the email to "jon@example.com". The email sender's MTA would look up "example.com" in DNS and find either an MX record or an A record for this site. Figure 63–8 shows the delivery process of an inbound message through the MTA.

Figure 63–8 Inbound MTA

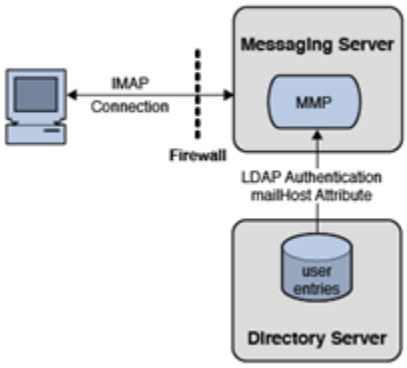


The Inbound MTA for the site may be configured to detect SPAM or Viruses. In this case, the message will be sent to a anti-spam/anti-virus verdict engine (such as Cloudmark or Symantec Brightmail). If the email passes, it is evaluated for message routing. The email address is evaluated in the Directory Server. If the email address is valid, it is then sent to the user's mailbox host (mailboxb).

User Access Mailbox

Figure 63–9 shows the process of users accessing their mailboxes.

Figure 63–9 User Access to Mailbox

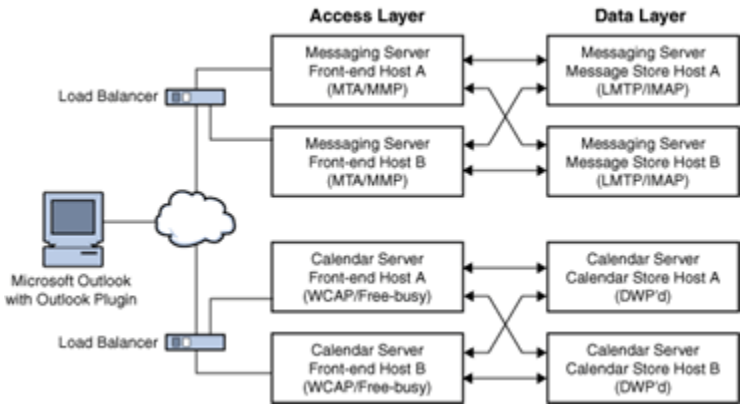


If users want to access their email, they use their email client. These clients, such as Mozilla Thunderbird or Apple Mail, connect to the mailbox using the IMAP protocol. In some cases, ISPs limit access to POP3 protocol.

The email client would connect to Messaging Server's Mail Multiplexor Proxy (MMP). The MMP would accept the user's email credentials (their username and password) and validate their login using the Directory Server. Should the authentication be successful, the MMP would then identify the user's backend message store server. The MMP would then connect to this backend message store on the user's behalf. The user can then access their mailbox data.

The value of the MMP is that it provides 1) Security and 2) Scalability. Security is improved by the use of the MMP by isolating the message store from the end-users. This helps to prevent unauthorized access (hacking) into the message store server data. Scalability (as seen in the figure below) is improved through Horizontal Scalability. This means that the architecture can grow by adding additional MMPs and backend message store servers. Figure 63–10 shows the capacity growth of the access layer through horizontal scalability.

Figure 63–10 Growing Access Layer Capacity



64

Message Store Command Reference

This chapter describes the Oracle Communications Messaging Server message store commands.

configutil

The **configutil** utility enables you to list and change Messaging Server configuration options.

For a list of all configuration options, see *Messaging Server Reference*.

All Messaging Server configuration options and values are stored locally in the **msg.conf** and **msg.conf.defaults** files. The **msg.conf.defaults** file must never be edited and contains defaults constructed during initial configuration. The **msg.conf** file contains settings that have been explicitly overridden from their default value using **configutil**. Use the **-H** option to **configutil** to view a setting's default value.Use **configutil** to edit **configutil** settings; do not edit these files directly.

> **Note:** If the administrator has defined any language-specific options (such as messages), you must use the **language** option at the end of the command in order to list or change them. Commands entered without a **language** option are only applied to attributes that do not have a specified language option.

Requirements: Must be run locally on the Messaging Server. You may run **configutil** as **root** or **mailsrv**. If you make changes to the servers, you must restart or refresh the servers, depending on the variable, for the changes to take effect.

Location: *MessagingServer_home***/bin/configutil**

You can use **configutil** to perform four tasks:

- Display particular configuration options using **-o** *option*.

  Add **;lang-***xx* after the option to list options with a specified language option. For example, **;lang-jp** to list options specified for the Japanese language.

- List configuration option values using the **-p** *pattern* option. (Can be used with the **-m** option.)

  Use **-p** *pattern* to just list those configuration options whose names contain the pattern specified in *pattern*. * is the wildcard character and is assumed to follow *pattern* if there is no wildcard already in *pattern*.

  Use **-m** to show whether or not the listed options are refreshable.

- Set configuration options using the **-o** *option* and **-v** *value* options.

  Add **;lang-***xx* after the option to set options for a specified language option. For example, **;lang-jp** to set options specified for the Japanese language.

- Import configuration option values from **stdin** using the **-i** option.

  Include the **-H** option to show settings with help.

  Examples:

```
configutil -H

Configuration option: alarm.diskavail.msgalarmdescription
        Description: Description for the diskavail alarm.
        Syntax: string
        Default: percentage mail partition diskspace available
alarm.diskavail.msgalarmdescription is currently set to: percentage mail
partition diskspace available

Configuration option: alarm.diskavail.msgalarmstatinterval
        Description: Interval in seconds between disk availability
checks. Set to 0 to disable checks of disk usage.
        Syntax: int
        Default: 3600
alarm.diskavail.msgalarmstatinterval is currently set to: 3600
[.....]

### Show help on all configuration parameters ending in ".port"

configutil -p \*.port -H
Configuration option: local.deploymap.port
        Description: The port Deployment Map option, deploymap.port, specifies
the TCP port on which the Deployment Map service listens for incoming TCP
connections. The default is 4570. Introduced in 8.0.
        Unified Config Name: deploymap.port
        Flags: NULL (unset)
        Syntax: uint16
        Default: 4570

Configuration option: local.ens.port
        Description: Port (and optionally, a specific IP address) ENS
server will listen on, in the format of [address:]port, for example,
7997 or 192.168.1.1:7997.
        Syntax: string
        Default: 7997
local.ens.port is currently set to: 7997

Configuration option: local.service.http.proxy.port
        Description: Configures the port number of the back-end
Messenger Express (HTTP) server with the Messaging Multiplexor.
        Syntax: uint
        Default: 80
local.service.http.proxy.port is currently set to: 80

Configuration option: local.snmp.port
        Description: SNMP subagent port number.
        Syntax: uint
        Default: 0
local.snmp.port is currently set to: 0

Configuration option: local.watcher.port
```

```
        Description: watcher listen port.
        Syntax: uint
        Default: 49994
local.watcher.port is currently set to: 49994

Configuration option: local.webmail.cert.port
        Description: Specifies a port number on the machine where the
Messaging Server runs to use for CRL communication. This port is used
locally for that machine only. The value must be greater than 1024.
        Syntax: int
        Default: 55443
local.webmail.cert.port is currently set to: 55443

Configuration option: local.webmail.da.port
        Description: Delegated Administrator port.
        Syntax: int
        Default: 8080
local.webmail.da.port is currently set to: 8080

Configuration option: local.webmail.sieve.port
        Description: The port of the web container where the Mail Filter
has been deployed.
        Syntax: string
        Default: NULL (unset)
local.webmail.sieve.port is currently unset

Configuration option: metermaid.config.port
        Description: Port number on which MeterMaid listens for
connections.
        Syntax: tcpport
        Default: 63837
metermaid.config.port is currently set to: 63837

Configuration option: pmxbl.port
        Description: RESTRICTED: The port pmxbl option specifies the TCP port
number on which the PureMessage IP Blocker service is running.
Unified Config Name: pmxbl.port
        Flags: USAGE RESTRICTED
        Syntax: uint16
        Default: 4466

Configuration option: service.imap.indexer.port
        Description: The port Indexer option specifies the TCP port on which
ISS listens for incoming TCP connections, i.e., the TCP port to which
Messaging Server should connect to communicate with ISS. The default is 8070.
If the indexer.sslusessl option (service.imap.indexer.sslusessl option in
legacy configuration) is set, the IMAP server uses SSL to authenticate to ISS
on this port.
        Unified Config Name: imap.indexer.port
        Flags: NULL (unset)
        Syntax: uint16
        Default: 8070

Configuration option: service.imap.port
        Description: IMAP server port number.
        Flags: MSG_RESTART_IMAP
        Syntax: uint
        Default: 143
service.imap.port is currently set to: 143
```

```
Configuration option: service.pop.port
        Description: POP server port number.
        Flags: MSG_RESTART_POP
        Syntax: uint
        Default: 110
service.pop.port is currently set to: 110

Configuration option: store.dbreplicate.port
        Description: The port Message Store dbreplicate option specifies the
mailbox list database replication TCP port number. The default is 55000. This
will be used to listen for incoming connections. Introduced in 7.0.5.
        Unified Config Name: store.dbreplicate.port
        Flags: NULL (unset)
        Syntax: uint16
        Default: 55000

### Show help on store.partition.\*.path

configutil -p store.partition.\*.path -H
Configuration option: store.partition.*.path
        Description: Controls the store index file directory path.
        Flags: MSG_RESTART_ALL
        Syntax: filepath
        Default: NULL (unset)
store.partition.primary.path is currently set
to: /opt/sun/comms/messaging64/data/store/partition/primary
store.partition.three.path is currently set
to: /opt/sun/comms/messaging64/data/store/partition/three
store.partition.two.path is currently set
to: /opt/sun/comms/messaging64/data/store/partition/two

Configuration option: store.partition.primary.path
        Description: Full path name of the primary partition.
        Flags: MSG_RESTART_ALL
        Syntax: filepath
        Default: <msg.RootPath>/data/store/partition/primary
store.partition.primary.path is currently set
to: /opt/sun/comms/messaging64/data/store/partition/primary
```

Syntax

```
configutil [-h] [-f configfile] [-o option [-v value]
configutil [-f configfile] [-p pattern] [-H] [-m] [-V]
configutil -i inputfile
```

Options

Table 64–1 describes the options for the **configutil** command.

Table 64–1 configutil Options

Option	Description
-d	Delete an option. Used with the **-o** option.
-f *configfile*	Specify local configuration file to use.
-h	Show usage statement.
-H	Get help on options. Used with the **-o** option.
-i *inputfile*	Import options from an import file.
-m	List meta data.

Table 64–1 (Cont.) configutil Options

Option	Description
-o *option*	Get an option. May be used with the **-H**, **-v**, **-d** options.
-p *pattern*	List only options with the given pattern (* is wildcard). Example: **configutil - p *enable -H**.
-v *value*	Specifies a value for a configuration option. Used with **-o** *option*.
-V	Validate configuration against meta data.

If you specify no command-line options, all configuration options are listed.

Examples

To list all configuration options and their values in both the Directory Server LDAP database and local server configuration file:

```
configutil
```

To import configurations from an input file named **config.cfg**:

```
configutil -i config.cfg
```

To list all configuration options with the prefix **service.imap**:

```
configutil -p service.imap
```

To display the value of the **service.smtp.port** configuration option:

```
configutil -o service.smtp.port
```

To set the value of the **service.smtp.port** configuration option to 25:

```
configutil -o service.smtp.port -v 25
```

To clear the value for the **service.imap.banner** configuration option:

```
configutil -o service.imap.banner -v ""
```

To display the refreshable status of the **service.pop** configuration options:

```
configutil -m -p service.pop
```

This example of the -m option could produce the following sample output:

```
service.pop.allowanonymouslogin = no [REFRESHABLE]
service.pop.banner = "%h %p service (%P %V)" [REFRESHABLE]
service.pop.createtimestamp = 20030315011827Z [REFRESHABLE]
service.pop.creatorsname = "cn=directory manager" [REFRESHABLE]
service.pop.enable = yes [NOT REFRESHABLE]
service.pop.enablesslport = no [NOT REFRESHABLE]
service.pop.idletimeout = 10 [REFRESHABLE]
service.pop.maxsessions = 600 [NOT REFRESHABLE]
service.pop.maxthreads = 250 [NOT REFRESHABLE]
```

Language Specific Options

To list or set options for a specific language, append **;lang-***xx* immediately after the option with no spaces, where *xx* is the two-letter language identifier. For example, to view the text of the Japanese version of the **store.quotaexceededmsg** message:

```
configutil -o "store.quotaexceededmsg;lang-jp"
```

The semicolon is a special character for most UNIX shells and requires special quoting as shown in the example.

Notes on the configutil Utility

- The **configutil** utility only applies to the subset of product options, specifically "configutil" options.
- The **msg.conf** and **msg.conf.defaults** files are only present in legacy configuration mode.
- The **msconfig** command should be used instead of **configutil** when using Messaging Server's Unified Configuration.

counterutil

The **counterutil** utility displays and changes counters in a counter object. It can also be used to monitor a counter object at fixed intervals. See "Gathering Message Store Counter Statistics by Using counterutil" for more information and examples.

Requirements: Must be run locally on the Messaging Server.

Location: *MessagingServer_home*/**bin**/

Syntax

List content of counter registry:

```
counterutil -l
```

Monitor a counter object:

```
counterutil -o counterobject [-i interval] [-n iterations]
```

Reset a counter:

```
counterutil -s -o counterobject -c counter
```

Options

Table 64–2 describes the options for the **counterutil** command.

Table 64–2 counterutil Options

Option	Description
-c *counter*	Specify a counter (for example **global.maxconnections**).
-i *interval*	Monitoring interval. Default is *5* seconds.
-l	List the content of the counter registry.
-n *iterations*	Specify number of iterations..
-o *counterobject*	Specify a *counterobject*, (for example **imapstat**).
-s	Reset counter to 0.

Examples

To list all counter objects in a given server's counter registry:

```
counterutil -l
```

To monitor the content of a counter object **imapstat** every 5 seconds:

```
counterutil -o imapstat
```

To reset the counter **global.maxconnections**, associated with the counter object **imapstat**, to zero:

```
counterutil -s -o imapstat -c global.maxconnections
```

See "Gathering Message Store Counter Statistics by Using counterutil" for usage information on **counterutil**.

deliver

The **deliver** utility delivers mail directly to the message store.

If you are administering an integrated messaging environment, you can use this utility to deliver mail from another MTA, a **sendmail** MTA for example, to the Messaging Server message store.

> **Note:** The **deliver** utility is only for use with files which are already completely and properly formed email messages (RFC 822).

Requirements: Must be run locally on the Messaging Server; the **stored** utility must also be running.

Location on UNIX: *MessagingServer_home*/**bin**/

Syntax

```
deliver [-h] [-r address] [-m folder] [-E encoding] [-g flag]
[-qcpx] userid... | pattern
```

You can specify multiple userids. If you specify no options, mail is delivered to the inbox of the user specified in *userid*.

Options

Table 64–3 describes the options for the **deliver** command.

Table 64–3 deliver Options

Option	Description
-c	Creates the folder if it doesn't exist (INBOX is created automatically).
-E *encoding*	Folder name encoding; for example IMAP-mailbox-name.
-g *flag*	Sets the flag on the message delivered. This option can be specified multiple times.
-h	Displays usage.
-m *folder*	Delivers mail to the specified folder instead of INBOX if folder exists or **-c** is specified.
-p	Expands wildcard (*) in *pattern*.
-q	Disables quota check.
-r *address*	Inserts a **Return-Path:** *address* header.
-x	Enables Indexed Search Converter (ISC) callout in delivery. This option is only valid for Cassandra message store, when ISC is installed.

Tip: The **mboxutil -lxp user/**userid/*mailbox* command can be used to check the access controls on *mailbox* for *userid*.

