InterMail® Mx

OPERATIONS GUIDE

Software Version 5.1

Document Version 1
Published March, 2000

Software.com®
THE INTERNET INFRASTRUCTURE COMPANY

Software.com, Inc.

www.software.com

525 Anacapa Street
Santa Barbara, CA 93101-1603
Tel: (805) 882-2470
Fax: (805) 882-2473

10 Maguire Road, Suite 400
Lexington, MA 02421-3130
Tel: (781) 274-7000
Fax: (781) 674-1080

InterMail Mx Operations Guide, Version 5.1

Document number: MO-000310

Table of Contents

Preface

Welcome to InterMail Mx!

The *InterMail Mx Operations Guide* includes instructions for the operation and administration of your InterMail system.

Intended Audience

This guide assumes that you are experienced with the UNIX operating system at a system administration level, and that you have an understanding of databases as well as of networking protocols and related technology.

Organization

This manual is organized as follows:

- Chapter 1, InterMail Overview, provides an overview of the InterMail Mx system.
- Chapter 2, Getting Started, outlines tasks that you must perform before the InterMail system goes into production mode. It also discusses some of the basic operations associated with administering InterMail.
- Chapter 3, Site Management, familiarizes you with classes of service, domains, and account management to help you effectively set up InterMail sites and organizations.
- Chapter 4, System Configuration, describes tasks related to viewing and updating the configuration database, and includes instructions for making configuration changes.
- Chapter 5, Security, describes the security features of the InterMail system, as well as two general types of security issues: the flow of unwanted message traffic through the system, and the viewing and retrieval of mail by unauthorized individuals.
- Chapter 6, Mail Routing, describes mail routing options of the InterMail system. These options help control the direction of mail flow.

- Chapter 7, Managing Mail in Process, describes temporary storage of mail, including the location and organization of temporarily stored mail, the automatic and manual handling of deferred mail, and mail queuing options.
- Chapter 8, Managing Mail Storage, describes the operations surrounding persistent storage of mail.
- Chapter 9, System Monitoring, describes monitoring and maintenance tasks you must perform to keep your InterMail system performing optimally.
- Chapter 10, Monitoring Tools, describes monitoring tools you can use to analyze and maintain the performance of your InterMail system.
- Chapter 11, Backup and Recovery, familiarizes you with the concepts of backup and recovery in an InterMail system to help you design effective backup and recovery procedures for your own environment.
- Chapter 12, Troubleshooting, describes possible problems with the InterMail system and suggested solutions.

Conventions

Convention	Description	Example
`$` at the start of a string	An environment variable (set at the time of installation)	`$spoolDir`
monospace type	• Commands • Directory and file names • Hostnames • Configuration keys and their values • Utility names	`imdbcontrol` command `cron` utility Set this key to `true`.
`<angle brackets>` in a command	A required variable	`imboxget <address>`
`[square brackets]` in a command	An optional parameter	`imctrl [-verbose]`
`|` (a vertical bar) between options in a command	Exclusive options, of which you can use only one	`impwdhash -a [md5-po|unix]`
`{braces}` around options in a command	A list of options, one of which is required	`immsgdelete {<msgID>...|-all}`
`...` (an ellipsis) after an optional entry in a command	An option for which you may have multiple entries	`imbucketscreate [<c1...cn>]`
`boldface` in an example	User input	`venus% **imservctrl stop**`

Related Documentation

This manual is one of a set. Other manuals in this set are:

- *InterMail Mx Reference Guide*, which contains contains background information about InterMail's servers and databases, configuration keys, directory management utilities, administrative utilities, APIs, and event messages.
- *Integrated Services Directory User Guide*, which contains conceptual information about the Integrated Services Directory (ISD) architecture, configuration keys, directory schema, directory structure, and directory management utilities, as well as procedural information to help you customize the ISD.
- *InterMail Mx Installation Guide*, which provides instructions for installing InterMail.

- *InterMail Mx Upgrade Guide*, which provides instructions for upgrading from previous versions of InterMail.
- *InterMail Mx Migration Guide*, which provides instructions for migrating to InterMail from the SendMail or Post.Office messaging product.

Questions and Comments

Your feedback is important to us! To suggest improvements or make comments on the content of this manual, please send e-mail to InterMail.Manual@Software.com.

1

InterMail Overview

InterMail is a scalable, high-performance, native Internet e-mail system for very large messaging environments. Each InterMail installation is a unique configuration created from a common set of software components. This chapter provides an overview of the InterMail system components and explains how they work together to speed message flow through the system.

This chapter describes:

- InterMail servers
- Information storage
- Supporting technology

InterMail Servers

The InterMail messaging system includes multiple servers, each with unique responsibilities. InterMail servers can be distributed among separate host machines for flexible configuration and system scaling.

A complete InterMail installation requires at least one server of each type, but typically includes multiple instances of all but the Configuration server. It is possible to add more servers as the message traffic or the number of end users increases.