Debugging and Troubleshooting Options

Table 64–4 describes the troubleshooting options for the **deliver** command.

Table 64–4 deliver Debugging and Troubleshooting Options

Option	Description
-t *seconds*	Deliver as many copies of the message as possible to the *userid* in *seconds* time. This option is primarily meant for load-testing a mailbox.

Examples

To deliver the contents of a file named **message** to Fred's **tasks** mailbox:

```
deliver -m tasks fred < message
```

In the above example, if the **tasks** mailbox does not grant "p" rights to the value of the authid passed in with the -a option or the **tasks** mailbox does not exist, the contents of **message** is delivered to the inbox of the user **fred**.

To deliver the contents of a file named **message** to the inbox of all existing users on the store

```
deliver -p '*' < message
```

To deliver the contents of a file named **message** to the inbox of all existing users in the **example.com** hosted domain.

```
deliver -p '*@example.com' < message
```

Tip: Use the **mboxutil** command to determine which users will receive a copy of the message based on a given user *pattern*, for example **mboxutil -lp "user/***pattern***/INBOX"**

To deliver the contents of a file named **message** to Fred's **Important** mailbox, and set the **'\Flagged'** IMAP flag on that new message.

```
deliver -m Important -g '\Flagged' fred < message
```

hashdir

The **hashdir** command calculates the hash where the specified user ID would be found in a store partition. If the user is not in the default domain, you should append @<domain> to the user ID.

Syntax

```
hashdir [-a] [-i] <userid>[@<domain>]
```

Options

Table 64–5 describes the options for the **hashdir** command.

Table 64–5 hashdir Options

Option	Description
-a	Appends the specified user ID to the output.

Table 64–5 (Cont.) hashdir Options

Option	Description
-i	Allows you to use the command in interactive mode.

Examples

```
hashdir user1
64/b1/
hashdir user1@domain.com
11/05/
```

> **Note:** The **hashdir** command does not validate the input to determine whether it is a valid userid or whether the mailbox exists in the store. Also, the user ID is case sensitive, specifying the wrong case will generate a different hash. To see where the folder exists in the store, the "mboxutil" command may be more useful.

imcheck

The **imcheck** utility prints mailbox data and metadata. In addition, **imcheck** prints **mboxlist** database data for specified database files and prints database statistics.

Syntax

Print mailbox data:

```
imcheck [-m mailbox  [-c|-b msgno] | -x dir [-c|-b msgno] | -p partition | -f
file] [-e | -H] [-D [-S sep]]
```

Print **mboxlist** database data:

```
imcheck -d db_name [-S sep]
```

Print database statistics:

```
imcheck -s [-n] [subsystem...]
```

Print maintenance queue:

```
imcheck -q
```

Options

Table 64–6 describes the options for the **imcheck** command.

Table 64–6 imcheck Options

Option	Description
-b *msgno*	Print message content of the specified message number.
-c *msgno*	Print cache data of the specified message number.
-d *db_name*	Dump the content of the specified database. Print the statistics of the specified database when used with the **-s** option. This option is only valid for Berkeley Database message store.
-D	Dump meta data for **imsbackup** verification.
-e	Print uids of the expunged messages.

Table 64–6 (Cont.) imcheck Options

Option	Description
-f *file*	Print meta data for the mailboxes listed in the specified file.
-F	Return only the values which do not require traversal of the database.
-H	Print mailbox header only.
-m *mailbox*	Print meta data for the specified mailbox. See "imcheck -m Metadata Output" for information about the output produced by this option. For Cassandra message store, **imcheck -m** *mailbox* does not list a partition or path, as Cassandra message store does not use the concept of a partition.
-n	No locking, for debugging db hang problems only.
-p *partition*	Print meta data for every mailbox in the specified partition.
-q	Print maintenance queue. See "Displaying the Maintenance Queue" for more information.
-s	Prints database statistics. This option is only valid for Berkeley Database message store.
-S *sep*	Use *sep* as the separator for **-D** output or a '|' as default. Also used with **-d** option to separate the columns in output with user provided input or a single space as default
-x *dir*	Print meta data for the mailbox under the specified directory.
subsystem	**mpool**, **lock**, **log**, **txn** or **rep**. (If *subsystem* is not specified, all available information is printed.)

Note: When using the separator, check output consistency by trying the separator string as a single or double quoted string with escape sequences if required.

imcheck -m Metadata Output

The **imcheck -m** *mailbox* command displays metadata that describes the entire mailbox and metadata that describes each message.

The message-specific data is displayed in a table. Most of the fields in the table are self-explanatory. However, the following fields need further explanation:

HSize - Header size

MT - Message type ID, defined by **configutil** options such as **store.messagetype.*** and **store.messagetype.enable**. For a list of these options, see *Messaging Server Reference*.

SFlags - System flags. The system flags are as follows:

```
R : Recent
S : Seen
D : Deleted
A : Answered
F : Flagged
T : Draft
B : Stubbed
C : Archived
E : Encrypted
N : NoLeadingDots
```

In the **imcheck -m** *mailbox* output, the system-flag metadata is displayed as a character. For example: **S**

indicates that the Seen flag has been set.

UFlags - Flags defined by the user. User-flags are displayed as a hex number. The binary form of the hex number represents the user-flag switches. For example, if the user has defined six flags, the value

3f0000

indicates that all six flags are set.

Examples

To dump metadata to verify the **imsbackup** operation performed on the user **jdoe**'s INBOX, separating the output with a colon (":"):

```
imcheck -m user/jdoe/INBOX -D -S ":"
```

To dump the contents of the **folder.db** database:

```
imcheck -d folder.db
```

To print metadata for the inbox of the user **jdoe**:

```
imcheck -m user/jdoe/INBOX
-------------------------------------------------------------
Name: user/jdoe
Version: 102
Exists: 10
Flags: 0
Largest Msg: 1094 bytes
Last Append: 20080801062616
Last UID: 1008276527
Oldest Msg: 20000621093214
Oldest Uid: 2
Quota Used: 10061
UID Validity: 1008099930
Cache Offset: 7856
Start Offset: 256
ACL: jdoe      lrswipcda
Userflags:
  f1
  f2
  f3
  f4
  f5
  f6
Subscribed: 0
Partition: primary
Path: /var/opt/sun/comms/messaging64/store/partition/primary/=user/64/b1/=jdoe
Msg Path: /var/opt/sun/comms/messaging64/store/partition/primary/=user/64/b1/=jdoe

MsgNo Uid Internal-Date Sent-Date Size HSize Cache-Id C-Offset C-Len Last-Updated
Save-Date MT SFlags UFlags Original-Uid Message-id
-------------------------------------------------------------------------------
-------------------------------------------------------------------------
1 1 20080801061303 20080801061259 752 744 1 16 1324 20080801061303 20080801061303
0 S 0.0.0 1217596383-1 <200808011312.m71DCxJp017783@dumbo.example.com>
2 2 20080801062616 20080801062615 781 772 1 1340 1440 20080801062616
20080801062616 0 S 0.0.0 1217596383-2
<200808011326.m71DQFYb018990@dumbo.example.com>
```

To print metadata for the Sent folder of a Japanese User **jauser**:

```
bash-3.00# ./mboxutil -E "M-UTF-7" -lp user/jauser/*
  msgs  Kbytes last msg         partition   quotaroot mailbox

     0       0 -          -     primary          5120 user/jauser/INBOX
     0       0 -          -     primary             - user/jauser/&MFQwf3ux-
     0       0 -          -     primary             - user/jauser/&Tgtm+DBN-
     1       0 2009/03/23 11:59 primary             - user/jauser/&kAFP4W4IMH8-
bash-3.00# ./imcheck -m "user/jauser/&kAFP4W4IMH8-"
--------------------------------------------------------------------------------
Name: user/jauser/&kAFP4W4IMH8-
Version: 102
Exists: 1
Flags: 0
Largest Msg: 898 bytes
Last Append: 20090323115905
Last UID: 1
Oldest Msg: 20090323115905
Oldest Uid: 1
Quota Used: 898
UID Validity: 1237768457
Cache Offset: 7856
Start Offset: 256
ACL: jauser     lrswipcda
Subscribed: 1
Partition: primary
Path:
/opt/sun/comms/messaging64/data/store/partition/primary/=user/20/b0/=jauser/=&k+A+
F+P4+W4+I+M+H8-
Msg Path:
/opt/sun/comms/messaging64/data/store/partition/primary/=user/20/b0/=jauser/=&k+A+
F+P4+W4+I+M+H8-

 MsgNo    Uid  Internal Date     Sent Date    Size HSize CacheOff   Last Updated
Save Date MT SFlags UFlags   Original-Uid    Message-id
--------------------------------------------------------------------------------
--------------------------------------------------------------
     1      1 20090323115905 20090323115905      898   549     7856 20090323115905
20090323115905  0 S      0.0.0 1237768457-1
<cd8b67384048.49c77989@dumbo.example.com>
```

imdbverify

The **imdbverify** utility takes and verifies message store database snapshots. See "Administering Message Store Database Snapshots (Backups)" for more information on running **imdbverify**.

Syntax

```
imdbverify [-s] [-m]
```

Options

Table 64–7 describes the options for the **imdbverify** command.

Table 64–7 imdbverify Options

Option	Description
-s	Take a full snapshot of the message store database and save it in the configured destination. Incremental snapshot is performed when this option is not specified.

Table 64–7 (Cont.) imdbverify Options

Option	Description
-m	Verify all database files.

imexpire

imexpire automatically expires messages in the message store based on administrator-specified criteria. The "impurge" command purges the messages.

Topics:

- Expire Actions
- Expire Criteria
- Requirements
- Location
- Syntax
- Options
- Examples
- Additional Functionality

Expire Actions

The expire phase can perform one of these actions:

- Discard - removes messages from the mailbox immediately.
- Archive - archives messages by using the AxsOne archive store.
- Fileinto - moves messages to a specified mailbox folder. If the specified folder does not exist, **imexpire** creates it.
- Report - prints the specified mailbox name, **uid**, and **uid** validity to **stdout**.

By default, **imexpire** discards messages.

> **Note:** Starting with Messaging Server 8.0.2, the action to archive messages by using the AxsOne archive store has been deprecated.

See the following sections for more information about how the **imexpire** actions can be performed:

- Message Store Maintenance Queue
- Configuring Message Expiration (Tasks)
- Message Store Message Expiration

Expire Criteria

The expire criteria can be set with **configutil** options or in a file called **store.expirerule**. Here are some of the criteria that can be specified:

- Folder pattern
- Number of messages in the mailbox
- Total size of the mailbox

- Age, in days, that messages have been in the mailbox. This criterion dates the age of a message from the time it first arrives in the message store (is first received by the user).
- Age, in days, that messages have been saved in a particular mailbox folder. This criterion dates the age of a message from the time it is moved to a particular folder or re-saved in a folder.
- Size of message and grace period (days that a message exceeding the size limit will remain in the message store before removal)
- Whether a message has been flagged as **seen** or **deleted**
- By message header field such as subject or message ID
- According to a sieve script, as defined in RFC 3028

You can use **imexpire** to install a local expire rule file (**store.expirerule**) without conflicting with existing expire rules. If an expire rule file configured for the same partition or mailbox is executing while you try to install a new expire rule file, a warning message appears and the new expire rule file is not installed. Use the **imexpire -i** option to install a local expire rule file.

You can exclude a particular user or mailbox folder from all expire criteria by setting the *exclusive* expire rule for that user or mailbox without specifying any other rules in the expire rule file.

Note: The functionality of **imexpire** has been expanded and the interface has changed since earlier versions of Messaging Server. However, this version continues to support older **imexpire** configurations.

Requirements

imexpire must run locally on the Messaging Server; the **stored** utility must also be running.

Location

The location of **imexpire** is *MessagingServer_home***/bin**.

Syntax

Expire messages from the Message Store:

```
imexpire  [-n] [-d] [-v num] [-p partition | -u user] [-t num][-r num] [-m num]
[-f config_file]
```

Install expire rule file:

```
imexpire -i {-p partition | -x mailbox | -u user} -f config_file
```

Options

Table 64–8 describes the options for the **imexpire** command.

Table 64–8 imexpire Command Options

Option	Description
-d	Display diagnostics to **stdout/stderr**.

Table 64–8 (Cont.) imexpire Command Options

Option	Description
-f *config_file*	Use expire rules in the specified *config_file* (all other expire rules are ignored). When used with the **-i** option, **-f** *config_file* refers to the full path name of the expire rule file.
-i	Install a local expire rule file.
-I	Include trailing spaces on **messageheader.***.
-m *num*	Maximum number of rules in a policy. Default is **128**.
-n	Trial run only: do not perform expire.
-p *partition*	Expire the specified partition. When used with **-i**, it refers to the partition name.
-r *num*	Maximum number of threads per partition. Default is **1**.
-t *num*	Maximum number of threads per process. Default is **50**.
-u *user*	Expire the specified user.
-v *num*	Log expire statistics. When **-d** is specified, messages are displayed to **stdout**. Otherwise, messages are logged to the default log file. *Num* is one of the following: 0: store level statistic (default) 1: user level statistic 2: mailbox level statistic 3: message level statistic
-x *mailbox*	Destination mailbox name, for example: **user/joe/INBOX**.

Examples

Install a local expire rule configuration file for the user **jdoe**. These expire rules will apply to **jdoe's** memos folder.

```
imexpire -i -x user/jdoe/memos -f /home/jdoe/store.expirerule
```

Additional Functionality

Messaging Server provides the following additional functionality to **imexpire**:

- Attributes for Spam and Virus Scanning Through an MTA Channel
- Sieve Body-Test Functionality

Attributes for Spam and Virus Scanning Through an MTA Channel

Table 64–9 describes the attributes for the **imexpire** command that facilitate spam and virus scanning through an MTA channel.

Table 64–9 imexpire Attributes

Attribute	Description
channel	An MTA channel.
rescanhours	Rescan messages that have not been scanned for a specified number of hours.

Sieve Body-Test Functionality

Sieve body-test functionality has been added to the **imexpire** utility. The test can find and perform actions on existing email messages in the message store on keywords in the body part.

To enable a Sieve body-test, set **ENABLE_SIEVE_BODY=1** in *option.dat*.

Example usage in **store.expirerule**:

folderpattern: *

sieve: require "body"; body :contains "bug";

action: discard

iminitquota

The **iminitquota** utility reinitializes the quota limit from the LDAP directory and recalculates the total amount of disk space that is being used by the users. It updates the message store **quota.db** database under the **mboxlist** directory in the message store. The **iminitquota** utility should be run after the **reconstruct -q** utility is run.

Location: *MessagingServer_home*/**bin**/

Syntax

```
iminitquota [-s | -q] -a | -u user
```

Options

Table 64–10 describes the options for the **iminitquota** command.

Table 64–10 iminitquota Options

Option	Description
-a	Initialize quota for all users in the message store.
-q	Initialize quota only.
-s	Initialize quota, usage, and overquota status.
-u *user*	Initialize quota for *user*.

You must specify either the **-a** or **-u** option with the **iminitquota** command.

immonitor-access

Monitors the status of Messaging Server components-Mail Delivery (SMTP server), Message Access and Store (POP and IMAP servers), Directory Service (LDAP server) and HTTP server. This utility measures the response times of the various services and the total round trip time taken to send and retrieve a message. The Directory Service is monitored by looking up a specified user in the directory and measuring the response time. Mail Delivery is monitored by sending a message (SMTP) and the Message Access and Store is monitored by retrieving it. Monitoring the HTTP server is limited to finding out weather or not it is up and running.

The internal operation of **immonitor-access** is as follows: first it does an ldapsearch of a test user created by the administrator. This checks the Directory Server. It can then connect to the SMTP port and send a message to the mail address to check the dispatcher. Then, it checks Message Access by using the IMAP and POP server to see if the message made it to the Message Store. The command logs a message in the default log file if any of the thresholds are exceeded.

The command creates a report that contains the following information:

- The state of the components
- The response time
- The round-trip time for that service

immonitor-access is typically run by **cron** at scheduled intervals to provide a snapshot of the status of the Message Access and Store components. **immonitor-access** can also connect to the IMAP/POP service and delete messages with the subject specified by **-k**. If **-k** is not specified, all messages containing the subject header, **immonitor**, are deleted.

The administrator must create a test user for use by this command before it can be executed.

Syntax

```
immonitor-access [-u user_name] [-w passwd | -j pwdfile] [service...]
[-D threshold] [-k mail_subject] [-m mailfile] [-r alert_recipients]
[-A alternate_mail_server] [-f from_user@domain] [-X] [-T] [-hdnvz]
```

where *service* is one or more of:

```
 Where service is one or more of
        -L LDAP_host[:port]=[threshold][,STLS|PORT] [-b searchbase -B bindDN]
        -S SMTP_host[:port]=[threshold][,STLS]
        -P POP_host[:port]=[threshold][,STLS|PORT]
        -I IMAP_host[:port]=[threshold][,STLS|PORT]
        -H HTTP_host[:port]=[threshold][,STLS|PORT] [-c cookiename]
        -C LMTP_host[:port]=[threshold][,STLS]
        -X (Enable SASL)
        -T (Enable SSL)
```

Options

Table 64–11 describes the task options for **immonitor-access** command.

Table 64–11 immonitor-access Command Options

Option	Description
-A *alternate_mail_server*	The SMTP server to send e-mail alerts to. This option helps in sending alert messages even when the primary mail server is down or heavily loaded. If **-A** is not specified, the SMTP server on the localhost is used.
-b *searchbase*	Use search base as the starting point for the searching in the Directory Server. It is the same as **-b** of **ldap-search(1)**. If *-b* is not specified, the utility uses the value of dcRoot of the configuration option **local.ugldapbasedn**.
-B *bindDN*	LDAP DN to bind as when performing an LDAP search via *-L*. If not specified, then the value of configuration option **local.ugldapbinddn** is used. This option as well as **-j** or **-w** is mandatory when the **-n** option is used.
-C *LMTP_host:* [*port*] = [*threshold*][**,STLS**]	Use the LMTP server and the port specified to check if Messaging Server is able to deliver the message to the store. The threshold is specified in seconds. Use **STLS** for STARTTLS.
-d	The debug mode: display the execution steps.
-D *threshold*	The delivery (also called round-trip time) threshold. The time taken to send the mail and the mail being visible to POP/IMAP. This option can be used only when **-I**/**-P** and **-S**/**-C** are used.

Table 64–11 (Cont.) immonitor-access Command Options

Option	Description
-f *alert_from*	When **immonitor-access** sends out an e-mail, it usually is sent as root@*domainname*. Specify this option to send out an e-mail as different user: **-f user@red.iplanet.com**
-h	Prints command usage syntax.
-H *HTTP_host:* [*port*] = [*threshold*][,**STLS** | **PORT**] [**-c** *cookiename*]	Use the HTTP server and the port specified to check if the HTTP server is able to accept requests on the specified port. When **-I -H** or **-P** is used, it is necessary to provide the test user password with **-w**. When **-S**/**-C**, **-I**/**-P** are specified together, the command does the following: + sends mail and retrieves with IMAP and POP + reports the per protocol response time + reports round-trip time o reports delivery time (the time taken to send the mail and be visible to IMAP/POP) Multiple **-I**, **-P**, and **-S** options can be specified, which helps in monitoring Messaging Server on various systems. Use **STLS** for STARTTLS. Use **PORT** to specify to use SSL port.
-I *IMAP_host:* [*port*] = [*threshold*][,**STLS** | **PORT**]	Use the IMAP server and the port specified to check the IMAP component of the Message Access. The threshold is specified in seconds. The threshold involves the time to login, retrieve, and delete the message. Use **STLS** for STARTTLS. Use **PORT** to specify to use SSL port.
-j *pwdfile*	A readable file containing the password corresponding to the user specified with **-u**. This option, or **-w**, is mandatory when the **-H**, **-I**, or **-P** options are used, or when **-n** is used in conjunction with **-L**. This might be available in a future release.
-k *subject*	Header subject of the messages to be sent/deleted. The utility, by default, uses the string "immonitor:<date>" as the subject in the header sent out with the **-S** option. If **-k** is specified, the string "immonitor:subject" is used in the subject header. This option can be used with **-z** to delete messages, if **-k** is not specified, all messages with the Subject header containing "immonitor" are deleted.
-L *LDAP_host:* [*port*] = [*threshold*][,**STLS** | **PORT**]	Use the LDAP server and the port specified to check the Directory Server. The threshold is specified in seconds. Use **STLS** for STARTTLS. Use **PORT** to specify to use SSL port.
-m *test_file*	The file that is mailed to the test user. You can get response and round-trip times for various mail sizes with this option. Specify only text files as non-text files result in unexpected behavior. If **-m** is not specified, the **mailfile.txt** file in **MessagingServer_ home/lib/locale/C/mailfile.txt** is used as the mail file.
-n	Operate without a Messaging Server configuration. Useful when running the utility on a remote system. This might be available in a future release.
-P *POP_host:* [*port*] = [*threshold*][,**STLS** | **PORT**]	Use the POP server and the port specified to check the POP component of the Message Access. The threshold is specified in seconds. The threshold involves the time to login, retrieve, and delete the message. Use **STLS** for STARTTLS. Use **PORT** to specify to use SSL port.
-r *alert_recipients*	A comma-separated list of mail recipients who will be notified. If this option is not specified, the command reports the alert messages on the standard output.

Table 64–11 (Cont.) immonitor-access Command Options

Option	Description
-S *SMTP_host:* [*port*] = [*threshold*][,**STLS** | **PORT**]	Use the SMTP server and the port specified to check if Messaging Server is able to accept mail for delivery. The threshold is specified in seconds. Use **STLS** for STARTTLS. Use **PORT** to specify to use SSL port.
-T	Enable SSL.
-u *user_name*	The valid test user account to use. This test mail user has to be created by the administrator. If the test mail user is in a hosted domain, *user@domain* should be specified.
-v	Run in verbose mode, with diagnostics written to standard output.
-w *passwd*	The password corresponding to the user specified with *u*. To read the input from standard input, "-" can specified with **-w**. ANY PASSWORD SPECIFIED ON THE COMMAND LINE WITH **-w** WILL BE VISIBLE TO OTHER USERS OF THE SYSTEM. Use of the **-j** option is strongly encouraged. This option, or **-j**, is mandatory when the **-H**, **-I**, or **-P** options are used, or when **-n** is used in conjunction with **-L**.
-X	Enable SASL.
-z	Delete messages containing the string specified by *-k* in the subject header. If *-k* is not specified, all messages with the subject header containing "immonitor" are deleted. Use **-z** only with **-I** or **-P**. Do note use **-z** with **-S** or **-C** as this can cause unexpected results.

The default ports are:

SMTP = **25**

IMAP = **143**

POP = **110**

LDAP = **389**

LMTP = **225**

HTTP = **80**

If either the port or threshold is not specified, default ports with the default threshold of 60 seconds is assumed. The threshold specified can be a decimal number.

Output

The command generates a report containing the various protocol execution times. For example:

```
Smtp Statistics for: thestork:25
Connect Time: 2.122 ms
Greeting Time: 5.729 ms
Helo Time: 2.420 ms
Mail From: Time: 2.779 ms
Rcpt To: Time: 4.128 ms
Data Time: 1.268 ms
Sending File Time: 94.156 ms
Quit Time: 0.886 ms
Total SMTP Time: 113.488 Milliseconds
```

If the alert recipients are specified and any of the threshold values are exceeded, the command mails the report containing the service name and the response time:

ALERT: <service> exceeds threshold Response time=secs/Threshold=secs

Note that in case of times reported for IMAP, the individual times might not add up to the exact value shown by the "Total IMAP time". This occurs because the message does not get to the store immediately. The utility loops until the message is found. Typically, the search time indicates only the successful search time. However, the total time includes each of the individual sleep and search times.

With POP, the utility needs to login and logout multiple times before the message is actually found in the store. Thus, the total time here is the accumulated time for all the logins and log outs.

Example 1: To monitor the SMTP, IMAP, and POP with the threshold 250 milliseconds more than the default value (60 seconds) on localhost use:

```
immonitor-access -S localhost:=60.25 -I localhost:=60.25 -P localhost:=60.25 -u
test_user -w passwd
```

This example assumes that **test_user** exists with password "**passwd**."

Example 2: To monitor LDAP on localhost with no Messaging Server configuration use:

```
immonitor-access -n -L 127.0.0.1:389 -b o=usergroup -u test_user -B "cn=directory
manager" -w passwd
```

The above example assumes that the Directory Manager password is **passwd** and that there is a user named **test_user**.

Example 3: To check remote connectivity and latency issues on **remotehost** use:

```
immonitor-access -S remotehost:=10 -I remotehost:=10 -P remotehost:=10 -u tester
-w passwd
```

Example 4: To run i**mmonitor-access** on a normal port use:

```
immonitor-access -u admin -w password -P host:110
```

Example 5: To run **immonitor-access** using STARTTLS:

```
immonitor-access -u admin -w password -P host:110=STLS -X
```

Example 6: To run **immonitor-access** using SSL via STARTTLS:

```
immonitor-access -u admin -w password -P host:110=STLS -T -X
```

Example 7: To run **immonitor-access** on the SSL port:

```
immonitor-access -u admin -w password -P host:995=PORT -T
```

Example 8: To run **immonitor-access** using SASL on SSL:

```
immonitor-access -u admin -w password -P host:995=PORT -T -X
```

Exit Status

The exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error. A different exit status is returned when various thresholds are exceeded. Table 64–12 describes the exit statuses.

Table 64–12 Exit Status Description

Status	Description
0	Successful execution with no errors or thresholds exceeded
1	Exceeded threshold of a service
2	Errors
64	Usage errors
75	Insufficient virtual memory

An alert message is written to the console when the response time of any server exceeds the threshold.

An error message is written to the console when any of the servers cannot be reached.

Warnings

The password passed with **-w** can be visible to a user using the **ps(1)** command. It is strongly advised that you create a test user to be specifically used by the monitoring utilities.

It is recommended that you use **-w** and enter the password through standard input. However, if the utility is executed through **cron**, the password can be stored in a file. This file can be redirected as the standard input for the utility.

```
cat passwd_file | immonitor-access -w -
immonitor-access -w - ... < passwd_file
```

Do not use the echo command such as:

echo password | immonitor-access .. -w - ..

because the **ps** might show the echo's arguments.

To delete the test mail sent by the **-S** option, invoke the **immonitor-access** command with the **-z** option separately. Do not use the two together.

impurge

The **impurge** command can be used to manually purge unused cache records and message files in mailboxes when the purge server daemon process is not running (**local.purge.enable** is disabled). For more information see *Messaging Server Reference*. The i**mpurge** command has the following options:

Requirements: Must be run locally on the Messaging Server.

Location: *MessagingServer_home*/**lib**/

Syntax

```
impurge [-d] [-r hours] [-e]
```

Options

Table 64–13 describes the options for the **impurge** command.

Table 64–13 impurge Options

Option	Description
-d	Debug mode

Table 64–13 (Cont.) impurge Options

Option	Description
-e	Exit when the work queue is empty
-r *hours*ed	Exit after the specified number of hours (hours must be larger than 0)

If both **-r** and **-e** are specified, **impurge** exits when the queue is empty or time has expired.

Only one **impurge** process can be executed at a time. Attempting to run the **impurge** command whilst the purge server daemon is running will result in the following error message in the Messaging Server debug logs:

```
[24/Feb/2009:14:43:59 +1100] hostname impurge[17471]: General Error: Could not get
purge session lock.
Possibly another impurge is running
```

imquotacheck

The **imquotacheck** utility administers user quotas and domain quotas in the message store. This utility can also compare mailbox size with a user's assigned quota. As an option, you can email a notification to users who have exceeded a set percentage of their assigned quota.

The **imquotacheck** utility lists users in the local mboxlist database. To list users in the LDAP directory, use the **imquotacheck -a** option.

Requirements: Must be run locally on the Messaging Server.

Location:*MessagingServer_home***/bin/**

Syntax

Report quota and usage:

```
imquotacheck [-u user | -d domain [-p partition] | -p partition ] [-a]
```

Enforce domain quota:

```
imquotacheck -f [-d domain]
```

Deliver over quota notification messages:

```
imquotacheck -n [-e] [-d domain] [-r rulefile] [-t msgfile] [-l]
```

Options

Table 64–14 describes the options for the **imquotacheck** command.

Table 64–14 imquotacheck Options

Option	Description
-a	List users in LDAP instead of message store.
-d domain	Report quota for the specified domain. When used with the **-f** option, enforce quota for the specified domain. When used with the **-n** option, deliver notification for the specified domain.
-f	Enforce domain quota by setting **mailDomainStatus** to **overquota**.
-h	Displays help for this command.
-n	Notify users who are above quota.

Table 64–14 (Cont.) imquotacheck Options

Option	Description
-p *partition*	Report quota for the specified partition.
-r *rulefile*	Use *rulefile* instead of default. To set up a default *rulefile*, copy the "Sample Rulefile" to *MessagingServer_home*/**config/imq.rulefile**. See "Rulefile Format" for more information.
-t *msgfile*	Use *msgfile* as a template instead of default. To setup a default message file, copy the "Notification File" to *MessagingServer_home*/**config**.
-u *user*	Report quota for the specified user.
-e	Report disk usage of the individual folders in notification messages.
-l	Log users to action log who are above quota.

Examples

To send a notification to all users in accordance to the default **rulefile**:

```
imquotacheck -n
```

To send a notification to all users in accordance to a specified **rulefile**, **myrulefile**, and to a specified mail template file, **mytemplate.file**:

```
imquotacheck -n -r myrulefile -t mytemplate.file
```

To enforce the domain quota for the **example.com** domain:

```
imquotacheck -f -d example.com
```

To list the usage of all users whose quota exceeds the least threshold in the **rulefile**:

```
imquotacheck
```

To list per folder usages for user **user1**:

```
imquotacheck -u user1
```

To only list the users in domain **example.com:**

```
imquotacheck -d example.com
```

The **imquotacheck** command has stable output formats that we support and allow customers to parse.

Sample imquotacheck output:

```
Name                 Quota(K) Usage(K)  %   Quota#  Usage#  %  OverDate WarnDate
-------------------- -------- -------- --- ------- ------- --- -------- --------
joe                         -        6   -       -       1   -        -        -
mary                      100      110 110       -      49   - 10/14/15 10/14/15
big                  52428800      420   0       -    1673   -        -        -
small                       -    1014B   -       -       3   -        -        -
--------------------------------------------------------------------------------
```

Rulefile Format

The **rulefile** format is organized into two sections: a general section and a rule name section. The general section contains attributes that are common across all rules. Attributes that are typically specified in the general section are the **mailQuotaAttribute** and the **reportMethod**. In the rule name section, you can write specific quota rules for notification intervals, trigger percentages, and so on. Attributes that are typically specified in the rule name section are **notificationTriggerPercentage**,

enabled, **notificationInterval**, and **messageFile**. Note that the attributes and corresponding values are not case-sensitive. The following rulefile format is used:

```
[General]
mailQuotaAttribute = [value]
reportMethod = [value]

[rulename1]
attrname=[value]
attrname=[value]

[rulename2]
attrname=[value]
attrname=[value]

[rulename3]
attrname=[value]
attrname=[value]
```

Table 64–15 describes the **rulefile** attributes.