The InterMail servers are categorized by function:

- Directory checking
- Message delivery
- Message storage
- Message retrieval
- System management

InterMail message flow shows the interaction of these servers.

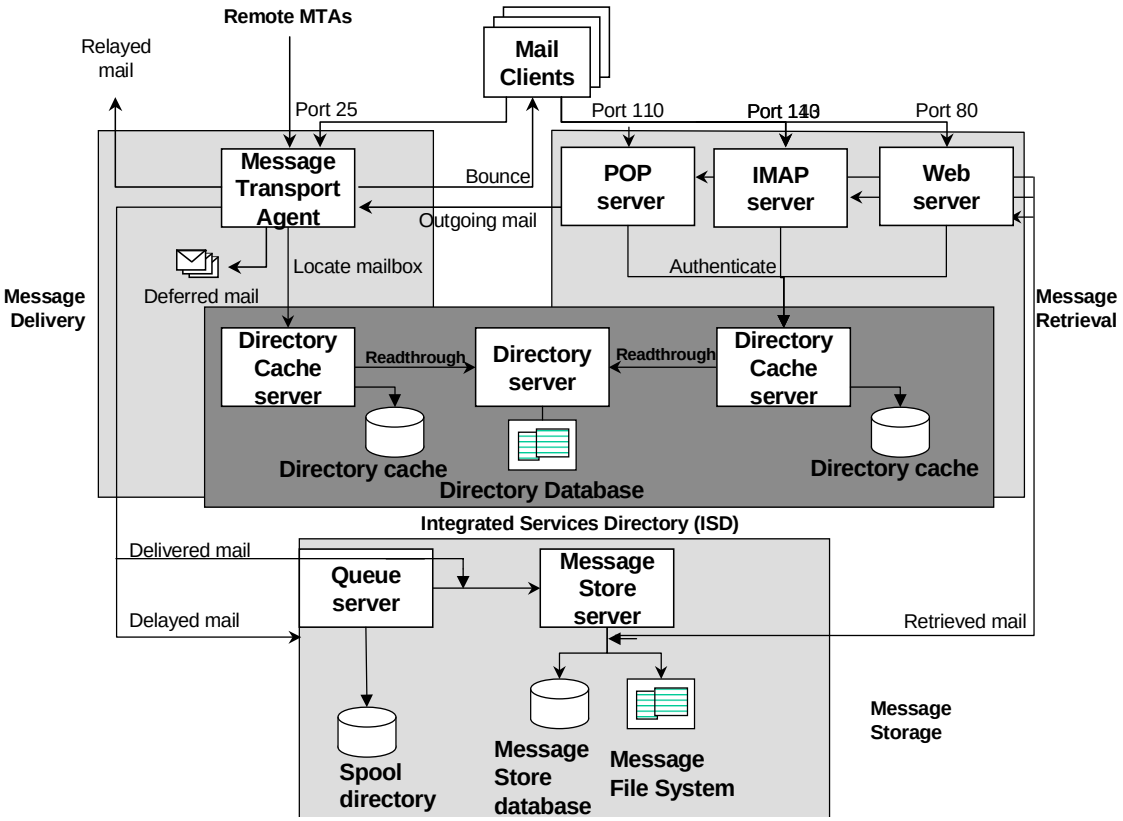

Figure 1 InterMail message flow

Directory Checking

The heart of the InterMail system is the Integrated Services Directory (ISD), as Figure 1 illustrates. The ISD is involved in message delivery, message storage, and message retrieval. It consists of the Directory Cache server, Directory server, Directory Cache database, and Directory database.

The **Directory Cache server** responds to requests for account information from other InterMail servers. It does this by consulting its cache, which contains a local copy of all or part of the Directory database. It communicates with the Directory server at configurable intervals to update its account information. This eliminates bottlenecks to the Directory database, ensuring that responses to all of the InterMail servers are fast, no matter how large the number of accounts in the Directory database and the number of messages being delivered per second across the entire system.

You can partition directory information among multiple caches running on different host machines. However, only one Directory Cache server can run on each host.

The **Directory server** relays account information between the Directory database and the Directory Cache servers. It is the only server to communicate directly with the Directory database, which is the ultimate authority for all Directory data.

Each Directory server automatically updates its local cache at regular intervals with the most recent information in the Directory database. A Directory server can read through to the Directory database at any time, requesting account information that is new and that its cache does not yet contain. When such a read-through occurs, the Directory Cache server automatically updates its cache with the account information for the new record, without waiting for the regularly scheduled update interval to pass.

Message Delivery

The following servers work together to provide message delivery:

- Message Transport Agent (MTA)
- Queue server

The **Message Transport Agent (MTA)** receives incoming messages by:

- Listening on the SMTP port
- Accepting messages from clients
- Determining whether the recipients are in a local or remote domain

For recipients in the local domain, the MTA obtains account information from the Directory Cache server and then hands the message to the Message Store Server (MSS). For recipients in other domains, it sends the message to the remote location over the Internet.