Table 64–15 rulefile Attributes

Attribute	Required Attribute?	Default Value	Description
mailQuotaAttribute	No	Value in **quotadb**	Specifies the name of the custom mailquota attribute. If not specified, the value in **quotadb** is used.
reportMethod	No	Not Applicable	You can provide your own code to customize the output of the quota report. The value of this attribute is specified as <library-path:function>, where <library-path> is the path of the shared library and <function> is the name of the report function. See "reportMethod Signature" for more information about what options your function must accept and return.
notificationTrigger Percentage	Yes	Not Applicable	Specifies the consumed quota percentage that will trigger notification. Value should be unique and an integer.
messageFile	No	<*MessagingServer_ home*>**/config/imq.msgfile**	Specifies the absolute path to the message file.
notificationInterval	Yes	Not Applicable	Indicates the number of hours before a new notification is generated.
enabled	No	**0** (FALSE)	Indicates if the particular rule is active. Applicable values are **0** for FALSE and **1** for TRUE.
notificationMethod	No	Not Applicable	You can provide your own code to perform the overquota notification. The value of this attribute is specified as <library-path:function>, where <library-path> is the path of the shared library and <function> is the name of the report function. See "notificationMethod Signature" for more information about what options your function must accept and return.

reportMethod Signature

If you override the imquotacheck **reportMethod()** with your own function, it must be defined as:

```
int symbol(QuotaInfo* info, char** message, int* freeflag)

typedef struct QuotaInfo {
  const char* username; /* user name (uid or uid@domain) */
  long quotakb; /* quota in kbytes */
  long quotamsg;  /* quota in number of messages */
  ulong usagekb; /* total usage in kbytes */
  ulong usagemsg; /* total usage in number of messages */
  FolderUsage* folderlist; /* folder list (for -e) */
   long num_folder;  /* number of folders in the folderlist */
  long trigger; /* not used */
   const char* rule; /* not used */
}

typedef struct FolderUsage {
   const char*foldername;
   ulong usagekb; /* folder usage in kbytes */
}
```

The address, **message**, points to the output message. The report function is expected to fill the value of ***message** and allocate memory for **message** when necessary. The **freeflag** variable indicates if the caller is responsible for freeing allocated memory for ***message**.

The return values are 0 for success and 1 for failure.

The **imquotacheck** function will invoke the **reportMethod** to generate the report output. If the **reportMethod** returns 0 and ***message** is pointing to a valid memory address, **message** will be printed.

If the ***freeflag** is set to 1, the caller will free the memory address pointed to by **message**. If the **-e** option is specified, the quota usage for every folder will be stored in the **folderlist**, an array in **FolderUsage**; the **num_folder** variable is set to the number of folders in the **folderlist**.

notificationMethod Signature

If you override the imquotacheck **notificationMethod()** with your own function, it must be defined as:

```
int symbol(QuotaInfo* info, char** message, int* freeflag)

typedef struct QuotaInfo {
   const char* username; /* user name (uid or uid@domain) */
   long quotakb; /* quota in kbytes */
   long quotamsg;  /* quota in number of messages */
   ulong usagekb; /* total usage in kbytes */
   ulong usagemsg; /* total usage in number of messages */
   FolderUsage* folderlist; /* folder list (for -e) */
   long num_folder;  /* number of folders in the folderlist */
   long trigger; /* the exceeded notificationTriggerPercentage */
   const char* rule; /* rulename that triggered notification */
}

typedef struct FolderUsage {
  const char *foldername;
   ulong usagekb; /* folder usage in kbytes */
}
```

The address, **message**, points to the notification message. The notification function is expected to fill in the value of this variable and allocate the memory for the message

when necessary. The **freeflag** variable indicates if the caller is responsible for freeing the memory allocated for **message**.

The return values are 0 for success and 1 for failure.

If the notification function returns a 0, and ***message** is pointing to a valid address, the **imquotacheck** utility will deliver the message to the user. If the ***freeflag** is set to 1, the caller will free the memory address pointed to by message after the message is sent.

If the **-e** option is specified, the quota usage for every folder will be stored in the **folderlist** variable, an array of **FolderUsage** structure; the **num_folder** variable is set to the number of folders in the **folderlist**.

Note: If the **messageFile** attribute is also specified, the attributed **messageFile** will be ignored.

Sample Rulefile

In the sample rulefile, the following files **libzz.so**, **libzz.sl**, and **libzz.dll** are library files. The */xx/yy* are directory paths that should be replaced by the relative paths to where these library files are located on the server.

```
#
# Sample rulefile
#
[General]
mailQuotaAttribute=mailquota
reportMethod=/xx/yy/libzz.so:myReportMethod [for Solaris only ]

[rule1]
notificationTriggerPercentage=60
enabled=1
notificationInterval=3
notificationMethod=/xx/yy/libzz.so:myNotifyMethod_60

[rule2]
notificationTriggerPercentage=80
enabled=1
notificationInterval=2
messageFile=/xx/yy/message.txt

[rule3]
notificationTriggerPercentage=90
enabled=1
notificationInterval=1
notificationMethod=/xx/yy/libzz.so:myNotifyMethod_90
#
# End
#
```

Threshold Notification Algorithm

1. Rule precedence is determined by increasing trigger percentages.
2. The highest applicable threshold is used to generate a notification. The time and the rule's threshold are recorded.
3. If users move into a higher threshold since their last quota notification, a new notification will be delivered based on the current set of applicable rules. This

notice can be immediately delivered to any user whose space usage is steadily increasing.

4. If usage drops, the notification interval of the current rule (lower threshold) will be used to check the time elapsed since the last notice.
5. The stored time and threshold for the user will be reset to zero if the user's mailbox size falls below all of the defined thresholds.

Notification File

The utility depends on the message file to have at minimum a Subject Header. There should be at least one blank line separating the Subject from the body. The other required headers are generated by the utility. The notification file format is the following:

```
Subject: [Warning] quota reached for %U%


Hello %U%,
Your quota: %C%
Your current mailbox usage: %M%
Your mailbox is now %Q% full. The folders consuming the most space are:
%R%.

Please clean up unwanted diskspace.

Thanks,
-Administrator
```

where:

%U% - Species the user ID.

%Q% - Specifies the percentage of the mailbox quota used.

%R% - Specifies quota usage details, including assigned quota, total mailbox size, and percentage used. When -e is specified on the command line, the report shows the mailbox usage of the individual folders.

%M% - Specifies the current mailbox size.

%C% - Specifies quota attribute value.

Note: If an account has less then 1KB of quota usage, **imquotacheck** prints out the usage in bytes (B) rather than kilobytes (KB) shown in the heading.

```
imquotacheck -u testquota
Name    Quota(K) Usage(K)  %   Quota#  Usage#  %  OverDate WarnDate
------------------- -------- -------- --- ------- ------- --- ----
testquota  256000   654B   0   100000    1     0      -        -
-----------------------------------------------------------------
```

Note: Localized versions of **imquotacheck** notification incorrectly convert the % and the $ signs. To correct the encoding, replace every $ with **\24** and replace every % with **\25** in the message file.

---

imsasm

The **imsasm** utility is an external ASM (Application Specific Module) that handles the saving and recovering of user mailboxes. **imsasm** invokes the **imsbackup** and **imsrestore** utilities to create and interpret a data stream.

During a save operation **imsasm** creates a save record for each mailbox or folder in its argument list. The data associated with each file or directory is generated by running the **imsbackup** or **imsrestore** command on the user's mailbox.

Location: *MessagingServer_home*/**lib/msg**

Syntax

```
imsasm -s [-benov] [-ix] [ -t time ] [ -f proto ] [ -p ppath ] file...
imsasm -r [-dnv] [ -i {nNyYrR} ] [ -m src=dst ] -z suffix [ -p path ] file...
imsasm -c [-nv] [ -p path ] file...
```

Options

The options used in the **imsasm** utility are also known as standard-asm-arguments, which are Legato NetWorker backup standards.

Either **-s** (saving), **-r** (recovering), or **-c** (comparing) must be specified and must precede any other options. When saving, at least one *path* argument must be specified. *path* may be either a directory or filename.

Table 64–16 describes the **imsam** options that are valid for all modes.

Table 64–16 imsasm Options

Option	Description
-n	Performs a dry run. When saving, walk the file system but don't attempt to open files and produce the save stream. When recovering or comparing, consume the input save stream and do basic sanity checks, but do not actually create any directories or files when recovering or do the work of comparing the actual file data.
-v	Turns on verbose mode. The current ASM, its arguments, and the file it is processing are displayed. When a filtering ASM operating in filtering mode (that is, processing another ASM's save stream) modifies the stream, its name, arguments, and the current file are displayed within square brackets.

Table 64–17 describes the options you can use when saving (**-s**).

Table 64–17 imsasm Saving Options

Option	Description
-b	Produces a byte count. This option is like the **-n** option, but byte count mode will estimate the amount of data that would be produced instead of actually reading file data so it is faster but less accurate than the **-n** option. Byte count mode produces three numbers: the number of records, i.e., files and directories; the number of bytes of header information; and the approximate number of bytes of file data. Byte count mode does not produce a save stream so its output cannot be used as input to another asm in recover mode.
-o	Produces an "old style" save stream that can be handled by older NetWorker servers.
-e	Do not generate the final "end of save stream" Boolean. This flag should only be used when an ASM invokes an external ASM and as an optimization chooses not to consume the generated save stream itself.
-i	Ignores all save directives from **.nsr** directive files found in the directory tree.

Table 64–17 (Cont.) imsasm Saving Options

Option	Description
-f *proto*	Specifies the location of a **.nsr** directive file to interpret before processing any files. Within the directive file specified by *proto*, *path* directives must resolve to files within the directory tree being processed, otherwise their subsequent directives will be ignored.
-p *ppath*	Prepends this string to each file's name as it is output. This argument is used internally when one ASM executes another external ASM. *ppath* must be a properly formatted path which is either the current working directory or a trailing component of the current working directory.
-t *date*	The date after which files must have been modified before they will be saved.
-x	Crosses file system boundaries. Normally, file system boundaries are not crossed during walking.

Table 64–18 describes the options you can use when recovering (**-r**).

Table 64–18 imsasm Recovering Options

Option	Description
-i *response*	Specifies the initial default overwrite response. Only one letter may be used. When the name of the file being recovered conflicts with an existing file, the user is prompted for overwrite permission. The default response, selected by pressing **Return**, is displayed within square brackets. Unless otherwise specified with the **-i** option, **n** is the initial default overwrite response. Each time a response other than the default is selected, the new response becomes the default. When either **N**, **R**, or **Y** is specified, no prompting is done (except when auto-renaming files that already end with the rename suffix) and each subsequent conflict is resolved as if the corresponding lowercase letter had been selected. The valid overwrite responses and their meanings are: ■ **n**-Do not recover the current file. ■ **N**-Do not recover any files with conflicting names. ■ **y**-Overwrite the existing file with the recovered file. ■ **Y**-Overwrite files with conflicting names. ■ **r**-Rename the conflicting file. A dot "." and a suffix are appended to the recovered file's name. If a conflict still exists, the user will be prompted again. ■ **R**-Automatically renames conflicting files by appending a dot "." and a suffix. If a conflicting file name already ends in a *.suffix*, the user will be prompted to avoid potential auto rename looping conditions.
-m *src=dst*	Maps the file names that will be created. Any files that start exactly with src will be mapped to have the path of dst replacing the leading src component of the path name. This option is useful if you want to perform relocation of the recovered files that were saved using absolute path names into an alternate directory.
-z *suffix*	Specifies the suffix to append when renaming conflicting files. The default suffix is **R**.
path	Restricts the files being recovered. Only files with prefixes matching path will be recovered. This checking is performed before any potential name mapping is done with the **-m** option. When path is not specified, no checking is performed.

Examples

To use **imsasm** to save the mailbox **INBOX** for user **joe**, the system administrator creates a directory file *backup_root***/backup/DEFAULT/joe/.nsr** with the following contents:

```
imsasm: INBOX
```

This causes the mailbox to be saved using **imsasm**. Executing the **mkbackupdir** utility will automatically create the **.nsr** file. See "mkbackupdir" for more information.

imsbackup

The **imsbackup** utility is used to write selected contents of the message store to any serial device, including magnetic tape, a UNIX pipe, or a plain file. The backup or selected parts of the backup may later be recovered via the **imsrestore** utility. The **imsbackup** utility provides a basic backup facility similar to the UNIX **tar** command.

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

Location: *MessagingServer_home***/bin**

For more information about **imsbackup** and backing up the message store, see "Backing Up and Restoring the Message Store".

Syntax

```
imsbackup -f file [-b blockfactor] [-d date] [-m link_count]
[-e encoding] [-u file] [-n path] [-ivlgrpx] name...
```

Options

Table 64–19 describes the options for the **imsbackup** command.

Table 64–19 imsbackup Options

Option	Description
-b *num*	Use *num* as the block factor.
-d *date*	Backup data newer than *date*, in the form YYYYMMDD[:HHMMSS].
-e *encoding*	Mailbox name encoding (for example, IMAP-MODIFIED-UTF-7).
-f *file*	Output file.
-g	Debug mode.
-i	Ignore links (for partial restore).
-l	Autoload.
-m *link_count*	Minimum link count for hashing.
-n *path*	Networker backup path.
-p	Print progress
-r	Read only mode (do not upgrade or repair mailbox automatically).
-u *file*	use object names in *file.*
-v	Verbose.
-x	Back up expunged messages.
name	Object names (can be a logical path, internal name or userid). If **-n** is specified, *name* is either a userid or mailbox name.

Examples

The following example backs up the entire message store to **/dev/rmt/0**:

```
imsbackup -f /dev/rmt/0 /
```

The following backs up the mailboxes of user ID **joe** to **/dev/rmt/0**:

```
imsbackup -f /dev/rmt/0 /primary/user/joe
```

The following example backs up all the mailboxes of all the users defined in the backup group **groupA** to **backupfile:**

```
imsbackup -f- /primary/groupA > backupfile
```

imsconnutil

Monitors user access of the message store. **imsconnutil** can provide the following information:

- Users currently logged in on IMAP or any HTTP webmail client.
- The last access time (log in or log out) for a specified user.
- For IMAP: lists the authentication method, the IP address from which the users are logged in, the IP address to which users are connected, and the port on which they are logged to and from.

This command requires root access by the system user, and you must set the configuration variables **imap.enableuserlist** and **http.enableuserlist** to 1.

See "Monitoring User Access to the Message Store" for more information and examples.

Location: *MessagingServer_home***/bin**

Syntax

```
imsconnutil -c | -a | -k [-s service] [-u uid] [-f filename]
```

Options

Table 64–20 describes the options for the **imsconnutil** command.

Table 64–20 imsconnutil Options

Option	Description
-c | **-a** | **-k**	At least one of **-c** or **-a**, or **-k** must be used.
-a	Gives information on the last access time of a service.
-c	Gives information on users connected to the service.
-f *filename*	Narrows the information to users specified in file. The format is one user per line in this file.
-k	Disconnect users (use with **-u** or **-f**).
-s *service*	Narrows the service to either IMAP or HTTP (use with **-c**).
-u *uid*	Narrows the information to one user uid.

Examples

- List every user ID currently logged into IMAP and HTTP.

imsconnutil -c

- List last IMAP, POP, or HTTP access (log in or log out) of every user ID.

imsconnutil -a

- List access history (last log off or log on) of all user IDs. Lists current user IDs logged into IMAP and HTTP.

imsconnutil -a -c

- List IMAP users currently logged on the message store.

imsconnutil -c -s imap

- Reveal whether user ID George is logged onto IMAP or not.

imsconnutil -c -s imap -u George

- Reveal whether user ID George is currently logged onto IMAP or HTTP, and lists the last time George was logged o or off.

imsconnutil -c -a -u George

- Disconnect the user ID George.

imsconnutil -k -u George

Notes

The "rehostuser" utility makes use of the **imsconnutil** command.

imscripter

The **imscripter** utility connects to an IMAP server and executes a command or a sequence of commands.

May be run remotely.

Location: *MessagingServer_home*/**bin**/

Syntax

Print help:

```
imscripter [-h]
```

Go interactive:

```
imscripter [options]
```

Execute command from *cmdfile*:

```
imscripter [options] -f cmdfile
```

Execute the given command with the given argument(s):

```
imscripter [options] -c command [cmddata1 ...]
```

Execute the given command for each argument line in the *datafile*:

```
imscripter [options] -c command -f datafile
```

Options

Table 64–21 describes the options for the **imscripter** command.

Table 64–21 imscripter Command Options

Option	Description
-c *command*	Executes *command* - see discussion of commands below.
-f *cmdfile*	The *cmdfile* may contain one or more commands, or a list of mailboxes on which commands are to be executed. See discussion of commands below.
-h	Displays help for this command.
-p *port*	Connect to the given port. The default is **143**.
-s *server*	Connect to the given server. The default is **localhost.**.
-u *userid*	Connects as *userid* (prompt if not specified).
-v *verbosity*	Verbosity string can contain the following characters: **E**: Show error **I**: Show informational messages **P**: Show prompt **C**: Echo input command **c**: Echo protocol command **B**: BAD or NO untagged response **O**: Other untagged response **b**: BAD or NO completion result **o**: OK completion result **A**: All of the above Default is **EPBboO**(**EBb** if **-f** specified)
-x *passwd*	Uses this *passwd* (prompt if not specified).

Recognized Commands

The imscripter command supports the following commands:

```
create mailbox
delete mailbox
rename oldmailbox newmailbox [partition]
getacl mailbox
setacl mailbox userid rights
deleteacl mailbox userid
```

If one or more of the above variables are included, the option executes the given command with that input. For example, **create lincoln** creates a mailbox for the user **lincoln**.

If the **-f** *file* option is used, the option executes the command on each variable listed in the file. For example:

```
cat folders-to-create
xyz
abc
def
imscripter -u <userid> -x <password> -c create -f folders-to-create
#
```

Raw IMAP Commands

In addition to the commands which imscripter interprets (above), it will pass thru any raw IMAP command prefixed with an exclamation mark ("!"). For example:

```
imscripter -u <userid> -x <password> -c \!list "" \*
   1) <= OK [CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS
CHILDREN BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT
THREAD=ORDEREDSUBJECT THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN
SASL-IR SEARCHRES XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1
X-UNAUTHENTICATE X-SUN-IMAP X-ANNOTATEMORE XUM1 ID IDLE AUTH=PLAIN]
s4u-280rd-zone1-bur02.east.sun.com IMAP4 service (Oracle Communications Messaging
Exchange Server 7u4-20.01 64bit (built Nov 21 2010))
   1) <= OK User logged in
   2) <= CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS CHILDREN
BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT THREAD=ORDEREDSUBJECT
THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN SASL-IR SEARCHRES
XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1 X-UNAUTHENTICATE
X-SUN-IMAP X-ANNOTATEMORE XUM1 ID IDLE
   2) <= OK Completed
   2) <= LIST (\NoInferiors) "/" INBOX
   2) <= LIST (\HasNoChildren) "/" Drafts
   2) <= LIST (\HasNoChildren) "/" Sent
   2) <= LIST (\HasNoChildren) "/" Trash
   2) <= LIST (\HasNoChildren) "/" abc
   2) <= LIST (\HasNoChildren) "/" def
   2) <= LIST (\HasNoChildren) "/" test2
   2) <= LIST (\HasNoChildren) "/" xyz
   2) <= OK Completed
   3) <= BYE LOGOUT received
   3) <= OK Completed
#
```

Interactive Mode

If you issue the imscripter command without the -f and/or -c switches, it will go into interactive mode. If you did not specify -u, it will prompt for userid. If you did not specify -x, it will prompt for password. You can then type imscripter commands or raw IMAP commands prefixed by "!" and see the response. See discussion of commands above.

For example:

```
imscripter
s4u-280rd-zone1-bur02 userid: <userid>
s4u-280rd-zone1-bur02 password for <userid>:
   1) <= OK [CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS
CHILDREN BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT
THREAD=ORDEREDSUBJECT THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN
SASL-IR SEARCHRES XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1
X-UNAUTHENTICATE X-SUN-IMAP X-ANNOTATEMORE XUM1 ID IDLE AUTH=PLAIN]
s4u-280rd-zone1-bur02.east.sun.com IMAP4 service (Oracle Communications Messaging
Exchange Server 7u4-20.01 64bit (built Nov 21 2010))
   1) <= OK User logged in
   2) <= CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS CHILDREN
BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT THREAD=ORDEREDSUBJECT
THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN SASL-IR SEARCHRES
XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1 X-UNAUTHENTICATE
X-SUN-IMAP X-ANNOTATEMORE XUM1 ID IDLE
   2) <= OK Completed
imscripter> !select inbox
   3) <= FLAGS (\Answered \Flagged \Draft \Deleted \Seen $Forwarded)
   3) <= OK [PERMANENTFLAGS (\Answered \Flagged \Draft \Deleted \Seen $Forwarded
\*)]
   3) <= EXISTS
   3) <= RECENT
```

```
   3) <= OK [UNSEEN 14]
   3) <= OK [UIDVALIDITY 1266949413]
   3) <= OK [UIDNEXT 354]
   3) <= OK [READ-WRITE] Completed
imscripter> quit
   5) <= BYE LOGOUT received
   5) <= OK Completed
#
```

Individual Command Mode

As with interactive mode (above), -s and -p default to localhost and port 143 and imscripter will prompt for -u and -x if not specified.

With -c, you can execute an individual imscripter command and optionally have it operate on several folders. See the example of creating several folders in the discussion of commands above.

Script Mode

As with interactive mode (above), -s and -p default to localhost and port 143 and imscripter will prompt for -u and -x if not specified.

With -f, you can put imscripter and raw IMAP commands in a file and have imscripter execute that script. For example:

```
cat script
create nmo
create blah
!list "" *
imscripter -u <userid> -x <password> -f script
#
```

Notice the above displayed no output. It executed the command but did not display the output. Use the -v switch it specify which output you want to see. For example:

```
imscripter -u <userid> -x <password> -f script -v A
Processing: script, verbosity: A, server: (<ask user>:143)
connecting to s4u-280rd-zone1-bur02:143...
addcallback CAPABILITY, NO, OK BAD
   1) => LOGIN <userid> *    *notice imscripter hid the password*
   1) <= OK [CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS
CHILDREN BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT
THREAD=ORDEREDSUBJECT THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN
SASL-IR SEARCHRES XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1
X-UNAUTHENTICATE X-SUN-IMAP X-ANNOTATEMORE XUM1 ID IDLE AUTH=PLAIN]
s4u-280rd-zone1-bur02.east.sun.com IMAP4 service (Oracle Communications Messaging
Exchange Server 7u4-20.01 64bit (built Nov 21 2010))
   1) <= OK User logged in
   2) => CAPABILITY
   2) <= CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS CHILDREN
BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT THREAD=ORDEREDSUBJECT
THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN SASL-IR SEARCHRES
XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1 X-UNAUTHENTICATE
X-SUN-IMAP X-ANNOTATEMORE XUM1 ID IDLE
   2) <= OK Completed
   3) Cmd: create nmo
   3) => CREATE nmo
   3) <= NO Mailbox already exists        *this command failed because it
succeeded the first time*
   4) Cmd: create blah
   4) => CREATE blah
```

```
   4) <= NO Mailbox already exists        *ditto*
   5) Cmd: !list "" *
   5) => list "" *
   5) <= LIST (\NoInferiors) "/" INBOX
   5) <= LIST (\HasNoChildren) "/" Drafts
   5) <= LIST (\HasNoChildren) "/" Sent
   5) <= LIST (\HasNoChildren) "/" Trash
   5) <= LIST (\HasNoChildren) "/" abc
   5) <= LIST (\HasNoChildren) "/" blah
   5) <= LIST (\HasNoChildren) "/" def
   5) <= LIST (\HasNoChildren) "/" nmo
   5) <= LIST (\HasNoChildren) "/" test2
   5) <= LIST (\HasNoChildren) "/" xyz
   5) <= OK Completed
   7) => LOGOUT
   7) <= BYE LOGOUT received
   7) <= OK Completed
#
```

imsexport

The **imsexport** utility exports Messaging Server folders into UNIX **/var/mail** format folders.

The **imsexport** utility extracts the messages in a message store folder or mailbox and writes the messages to a UNIX file under the directory specified by the administrator. The file name is the same name as the IMAP folder name. For message store folders that contain both messages and sub-folders, **imsexport** creates a directory with the folder name. For classic message store, it also creates a file with the folder name plus a **.msg** extension. The *folder***.msg** file contains the messages in the folder. The *folder* directory contains the sub-folders.

Table 64–22 shows the UW-IMAP MBOX header fields utilized by the **imsexport** utility.

Table 64–22 UW-IMAP MBOX Header Fields

Header	Syntax	Description
X-IMAPbase	UIDVALIDITY LASTUID KEYWORD...	Specifies IMAP uidvalidity, last uid, and a list of used keywords.
X-UID	uid	Specifies IMAP uid.
X-KEYWORDS	user flags	Specifies a list of space-separated user flags.
STATUS	[R][O]	Specifies status as R:Read (\Seen), O:not recent (\Recent)
X-STATUS	[A][F][T][D]	Specifies A:\Answered F:\Flagged T:\Draft D:\Deleted flags.

Location: *MessagingServer_home*/**bin**

Syntax

```
imsexport [-g] [-v 0|1|2] [-f format] [-s mailbox] [-e encoding] -u user -d dir
```

Options

Table 64–23 describes the options for the **imsexport** command.

Table 64–23 imsexport Options

Option	Description
-d *dir*	Destination directory (UNIX path name).
-e *encoding*	Mailbox name encoding (for example, **IMAP-mailbox-name**).
-f *format*	Export format, where *format* is BSD, SYSV or RN. Default is Solaris.
-g	Debug mode.
-s *folder*	Source folder name.
-u *user*	User name.
-v *mode*	Verbose mode, 0-2. Default is 2.

Example

In the following example, **imsexport** extracts all email for user **smith1**. **smith1** is a valid user account in the message store. User **smith1** has three folders on the store: **INBOX** (the normal default user folder), **private**, and **private/mom**. The destination directory will be **/tmp/joes_mail**.

```
imsexport -u smith1 -d /tmp/joes_mail/
```

imsexport then transfers each message store folder into a **/var/mail** conforming file. Thus you will get the following files:

- **/tmp/joes_mail/INBOX**
- **/tmp/joes_mail/private**
- **/tmp/joes_mail/private.msg**
- **/tmp/joes_mail/private/mom**

imsimport

The **imsimport** utility migrates UNIX **/var/mail** format folders into a Messaging Server message store.

The **imsimport** utility extracts the messages stored in **/var/mail** mailboxes and appends them to the corresponding users' mailbox in the Messaging Server message store. Files in the directory that are not in the standard UNIX mailbox format are skipped. If the corresponding users do not exist in the message store, **imsimport** creates them. When the user quota is exceeded, **imsimport** bypasses the message store quota enforcement, so the user does not receive an "over quota" message. The **imsimport** utility preserves the UIDVALIDTY, LASTUID, and UID in the **/var/mail** folder if it is possible.

The **imsimport** utility can be run while Messaging Server is running. If mail delivery is enabled for the mailbox you are importing, old mail can get mixed with new mail, so you might want to hold the delivery of this user during the migration. Mailbox access should not be a problem.

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

Note: **imsimport** does not use the IMAP server. However, the **stored** utility must be running to maintain message store integrity. The LDAP server should be running if **imsimport** is expected to create new users.

Table 64–24 shows the UW-IMAP MBOX header fields utilized by the **imsimport** utility.

Table 64–24 UW-IMAP MBOX Header Fields

Header	Syntax	Description
X-IMAPbase	UIDVALIDITY LASTUID KEYWORD...	Specifies IMAP uidvalidity, last uid, and a list of keywords.
X-UID	uid	Specifies IMAP uid.
X-KEYWORDS	keyword...	Specifies a list of space-separated keywords.
STATUS	[R][O]	Specifies status as R:\Seen, O:non-\Recent
X-STATUS	[A][F][T][D]	Specifies A:\Answered F:\Flagged T:\Draft D:\Deleted flags.
Content-Length	length	Specifies length of the message body in bytes.

Location: *MessagingServer_home*/**bin**/

Syntax

```
imsimport [-g] [-v 0|1|2] [-c y|n] [-d mailbox] [-n] [-e encoding] [-i]
[-b] -u user -s file
```

Options

Table 64–25 describes the options for the **imsimport** command.

Table 64–25 imsimport Command Options

Option	Description
-b	Subscribe new destination mailbox.
-c y | **n**	Answer yes/no to the question “Do you want to continue?”
-d *mailbox*	Destination folder name.
-e *encoding*	Mailbox name encoding (for example, **IMAP-mailbox-name**).
-g	Debug mode.
-i	Ignore content-length.
-n	If mailbox exists, create a new mailbox with a *date* extension (without this option, restore will append messages to the existing mailbox).
-s *file*	**/var/mail** source file name (path name).
-u *user*	User name.
-v *mode*	Verbose mode, 0-2. Default is 2.
-N	Generate event notifications.

Examples

imsimport migrates the specified **/var/mail/folder** for the specified user to the Messaging Server message store. If the destination folder is not specified, **imsimport** calls the destination folder by the same name as the source folder. In the following example, the command migrates the default **/var/mailINBOX** for the user **smith**, to the **INBOX**.

```
imsimport -u smith -s /var/mail/smith -d INBOX
```

Similarly, if you are trying to move a folder called test from **/home/smith/folders/** to the Messaging Server message store, use the following command:

```
imsimport -u smith -s /home/smith/folders/test -d test
```

If a destination folder called **test** already exists in the Messaging Server message store, **imsimport** appends the messages to the existing folder in the mailbox.

imsrestore

The **imsrestore** utility restores messages from the backup device into the message store. See "Backing Up and Restoring the Message Store" for more information.

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

Location: *MessagingServer_home***/bin**

Syntax

```
imsrestore -f device [-b blockfactor] [-e encoding] [-v mode] [-c y|n]
[-m mapfile] [-thngspxDS] [-i|-E] [-u file] [-r file] [-d path]
[-P partition] [name...]
```

Options

Table 64–26 describes the options for the **imsrestore** command.

Table 64–26 imsrestore Options

Option	Description
-b *num*	Use *num* as the block factor.
-c y | n	Answer yes/no to the question "Do you want to continue?"
-d *path*	Restore all mailboxes under *path* (for multiple incremental restores to new folders).
-D	Dump messages in **imcheck -D** format.
-e *encoding*	Mailbox name encoding (for example, **IMAP-mailbox-name**).
-E	Update and expunge existing messages.
-f *device*	Input device. Use **-** for **stdin**.
-h	Dump the header.
-g	Debug mode.
-i	Ignore existing messages (do not check for existing messages before restore).

Table 64–26 (Cont.) imsrestore Options

Option	Description
-m *file*	User name mapping file (for rename users).
-n	If mailbox exists, create a new mailbox with a *date* extension (without this option, restore will append messages to the existing mailbox).
-p	Restore mailboxes to original partition (classic store only).
-P *partition*	Restore all mailboxes to this partition (classic store only).
-r *file*	Reference file name (will restore all links in *file*).
-s	No seek.
-S	Dump records separator. Default is |.
-t	Print table of contents (will not restore anything).
-u *file*	Object name file.
-v *mode*	Verbose mode, 0-5. Default is 2.
-x	Restore stub.
name	Object names. Can be a logical path, internal name, or user ID. See "imsbackup" for description.

Examples

The following example restores the messages from the file **backupfile**:

```
imsrestore -f backupfile
```

The following example restores the messages for **joe** from the file **backupfile**:

```
imsrestore -f backupfile /primary/user/joe
```

The following example lists the content of the file **backupfile**:

```
imsrestore -f backupfile -t
```

The following example renames users in the file **mapfile**:

```
imsrestore -m mapfile -f backupfile
```

where the **mapfile** format is **oldname=newname**:

```
userA=user1
userB=user2
userC=user3
```

mboxutil

The **mboxutil** command lists, creates, deletes, renames, or moves mailboxes (folders). You can use the **mboxutil** command to report quota information, restore expunged messages, and list mailbox subscriptions.

> **Note:** Do not "kill" the **mboxutil** process (**SIGKILL** (**kill -9**) command) in the middle of execution. Otherwise, you will need to restart the Messaging Server process.

You can use POSIX regular expressions in the **mboxutil** command.

Requirements: Must be run locally on the Messaging Server. The **stored** utility must also be running.

Location: *MessagingServer_home*/**bin**

Mailbox Naming Conventions

You must specify mailbox names in the following format:

user/*userid*/*mailbox*

where *userid* is the user that owns the mailbox and *mailbox* is the name of the mailbox.

For hosted domains, *userid* is **uid@domain.**

For example, the following command creates the mailbox named **INBOX** for the user whose user ID is **crowe**. **INBOX** is the default mailbox for mail delivered to the user **crowe**.

```
mboxutil -c user/crowe/INBOX
```

Important: The name **INBOX** is reserved for each user's default mailbox. **INBOX** is the only folder name that is case-insensitive. All other folder names are case-sensitive.

Syntax

List mailboxes:

```
mboxutil -l [-B bound] [-E encoding] [-p pattern | -P regular expression] [-x | -s] [-D] [-z]
```

Create mailboxes:

```
mboxutil -c [-E encoding] {mailbox_name | -f file}
```

Delete mailboxes:

```
mboxutil -d [-E encoding] {-p pattern | -P regexp | -f file | mailbox}
```

Rename mailboxes:

```
mboxutil -r [-E encoding] {old_name new_name | -f file} [partition]
```

Expunge mailboxes:

```
mboxutil -e [-E encoding] [-p IMAP-mailbox-name pattern | -P regular expression]
```

Schedule cleanup:

```
mboxutil -C [-g num] [-E encoding] [-p IMAP-mailbox-name pattern | -P regular expression]
```

List orphan/inactive mailboxes:

```
mboxutil -o [-w file ] [-t number of days]
```

Restore expunged messages that have not been purged:

```
mboxutil -R [-E encoding][-m dest_mailbox_name][-U uid_sequence] mailbox_name
```

List personal mailbox subscriptions:

```
mboxutil -S [-n [-f file] | -u -f file]
```

Options

Table 64–27 describes the options for the **mboxutil** command.

Table 64–27 mboxutil Command Options

Option	Description
-B *bound*	Used with **-l** option to bound each column in output with user provided input or a single space as the default.
-c *mailbox*	Create mailboxes.
-C	Schedule cleanup.
-d *mailbox*	Delete mailboxes.
-D	List deleted mailboxes.
-e	Expunge mailboxes.
-E *encoding*	Mailbox name encoding (UTF-16 and UTF-32 are not currently supported. UTF-8 is a recommended substitute).
-f *file*	Create, delete, or rename mailboxes with names specified in *file*. When used with the **-S** option, input/output *file*. You can also use the **-f** *file* option with the *partition* option to move users to a new partition (on the same store host). For more information, see "To Move Mailboxes to a Different Disk Partition". Note: When using the **-f** *file* option, the file must consist of pairs of duplicate lines, because the other use of the **mboxutil -r** command is to rename folders. Thus, the first line of each pair is the old folder name and the second line is the new folder name. In the case of moving a user from one partition to another, the folder names are the same.
-g *num*	Minimum cleanup size, *num* must be > 0. Default is **store.cleanupsize**).
-l	List mailboxes.
-m *dest_mailbox_name*	Destination mailbox name. Duplicate restore is allowed on the Classic message store when the destination mailbox is different from the source mailbox.
-n	List personal non-existing mailbox subscriptions.
-o	List orphan/inactive mailboxes.
-p *pattern*	When used with the **-l** option, list mailboxes that match the specified *IMAP-mailbox-name* pattern. When used with the **-d**option, delete mailboxes that match the specified *IMAP-mailbox-name* pattern. When used with the **-e** option, expunge mailboxes that match the specified IMAP-mailbox-name pattern. When used with the **-C** option, check mailboxes that match the specified *IMAP-mailbox-name* pattern
-P *regexp*	List, delete, expunge, or check mailboxes that match the specified regular expression. **Note**: You must not use **-E** with **-P** because the regular expressions used with **mboxutil -P** are limited to the locale charset and are not impacted by the **-E** charset.
-r *oldname newname* [*partition*]	Move mailboxes to the specified partition.

Table 64–27 (Cont.) mboxutil Command Options

Option	Description
-R *mailbox*	Restore expunged messages that have not been purged
-s	Print mailbox name only.
-S	Lists personal mailbox subscriptions.
-t *days*	Include mailboxes that have not been accessed for the specified number of days (require last access update).
-u	Unsubscribe personal non-existing mailbox subscriptions.
-U *uid_sequence*	IMAP UID sequence set.
-w *file*	Save output to *file*.
-x	Print path and acl in addition to the other data.
-X *hours*	Exclude mailboxes last appended to within the specified number of hours. Default is 24.
-z	Print mailbox count in addition to the other data.

Note: When using the separator, check output consistency by trying the separator string as a single or double quoted string with escape sequences if required.

Examples

To list all mailboxes for all users:

```
mboxutil -l
```

To list all mailboxes and also include path and ACL information:

```
mboxutil -l -x
```

To list all mailboxes displaying only the mailbox names:

```
mboxutil -l -s
```

To create the default mailbox named **INBOX** for the user **daphne**:

```
mboxutil -c user/daphne/INBOX
```

To delete a mail folder named **projx** for the user **delilah**:

```
mboxutil -d user/delilah/projx
```

To delete the default mailbox named **INBOX** and all mail folders for the user **druscilla**:

```
mboxutil -d user/druscilla/INBOX
```

To rename Desdemona's mail folder from **memos** to **memos-april**:

```
mboxutil -r user/desdemona/memos user/desdemona/memos-april
```

To schedule a cleanup of Dorothea's mailbox if there are at least 50 expunged messages:

```
mboxutil -C -g 50 -p user/dorothea/*
```

To restore UID 1 to 3 from joe's INBOX to his Trash folder:

```
mboxutil -R -m user/joe/Trash -U 1:3 user/joe/INBOX
```

To move the mail account for the user **dimitria** to a new partition:

```
mboxutil -r user/dimitria/INBOX user/dimitria/INBOX partition
```

where partition specifies the name of the new partition.

To move the mail folder named **personal** for the user **dimitria** to a new partition:

```
mboxutil -r user/dimitria/personal user/dimitria/personal partition
```

To list orphaned mailboxes and mailboxes that have not been accessed in 60 days:

```
mboxutil -o -w orphanfile -t 60
```

The preceding example writes the list of orphaned and inactive mailboxes to a file named **orphanfile**.

To delete orphaned and inactive mailboxes:

```
mboxutil -d -f orphanfile
```

where **orphanfile** is a file that has stored a list of orphaned and inactive mailboxes identified with the *-o* option.

To list personal, non-existing mailbox subscriptions for user mailboxes listed in a file named **orphanfile**:

```
mboxutil -S -n -f orphanfile
```

To unsubscribe the non-existing mailbox subscription list generated by the previous example:

```
mboxutil -S -u -f orphanfile
```

The **mboxutil** command has stable output formats that we support and allow customers to parse. **mboxutil -l -p user/%/INBOX -s** lists all accounts (INBOXES) in the store. **mboxutil -o -w -t** lists inactive accounts. The formats are the same: one line per mailbox name. To generate an active list, remove the inactive mailboxes from the all mailbox list. Sample **mboxutil** output:

```
user/joe/INBOX
user/mary/INBOX
```

Mechanism to Schedule Cleanup Immediately

Normally, cleanup is scheduled automatically when mailboxes are expunged. This command can be used to schedule cleanup manually when:

- The storage is very full.
- **store.cleanupsize** is reduced.

For example, the following command schedules cleanup of mailboxes with at least 50 expunged messages. Messages are removed when **store.cleanupage** has expired.

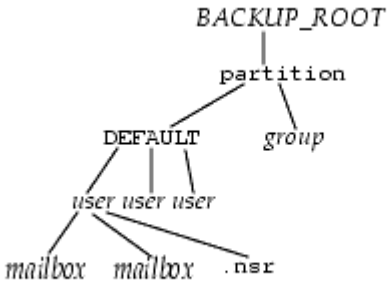
```
mboxutil -C -g 50
```

mkbackupdir

The **mkbackupdir** utility creates and synchronizes the backup directory with the information in the message store. It is used in conjunction with Solstice Backup (Legato Networker). The backup directory is an image of the message store. It does not contain the actual data. **mkbackupdir** scans the message store's user directory, compares it with the backup directory, and updates the backup directory with the new user names and mailbox names under the message store's user directory.

The backup directory is created to contain the information necessary for Networker to backup the message store at different levels (server, group, user, and mailbox). Figure 64–1 displays the structure.

Figure 64–1 Backup Directory Hierarchy



Location: *MessagingServer_home*/**bin**

Table 64–28 describes the variables in the backup directory contents.

Table 64–28 mkbackupdir Variables

Variable	Description
BACKUP_ROOT	Message store administrator root directory.
partition	Store partition.
group	System administrator-defined directories containing user directories. Breaking your message store into groups of user directories allows you to do concurrent backups of groups of user mailboxes.To create groups automatically, specify your groups in the *MessagingServer_home*/**config/backup-groups.conf** file. The format for specifying groups is:*groupname= patterngroupname* is the name of the directory under which the user and mailbox directories will be stored, and *pattern* is a folder name with IMAP wildcard characters specifying user directory names that will go under the *groupname* directory.
user	Name of the message store user.
folder	Name of the user mailbox.
mailbox	Name of the user mailbox.

The **mkbackupdir** utility creates:

- A default *group* directory (**ALL**) or the group directories defined in the **backup-groups.conf** configuration file. The following is a sample **backup-groups.conf** file:

```
groupA=a* (regexp)
groupB=b*
```

```
groupC=c*
.
.
.
```

- A *user* directory under the backup directory for each new user in the message store.
- A **0** length mailbox file for each mailbox.
- A **.nsr** file for each subdirectory that contains user mailboxes.

The **.nsr** file is the NSR configuration file that informs the Networker to invoke **imsasm**. **imsasm** then creates and interprets the data stream.

Each user mailbox contains files of zero length. This includes the **INBOX**, which is located under the *user* directory.

> **Note:** Make sure the backup directory is writable by the message store owner (mailsrv).

Syntax

```
mkbackupdir [-i|-f|-u] [-vg] [-a asm] [-e encoding] [-p path] [-t max_thread]
```

Options

Table 64–29 describes the options for the **mkbackupdir** command.

Table 64–29 mkbackupdir Options

Option	Description
-a *asm*	Creates **.nsr** files using the specified asm name. This can be used for when you have multiple instances of Messaging Server as in symmetric HA environments.
-e *encoding*	Specify an encoding option.
-f	Backs up the folders only. By default, all mailboxes are backed up.
-g	Executes the command in debug mode.
-i	Backs up the inbox only. By default, all mailboxes are backed up.
-p *path*	Specifies the directory for the backup image. This is a required option when **local.store.backupdir** is not configured.The max_thread option must be set between 1 and 256. Note: The Networker has a limitation of 64 characters for *saveset* name. If your default backup directory pathname is too long, you should use this option to specify another pathname.
-t *max_thread*	Specifies the number of threads that can be used to create the backup directory. The default is one thread for each partition, which is usually adequate. If you have many partitions, and you do not want **mkbackupdir** to consume all your resources, you can lower this number.
-u	User level backup. Instead of backing up each folder as a file, create a backup file per user.
-v	Executes the command in verbose mode.

Examples

To create the **mybackupdir** directory, enter the following:

```
mkbackupdir -p /mybackupdir
```

msprobe

msprobe is a daemon that probes servers to see if they respond to service requests. When **msprobe** detects a possible problem, it can, depending upon other configuration, potentially let the Watcher know (at which point the Watcher can attempt to restart a troubled component) and/or generate an alarm message; see the Watcher options and Alarm options, respectively in *Messaging Server Reference*.

For additional information on **msprobe**, see *Messaging Server Reference*.

Location: *MessagingServer_home*/**lib**

Syntax

```
msprobe [-d] [-n] [-r] [server]
```

Options

Table 64–30 describes the options for the **msprobe** command.

Table 64–30 msprobe Command Options

Option	Description
-d	Debug mode.
-n	Disable auto-restart.
-r	Report server, port, response_time, status, message to **stdout**. Status is: **OK**, **SLOW**, or **BAD**. Response time is in millisecond.

server	The accepted values for *server* are:
	`imap`
	`pop`
	`http`
	`ens`
	`cert`
	`job_controller`
	`smtp`
	`lmtp`
	`submit`
	`metermaid`
	`deploymap`
	If server is not specified, all enabled servers are probed.

Example

```
$ msprobe -r
imap,143,0,OK,"No error"
pop,110,-1,BAD,"Connection refused"
ens,7997,0,OK,"No error"
http,8990,0,OK,"No error"
job_controller,27442,0,OK,"No error"
smtp,25,35,SLOW,"No error"
submit,587,1,OK,"No error"

$ msprobe -r imap
imap,143,0,OK,"No error"
```

msuserpurge

When user and domain mailboxes are marked for deletion, the **msuserpurge** command purges those user and domain mailboxes from the message store. Specifically, this command scans the following domain and user status attributes in LDAP for a value of deleted: **inetDomainStatus**, **mailDomainStatus**, **inetUserStatus**, **mailUserStatus**. This command can be run at the command line, or can be scheduled for execution with the **configutil** option **local.sched.userpurge**.

Requirements: If run manually, it must be manually run locally on the messaging server.

Location: *MessagingServer_home*/**lib**

Syntax

```
msuserpurge [-v|-r] [-d domain] [-g days]
```

Options

Table 64–31 describes the options for the **msuserpurge** command.

Table 64–31 msuserpurge Options

Option	Description
-v	Verbose (list deleted users).
-r	Report (list all users, do not purge).
-g *days*	Specify a grace period.
-d *domain*	Specify a domain.

Examples

```
msuserpurge -d example.com -g 0
```

readership

The owner of an IMAP folder can grant permission for other users to access the folder. A folder that other users are allowed to access is called a shared folder. See "Managing Shared Folders" for more information. Users can modify the access rights on folders if their mail provides an interface to the **SETACL IMAP** commands. Administrators can use the **readership** utility to set or remove access rights on the folder, and to see how many users other than the owner are accessing a shared folder.

To list the rights on all shared folders, the mail administrator can use the **imcheck -d lright.db** command. To list rights on individual folders, use the **mboxutil -lxp** *folder* command.

Requirements: Must be run locally on the Messaging Server. The **stored** process must also be running.

Location: *MessagingServer_home*/**bin/**

Syntax

Count the number of users who have accessed their shared folders:

```
readership [ -d days |-p months ]
```

Set an acl for a mailbox:

```
readership [-e encoding] -s { -m pattern | mailbox } identifier acl
```

Options

Table 64–32 describes the options for the readership command.

Table 64–32 readership Command Options

Option	Description
-d *days*	Display the number of users who have accessed the shared folders within the specified days .
-e *encoding*	Mailbox name encoding.
-m *pattern*	MUTF-7 IMAP pattern.
-p *months*	Remove the seen flags from the users who have not accessed the shared folders within the specified months before counting.
-s *folderidentifieracl_right*	Sets ACL rights character for folder, where *folder* is the name of the folder for which you are setting rights, *identifier* is the person or group to whom you are assigning the rights, and *acl_rights* are the rights you are assigning. See Table 64–33, " ACL Rights Characters". To remove a rights setting, specify a null set of rights.
-s -m *folder_ patternidentifieracl_right*	Sets ACL rights as **-s** but on all folders matching the specified pattern.

Table 64–33 describes the ACL rights characters.

Table 64–33 ACL Rights Characters

Character	Description
l	lookup - User can see and subscribe to the shared folder. (IMAP commands allowed: **LIST** and **LSUB**)
r	read - Users can read the shared folder. (IMAP commands allowed: **SELECT**, **CHECK**, **FETCH**, **PARTIAL** , **SEARCH**, **COPY** from the folder)
s	seen - Directs the system to keep seen information across sessions. (Set IMAP **STORESEEN** flag)
w	write - Users can mark as read, and delete messages. (Set IMAP **STORE** flags, other than **SEEN** and **DELETED**)
i	insert - Users can copy and move email from one folder to another. (IMAP commands allowed: **APPEND**, **COPY** into folder)
p	post - Users can send mail to the shared folder email address. (No IMAP command needed)
k	create - Users can create new sub-folders. (IMAP command allowed: **CREATE**)
x	delete - Users can delete entries from the shared folder. (IMAP commands allowed: **EXPUNGE**, set **STOREDELETED** flag)
a	administer - Users have administrative privileges. (IMAP command allowed: **SETACL**)
t	delete - For messages, sets or clears \DELETED flag via **STORE**, or sets **DELETED** flag during **APPEND**/**COPY**.
e	expunge - Performs **EXPUNGE** and expunge as a part of **CLOSE**.
n	access - Retrieves annotation information about the folder (see RFC 5257 at http://tools.ietf.org/html/rfc5257#section-4.10).

Example

```
readership -s user/joe/golf mary lpr
readership -s -m user//joe/* mary lpr
```

reconstruct

The **reconstruct** utility rebuilds one or more mailboxes, or the master mailbox file (the mailboxes database), and repairs any inconsistencies. Use this utility to recover from almost any form of data corruption in the message store.

In a Cassandra message store, the mailboxes are stored in a Cassandra database. The only **reconstruct** command that works on Cassandra message store is:

```
reconstruct [-r] [-f | -n | -x] [-E encoding] [mailbox...]
```

On a Berkeley Database message store, a mailbox consists of files under the user partition directory. The mailboxes database is the mboxlist database.

Requirements: Must be run locally on the Messaging Server host; the **stored** utility must also be running.

Location: *MessagingServer_home***/bin**

Syntax

Repair/validate mailboxes:

```
reconstruct [-r] [-f | -n | -x] [-E encoding] [-p partition | mailbox...]
```

Repair mboxlist database (Berkeley Database only):

```
reconstruct -m [-p partition [-u user]]
```

Repair quotas (Berkeley Database only):

```
reconstruct -q [-p partition]
```

Repair subscriptions (Berkeley Database only):

```
reconstruct -s
```

Repair ACLs db (**lright.db**) (Berkeley Database only):

```
reconstruct -l
```

Repair annotation db (Berkeley Database only):

```
reconstruct -A [-u userid]
```

Compact a database (Berkeley Database only):

```
reconstruct -c [-f] db_file_name
```

Comprehensively repair all aspects of a mailbox (Berkeley Database only):

```
reconstruct -a [-t nthreads] [mailbox...]
```

Options

Table 64–34 describes the options for the **reconstruct** command.

Table 64–34 reconstruct Options

Option	Description
-a *mailbox...*	Comprehensively repair all aspects of a mailbox. If **-f** or **-n** is not specified, **reconstruct -a** will only rebuild folder index files if folder fails simple tests (for performance since parsing all messages is time consuming).
-A [**-u** *userid*]	Repair annotation database.
-c	Compact a database.
-E *encoding*	Mailbox name encoding.
-f	Force repair. When used with the **-a** option, force the rebuild of folder index files (parse all messages).
-l	Repair ACLs db (**lright.db**).
-m	Repair mboxlist database.
mailbox	Check the specified *mailbox*.
-n	Report only, no repair. When used with the **-a** option, report basic errors found but try not to rebuild most folder files if possible (This may be used to catch errors before repair so they may be studied.)
-p *partition*	Check or repair the specified partition.
-q	Repair quotas.
-r [*mailbox*]	Recursively process sub-folders (default if mailbox is not specified). When used with the **-a** option, if no mailbox is specified, **-r** is implied for the entire store.
-s	Repair subscriptions.
-t *nthreads*	Number of threads per process. Default is one per partition. Max is 64 single threaded if specific mailbox is specified.
-u *user*	Repair the specified user.
-x	Recover partially delivered message.

The *mailbox* argument indicates the mailbox to be repaired. You can specify one or more mailboxes. Mailboxes are specified with names in the format **user/***userid***/***sub-mailbox*, where *userid* is the user that owns the mailbox. For example, the inbox of the user **dulcinea** is entered as: **user/dulcinea/INBOX**.

Examples

The following command performs a reconstruct on a specific mailbox:

```
reconstruct user/dulcinea/INBOX
```

The following checks the specified mailbox, without performing a reconstruct:

```
reconstruct -n user/dulcinea/INBOX
```

The following command checks all mailboxes in the message store:

```
reconstruct -n -r
```

To Rebuild Mailboxes

To rebuild mailboxes, use the **-r** option. You should use this option when:

- Accessing a mailbox returns one of the following errors: "System I/O error" or "Mailbox has an invalid format".

- Accessing a mailbox causes the server to crash.
- Files have been added to or removed from the spool directory.

reconstruct -r first runs a consistency check. It reports any inconsistencies and rebuilds only if it detects any problems. Consequently, performance of the **reconstruct** utility is improved with this release. You can use **reconstruct** as described in the following examples: To rebuild the spool area for the mailboxes belonging to the user **daphne**, use the following command:

```
reconstruct -r user/daphne
```

To rebuild the spool area for all mailboxes listed in the mailbox database:

```
reconstruct -r
```

You must use this option with caution, however, because rebuilding the spool area for all mailboxes listed in the mailbox database can take a very long time for large message stores. (See the discussion about **reconstruct** Performance below.) A better method for failure recovery might be to use multiple disks for the store. If one disk goes down, the entire store does not. If a disk becomes corrupt, you need only rebuild a portion of the store by using the **-p** option as follows:

```
reconstruct -r -p subpartition
```

To rebuild mailboxes listed in the command-line argument only if they are in the **primary** partition:

```
reconstruct -p primary mbox1 mbox2 mbox3
```

If you do need to rebuild all mailboxes in the **primary** partition:

```
reconstruct -r -p primary
```

If you want to force reconstruct to rebuild a folder without performing a consistency check, use the **-f** option. For example, the following command forces a reconstruct of the user folder **daphne**:

```
reconstruct -f -r user/daphne
```

To check all mailboxes without fixing them, use the **-n** option as follows:

```
reconstruct -r -n
```

reconstruct -r -f focuses on fixing the folder index files. It assumes the folder record is good, and the peruser record is good. There are other commands to address these other data areas, such as reconstruct **-m**.

reconstruct -a attempts to address these all at once.

So if you think you need to repair an index file, but you're not sure if it is in the **folder.db**, you should not need to worry about running a **reconstruct -m** first, and whether the index corruption will be handled correctly there, or if both these will result in something that will conflict with the peruser entry later.

If you know your problem is with the index file, and there are no other complications, then you can go ahead and use **reconstruct -r -f** to save time.

Checking and Repairing Mailboxes

To perform a high-level consistency check and repair of the mailboxes database:

```
reconstruct -m
```

To perform a consistency check and repair of the primary partition:

```
reconstruct -p primary -m
```

> **Note:** Running **reconstruct** with the **-P** and **-m** flags together will not fix **lright.db**. This is because fixing the **lright.db** requires scanning the ACLs for every user in the message store. Performing this for every partition is not very efficient. To fix the **lright.db** run **reconstruct -l**

To perform a consistency check and repair of an individual user's mailbox named **john**:

```
reconstruct -p primary -u john -m
```

You should use the **-m** option when:

- One or more directories were removed from the store spool area, so the mailbox database entries also need to be removed.
- One or more directories were restored to the store spool area, so the mailbox database entries also need to be added.
- The **stored -d** option is unable to make the database consistent. If the **stored -d** option is unable to make the database consistent, you should perform the following steps in the order indicated:
  - Shut down all servers.
  - Remove all files in *store_root***/mboxlist**.
  - Restart the server processes.
  - Run **reconstruct -m** to build a new mailboxes database from the contents of the spool area.

reconstruct Performance

The time it takes **reconstruct** to perform an operation depends on the following factors:

- The kind of operation being performed and the options chosen
- Disk performance
- The number of folders when running **reconstruct -m**
- The number of messages when running **reconstruct -r**
- The overall size of the message store
- What other processes the system is running and how busy the system is
- Whether or not there is ongoing POP, IMAP, HTTP, or SMTP activity. Note that **reconstruct** is designed to run with the store services up. It is not necessary to keep the store offline to run **reconstruct**.

The **reconstruct -r** option performs an initial consistency check; this check improves **reconstruct** performance depending on how many folders must be rebuilt. The following performance was found with a system with approximately 2400 users, a message store of 85GB, and concurrent POP, IMAP, or SMTP activity on the server:

- **reconstruct -m** took about 1 hour

- **reconstruct -r -f** took about 18 hours

> **Note:** A **reconstruct** operation may take significantly less time if the server is not performing ongoing POP, IMAP, HTTP, or SMTP activity.

rehostuser

The **rehostuser** utility enables you to move a Messaging Server user's mail store from one mailhost to another. It also disconnects any active session, locks the store to ensure atomicity of the move from the user's perspective (no loss of data, flag change, and so on), changes the user's LDAP entry, flushes LDAP caches as necessary, and causes any queued mail to be rerouted to the new store.

Requirements: The following setup and configuration is required before you can use **rehostuser**:

1. Ensure that both the source and destination mailhosts are running Messaging Server 8.0.1 or later.
2. For Cassandra message store, you must use Local Mail Transfer Protocol (LMTP) to deliver messages into the Cassandra message store. If you are not currently using LMTP, you must configure it before you can use **rehostuser** to move the user's mail store from a classic message store to Cassandra message store.
3. Ensure that both the source and destination mailhosts have IMAP IDLE configured. See "Configuring IMAP IDLE" for more information.
4. Configure **ssh** on the source and destination mail hosts to allow the mail server user to perform a remote login. Even if you start **rehostuser** as **root**, it still needs to run and execute **ssh** as the mail server user.
5. Make sure that **ugldapbinddn** has read and write access to the **mailHost**, **mailUserStatus**, and **mailMessageStore** attributes. New installations of Messaging Server should have this access made as part of the installation or upgrade process. The **rehostuser** utility checks for these permissions at startup and exits with an error if the permissions are insufficient.
6. The source mailhost must be capable of sending mail to the destination mailhost. In particular, the **tcp_intranet** channel on the source mailhost must be open for relaying, the mail must be routable to the new mailhost (directly or following **mx** records), and the destination mailhost must accept mail coming from the source mailhost.
7. **rehostuser** needs to consider LDAP replication delay. If either the source or destination Messaging Server system (or both) are configured to use an LDAP replica server instead of a master, there may be problems where attribute changes do not show up on the replica as quickly as required. This issue can be addressed as follows:
   a. If a Messaging Server points to a single Directory Server, configure the Directory Server in multi-master replication (MMR) mode so that it writes updates to its local database and replicates the change. (Note that MMR is the recommended way to setup Directory Server as opposed to the old slave/master model where slaves refer writes to the master, then have to wait for replication to get the updated data).
   b. If Directory Server failover capability is needed, use Directory Server proxy and enable one of the client affinity modes that guarantees that any single client will always see the updated data it just wrote.

c. OpendDS 2.0, (possibly released 03/2009), may offer an *assured replication* feature where once a write is committed, it is guaranteed that the whole farm of replicated

Syntax

```
rehostuser -u userid -d mailhost [-p partition] [-c imsconnutil]
[-b imsbackup] [-s ssh] [-r imsrestore] [-e] [-x] [-n]
[-o iss_src -t iss_dest -y ssh_user [-z src_path] [-w dest_path]]
```

Options

Table 64–35 describes the options for the **rehostuser** command.

Table 64–35 rehostuser Options

Option	Description	Default
-b	**imsbackup** path.	*MessagingServer_home***/bin**.
-c	**imsconnutil** path.	*MessagingServer_home***/bin**.
-d	Destination mailhost.	NA
-e	Extended availability of user's mailbox during move.	NA
-n	Do not remove source mailbox.	NA
-o	Move ISS index from the specified source host.	NA
-p	Destination partition.	NA
-r	**imsrestore** path on remote host.	*MessagingServer_home***/bin**
-s	**ssh** path. Do not substitute **rsh** for **ssh**. The **rsh** command cannot return **imsrestore** exit status.	**/usr/bin/ssh**
-t	Move ISS index to the specified destination host.	NA
-u	User to move.	NA
-w	ISS destination installation path.	Source install path
-x	Move expunged message files.	NA
-y	ISS ssh/scp user name.	NA
-z	ISS source install path.	*IndexingServer_home*

rehostuser Example

This example shows how to move **user2** to mail server **bigdipper** where the Messaging Server software is installed in the **/opt/sun/comms/messaging** directory and the mail server user is **mailuser**.

```
rehostuser -u user2 -d bigdipper.example.com -r
/opt/sun/comms/messaging/bin/imsrestore -s "/usr/bin/ssh -x -l mailuser"

disconnecting user2
-----------------------------------------
Tape Version : 2
Backup Date : 2008/01/08 17:45:16
Message Store : host1.example.com
Block factor : 20
-----------------------------------------
/second/user/user2/INBOX restoring...
/second/user/user2/INBOX/1 restoring...
```

```
/second/user/user2/INBOX/2 restoring...
/second/user/user2/INBOX/3 restoring...
/second/user/user2/folder20 restoring...
/second/user/user2/folder22 restoring...
/second/user/user2/folder33 restoring...
/second/user/user2/folder38 restoring...
/second/user/user2/folder49 restoring...
Mailbox user2 copied successfully.
Updated LDAP entry for uid=user2, ou=People, o=example.com, o=usergroup
Source mailbox deleted successfully.
```

relinker

Note: The **relinker** feature is intended to repair the situation where the normal single-copy nature of the message store has become broken for some reason. You should only need to use the relinker if you have done something which could have caused duplicate messages to become individual copies instead of the normal single-copy. This feature is not the normal way the store normally accomplishes its single-copy feature. You should not need to keep the real-time relinker feature enabled for long periods of time. You should not need to use the relinker command on an ongoing basis. You should only need this feature if you have done (or will soon be doing) something which would break the single-copy feature of the store. See "How the Message Store Works" for more information about single-copy.

relinker finds and relinks duplicate messages. Refer to "Reducing Message Store Size Due to Duplicate Storage" for more information.

Requirements: You may run **relinker** as **root** or **mailsrv**.

Location: *MessagingServer_home*/**bin**

Syntax

```
relinker [-p partitionname] [-d]
```

Options

Table 64–36 describes the options for the **relinker** command.

Table 64–36 relinker Options

Option	Description
-d	Delete digest repository.
-p *partitionname*	Process only this partition.

Examples

To relink a message store:

```
relinker

Processing partition: primary
Scanning digest repository...
Processing user directories...
-------------------------------------------------------
Partition statistics                Before        After
```

```
-------------------------------------------------------
Total messages                          73          73
Unique messages                         41          40
Message digests in repository            1           1
Space used                              55Kb        51Kb
Space savings from single-copy          40Kb        43Kb
-------------------------------------------------------

Note: run-time relinker (local.store.relinker.enabled) is not
enabled, so the repository will not be automatically purged.
When you're done with relinker, remember to purge the
repository by running "relinker -d"
```

After the "Scanning digest repository..." text shown above, **relinker** displays another '.' for every 100,000 messages message digests it finds in the repository and another '.' for every 100,000 messages as it is scanning messages. This indicates the progress of the **relinker** command.

To delete the digest repository:

```
relinker -d

Processing partition: primary
Purging digest repository...
-------------------------------------------------------
Partition statistics               Before        After
-------------------------------------------------------
Message digests in repository           1            0
-------------------------------------------------------
```

Note that command-line **relinker** is also controlled by the **store.relinker.maxage** (Unified Configuration) or **local.store.relinker.maxage** (legacy configuration) option, which defaults to 24 (hours). To have command-line **relinker** consider all messages in the store, rather than just those delivered in the past day:

For Unified Configuration, run:

```
msconfig set store.relinker.maxage -v 24
msconfig set store.relinker.maxage -1
```

For legacy configuration, run:

```
configutil -o local.store.relinker.maxage 24
configutil -o local.store.relinker.maxage -v -1
OK SET
relinker

Processing partition: primary
Scanning digest repository...
Processing user directories...
-------------------------------------------------------
Partition statistics               Before        After
-------------------------------------------------------
Total messages                          75          75
Unique messages                         40          22
Message digests in repository            1          22
Space used                              51Kb        29Kb
Space savings from single-copy          50Kb        73Kb
-------------------------------------------------------

Note: run-time relinker (local.store.relinker.enabled) is not
enabled, so the repository will not be automatically purged.
```

```
When you're done with relinker, remember to purge the
repository by running "relinker -d"
# msconfig set store.relinker.maxage 24 (Unified Configuration)
# configutil -o local.store.relinker.maxage -v 24 (legacy configuration)
OK SET
#
```

stored

The **stored** utility starts a daemon that performs the following functions:

- Performs checkpoint database transactions.
- Deadlock detection and rollback of deadlocked database transactions.
- Cleanup of temporary files and lock files on startup.
- Creates a database snapshot archive.
- Database recovery as necessary (see "Message Store Automatic Recovery On Startup" for more information.)

If any server daemon crashes, you must stop all daemons and restart all daemons including **stored**.

Requirements: Must be run locally on the Messaging Server.

Location: *MessagingServer_home***/lib/**

Syntax

To run **stored** as a daemon process:

```
stored -r | -R | -t [-v] | [-m] -d [site] | [-m]
```

Options

Table 64–37 describes the options for the **stored** command.

Table 64–37 stored Options

Option	Description
-r	Remove db environment.
-R	Remove db **tmp** files.
-t	Test stored readiness.
-v	Display diagnostics to **stdout**/**stderr**.
-m	Berkeley Database replication master.
-d *site*	Delete a **mboxlist** replication site. Site format is *host*[:*port*]. Default site is local site.

Part V

Managing the Cassandra Message Store

Part V describes the Cassandra message store and Solr indexing and search engine, and contains the following chapters:

65

Overview of Cassandra Message Store

This chapter provides an overview of the Oracle Communications Messaging Server Cassandra message store.

About the Cassandra Message Store

Apache Cassandra message store is a free and open-source NoSQL database designed for large amounts of data.

In Messaging Server, the Cassandra message store consists of *keyspaces*. A keyspace, in a NoSQL data store, is a namespace that defines data replication on nodes. (It resembles the schema concept in Relational database management systems.) The Messaging Server Cassandra message store consists of the following keyspaces:

1. **msg**: Contains the email message "blobs." It can be very large.
2. **mbox**: Contains user and mailbox metadata. It is relatively small and has a lot of mutations.
3. **cache**: Contains cache tables for converted message blobs.

Differences in Cassandra Message Store and Classic Message Store

The Cassandra message store differs from the classic message store in the following ways:

- The Cassandra message store uses a single maintenance queue. Classic message store uses three maintenance queues.
- The **imcheck -d** and **imcheck -s** commands are supported only by classic message store.
- Other **imcheck** command differences with classic message store:
  - **imcheck -m** output: The Cassandra message store does not have start offset, cache ID and cache offset.
  - **imcheck -q**: The RecNo is not unique in Cassandra message store.
- When you rename a mailbox on Cassandra, the subscription list is updated automatically. IMAP has **SUBSCRIBE** and **UNSUBSCRIBE** commands, which are typically used for shared folders. There is no requirement in the IMAP specification for the subscription to recognize the new name when a shared folder is renamed. Classic message store subscriptions do not follow renames, whereas Cassandra message store does.

- Cassandra message store folder purge is always deferred. This is optional in classic message store. Also, Cassandra message store mailbox purge and cleanup tasks are combined into one task.
- Cassandra message store does not advertise the IMAP ACL extension (RFC 4314). Although the ACL commands are implemented, the semantics of ACLs with respect to shared folders are not fully implemented. As a result, it is inappropriate to advertise the IMAP ACL extension in this version of Cassandra message store.
- Cassandra message store does not support the IMAP ANNOTATE capability. (The ANNOTATE extension to IMAP permits clients and servers to maintain "meta data" for messages, or individual message parts, stored in a mailbox on the server.)
- In Cassandra message store, delete user does not remove access rights from the ACL of the other user's folder. When a user is deleted and recreated, shared folders are still accessible.
- User ID (puid) and folder ID (fid) are permanently unique in Cassandra message store.
- Cassandra message store does not use a message store partition. Thus, you no longer are able to perform maintenance tasks such as expire, backup, and reconstruct by store partition.
- The **imexpire** command cannot install expire rules by partition.
- Cassandra message store supports only mailbox reconstruct.
- Cassandra message store only supports Unified Configuration.
- IMAP subscribe: Cassandra message store does not permit users to subscribe to non-existing folders.
- Specific differences that apply to Messaging Server 8.0.x:
  - The **reconstruct -x** command has been obsoleted (it is now always enabled).
  - **imcheck -e** output: The File type column has been removed.
  - **imcheck -q** output: RecNo has a queue prefix.
- Classic message store allows use of invalid identifiers in IMAP ACLs; they are simply ignored when evaluating the ACL. Cassandra message store does not allow use of invalid identifiers in an ACL; an attempt to use an invalid identifier will result in no change to the ACL.
- Cassandra message store supports both an external identifier and a persistent identifier for each user. Changing the persistent identifier of an existing user is not recommended. The external identifier for a user can be changed freely and is used for ACLs and shared folders instead of the persistent identifier. See the topic on User Identifiers in *Messaging Server Reference*, as well as the **ldap_permid** and **ldap_extid** options for more information.
- A **SETACL** command for cross-domains is not allowed. Thus, for a Cassandra message store, the value for the **store.privatesharedfolders.restrictdomain** configuration option is always **1** (to disallow regular users from sharing private folders to users in another domain.) In addition, neither the default domain nor the canonical domain name can ever be changed.

Scaling Your Cassandra Message Store Deployment Horizontally

You can scale your Cassandra message store deployment horizontally by adding more nodes to an existing deployment. For adding a node, datacenter, or cluster, see the Cassandra documentation.

Adding an Access-Tier Node (IMAP/LMTP Server/enpd)

To add a message access tier (IMAP/LMTP-server/**enpd**) node to a store affinity group:

1. Configure and start the new server.
2. Test the new server.

   You should be able to access IMAP accounts through the new server, although LMTP, IMAP IDLE, and notifications for IMAP APPEND operations are not yet live.
3. Update the recipe that you used to set up store affinity groups to add this server to the store affinity group in question.
4. Run the store affinity group recipe on all message access tier hosts in the deployment and use the **refresh** command to activate the change in the deployment.

   EA Note: For Early Access, this is technically only necessary on the store affinity group in question. However, when shared folder support is added, that will no longer be true.
5. At this point, IMAP IDLE should work on the new host but it is not yet servicing customer requests.
6. Run the recipe on all front-end tier hosts in the deployment and use **refresh** for MMP/**mshttpd**, and **imsimta reload** to update the store affinity group configuration.

Creating a new store affinity group is a similar procedure but you must have at least two hosts in the new group.

Managing Your Cassandra Message Store Availability

To remove a node, datacenter, or cluster, see the Cassandra documentation.

Removing an Access-Tier Node (IMAP/LMTP Server/enpd)

To remove a message access tier (IMAP/LMTP-server/**enpd**) node from a store affinity group:

- Update the store affinity group recipe to remove the node.

  Remember, keep at least two nodes in an affinity group for robustness.
- Run the recipe on the front-end tier, and use the **refresh** or **imsimta** command to update information.
- Run the recipe on the message access tier and refresh (run on host being removed last).

- Wait for active IMAP/LMTP connections to finish or forcefully disconnect users with the **imsconnutil** command or by stopping the server.

66

About Elasticsearch

This chapter provides an overview of the Elasticsearch indexing and search service integrated with the Oracle Communications Messaging Server different message stores.

Elasticsearch Indexing and Search

The Classic message store and Cassandra message store supports Elasticsearch indexing and search engine. Elasticsearch is a distributed search engine from the Apache Lucene project. It provides full text search services with built-in high availability, replication, horizontal scaling, and automatic load balancing.

For Elasticsearch, a message store deployment includes an Elasticsearch cluster. Multiple Classic message stores can share the same Elasticsearch cluster. Typically, all message store servers in a deployment share the same Elasticsearch clusters.

The classic message store does not enable index and search services by default. When Elasticsearch is enabled, the message store sends the message content to Elasticsearch for indexing when messages are appended to the mailboxes; the IMAP server sends search queries to Elasticsearch to perform textual searches. The message store uses Indexed Search Converter (ISC) to convert binary message content to text before indexing. Therefore, ISC must be configured on the message store server when Elasticsearch is enabled.

In classic store, ISC uses the file system to cache the text content. In Cassandra store, ISC uses Cassandra to store the cache content.

> **Note:** ISC requires read access to the message store in order to convert the content of a message. When used with classic store, ISC can be co-located with other message store processes or can run on a separate server with shared filesystem access to message store data.

Enabling Elasticsearch

To enable Elasticsearch on the message store, set the following options:

```
store.searchengine = elastic
elasticsearch.hostlist = a space separated list of elasticsearch hosts
elasticsearch.numshards = number of shards in elasticsearch cluster
elasticsearch.numreplicas = number of replicas in elasticsearch cluster
elasticsearch.storesource = false, if the storage space is limited
```

where,

searchengine specifies the message store search engine type.when the **searchengine** is set to elastic, Elasticsearch options control IMAP search operations.

hostlist specifies a space separated list of Elasticsearch hosts. Host format is *host[:port*]. If port is not specified, the port number is determined based on the setting of the **port** Elasticsearch option.

numshards specifies the number of shards in the Elasticsearch cluster. The **numshards** is used by store to create the message store index in Elasticsearch. The number of shards cannot be changed after the index is created. (Default : 5)

numreplicas specifies the number of replicas in the Elasticsearch cluster. The **numreplicas** is used by stored to create the Elasticsearch message store index. The message store does not update the number of replicas after the index is created. The number can be updated in Elasticsearch manually. (Default : 1)

The **_source** field in Elasticsearch contains the document body that was passed at index time. If **storesource** is enabled, the message store will create the Elasticsearch store/message index mapping with the **_source** field enabled; IMAP copy will use the **_source** field from Elasticsearch to index the message on the destination folder. The **_source** field data consumes a lot of disk space. You might want to disable the **_source** field if storage space is limited. Disabling the **_source** field will disable the ability to reindex, upgrade or repair index from Elasticsearch. (Default : false)

For a list of Elasticsearch options, see *Messaging Server Reference*.

Differences Between Elasticsearch and Brute-Force IMAP Searching

This section describes the differences in IMAP searches between Elasticsearch and Brute-force.

Wildcard Search

Wildcard searching refers to using a single character, such as an asterisk (*), to represent several characters or an empty string in an email search. Wildcard searching differentiates between the following types:

- Prefix wildcard search: The wildcard is used at the end of the string, for example: appl*
- Suffix wildcard search: The wildcard is used at the beginning of the string, for example: *pple
- Substring search: The wildcard is used at the beginning and end of the string: *ppl*

When a search is issued in brute-force, it linearly searches through email content to match the exact sequence of characters provided in the search command. This implicitly includes prefix and suffix search.

For example, consider an email body that contains the following text:

```
Hello World! This is a test.
```

All the following searches in brute-force return this email:

```
search body world
search body wor
search body llo
search body "lo Wor"
search body !
```

In Elasticsearch, the default search does not include prefix, suffix, and substring search. You must set the following configuration options to be able to conduct prefix and suffix search in Elasticsearch:

- To enable prefix search:

```
msconfig set imap.indexer.prefix_search "subject body from to cc text bcc"
```

- To enable suffix search:

```
msconfig set imap.indexer.suffix_search "subject body from to cc text bcc"
```

- To enable substring search:

```
msconfig set imap.indexer.substring_search "subject body from to cc text bcc"
```

However, in Elasticsearch, wildcarding is allowed only for single-word search terms. Elasticsearch does not support wildcarding within phrases. Thus, even with wildcarding enabled, the following search using the previous example does not return the email in Elasticsearch.

```
search body "lo Wor"
```

Also, the following search returns all email message in that folder because special characters like punctuation marks are discarded by both indexing and search:

```
search body !
```

The following searches in Elasticsearch do return the example email when prefix and suffix searches are enabled:

```
search body world
search body wor
search body llo
```

Special Characters and Searching

The Elasticsearch standard tokenizer splits text fields into tokens, treating whitespace and special characters (like punctuation) as delimiters. Delimiter characters are discarded, with the following exceptions:

- Periods (dots) that are not followed by whitespace are kept as part of the token, including Internet domain names.
- The "@" character is among the set of token-splitting punctuation, so email addresses are not preserved as single tokens.

This means that if an email field contains words with special characters, such as "@," Elasticsearch splits that word around the special character and indexes it as two words.

For email address wildcard search, Elasticsearch treats the email address as two words in sequence (user name and domain name). The term in the search should not mix user name and domain together when wildcarding is enabled. That is, searching on **user1@example.com** is not a valid search in Elasticsearch.

The following example wildcard email address searches are all valid in Elasticsearch:

- Prefix search: `FROM user7642@example.com`:

```
SEARCH FROM user
SEARCH FROM u
SEARCH FROM exampl
SEARCH FROM example.c
```

- Suffix search: `TO test1.central.example.com`:

```
SEARCH TO st1
SEARCH TO st1@central.example.com
SEARCH TO mple.com
SEARCH TO le.com
SEARCH TO om
```

- Substring search: `CC: user7170@us.example.com`:

```
SEARCH CC ser71
SEARCH CC exampl
SEARCH CC s.exampl
SEARCH CC s.example.co
```

The following example wildcard email address searches are invalid in Elasticsearch:

- `FROM jean-marie@oracle.com`:

```
SEARCH FROM ean-ma
SEARCH FROM marie@or
```

Words Not Indexed by Elasticsearch

The following words are filtered out from emails by the Elasticsearch **English stopwords** filter. These words are not indexed by Elasticsearch, and so are not searchable.

"a", "an", "and", "are", "as", "at", "be", "but", "by", "for", "if", "in",

"into", "is", "it", "no", "not", "of", "on", "or", "such", "that", "the",

"their", "then", "there", "these", "they", "this", "to", "was", "will",

"with"