The MTA enforces relay restrictions and redirection instructions, and manages any errors that occur during the delivery process. It also produces and sends bounce messages as required.

One MTA can run per host. It is possible to add more independent hosts as needed, distributing message traffic with a load-leveling mechanism such as round-robin DNS.

The **Queue server** temporarily stores (queues) messages that the MTA cannot deliver immediately. At regular intervals, the MTA requests stored messages from the Queue server and attempts to deliver them again. The Queue server is an optional server that enables the MTA to execute if a failure occurs.

Message Storage

The following servers work together to provide message storage:

- Message Store Server (MSS)
- Queue server

The **Message Store Server (MSS)** is responsible for storing messages in end users' mailboxes. It takes delivery of messages from the MTA, and handles requests for message retrieval from the POP and IMAP servers.

When a message is delivered to the MSS, the MSS must check that it does not exceed quotas associated with the destination mailbox. To accomplish this, the MSS communicates with the Directory Cache server to acquire class-of-service information for the account associated with each destination mailbox.

The MSS also maintains information about mailboxes, folders, message headers, and message structure in the Message Store database, and stores the messages themselves as individual files in the Message File system. Several MSS processes can run simultaneously on a single host.

The **Queue server** is responsible for storing any messages that the MTA cannot deliver immediately. For example, when a remote host is temporarily offline, the Queue server stores all messages destined for this host in a spool directory. The Queue server also spools local messages that are too large for the MTA to deliver immediately. These messages are then processed from the spool rather than completely in memory, since processing too large a message in memory might leave the network vulnerable to timeouts.

The Queue server is frequently installed on the same host as the MSS so that temporarily spooled messages are kept on the same machine as permanently stored messages. However, this is not a requirement. The Queue server is an optional component that simplifies operation of the system by avoiding the need for MTA data recovery.

Message Retrieval

The following servers work together to provide message retrieval:

- POP server
- IMAP server
- Web server

The **POP (Post Office Protocol) server** handles requests for message retrieval from any client that supports the POP3 protocol. It communicates with the Directory Cache server to validate each end user's login name, password, and class of service, and to obtain the name of the host on which the user's mailbox resides. The POP server also communicates with the MSS to handle requests for message retrieval on behalf of the client.

One POP server can run per host. If needed, several hosts can run POP servers, distributing the work with a load-leveling mechanism such as round-robin DNS.

The **IMAP (Internet Message Access Protocol) server** handles requests for message retrieval from mail clients that support the IMAP protocol. It communicates with the Directory Cache server to validate each end user's login name, password, and class of service, and to obtain the name of the host on which the user's mailbox resides. The IMAP server also communicates with the MSS to handle requests for message retrieval on behalf of the client.

The IMAP server allows most IMAP-enabled clients to send and receive messages in online, offline, and disconnected modes. Unlike POP users, whose messages are automatically downloaded to a local machine each time they connect, IMAP users have the option of working with their messages directly on the server. As a result, average IMAP connection times are likely to be longer than POP connection times.

One IMAP server can run per host. If needed, several hosts can run IMAP servers, distributing the work with a load-leveling mechanism such as round-robin DNS.

End users commonly compose e-mail messages from clients running on desktop machines, and transmit them using the SMTP protocol to the MTA. It is also possible for users to compose messages using a Web browser and to send them using the HTTP protocol to a Web server running the WebEdge client application. The Web server then packages the messages and sends them to the MTA for delivery.

The **Web server** allows end users to retrieve mail using a Web browser and WebEdge. When an end user attempts to retrieve mail, the Web server receives the request and executes a script that calls the underlying InterMail API. The API calls the Directory Cache server to retrieve directory information, and calls the MSS to access the end-user's mailbox. The message headers are displayed on the end user's screen.

System Management

The following servers work together to control and monitor the InterMail system functions:

- Manager server
- Configuration server
- SNMP server

A **Manager server** is installed on each InterMail host. It allows you to log on to an InterMail host and start or shut down any of the servers running on any other host in the InterMail system. When you issue a startup or shutdown command from one host, the instructions pass to the Manager server on each affected host for implementation.

The **Configuration server** holds the master Configuration database, which contains a complete collection of system settings and distributes the latest version of this master database to all the other hosts in the InterMail system. Each of these hosts also has its own Configuration database containing a complete set of system settings.

When you make changes to the master Configuration database and elect to propagate them, all the InterMail hosts check to see if the master Configuration database is different from the one they are currently using. If it is, they copy the latest version from the Configuration server.

The **SNMP (Simple Network Management Protocol) server** gathers useful system information from the other InterMail servers, including current server status, number of connections since the server started, and number of messages stored in the MSS. If your InterMail system includes a monitoring station (this is an optional third-party component), it can request this information from the SNMP server for real-time viewing and processing. InterMail's SNMP server can monitor all servers except the Manager and Configuration servers.

Information Storage

The InterMail servers are complemented by the following databases, which maintain information about InterMail end users, messages, and the servers themselves:

- The **Directory database** is a single Oracle relational database that is the definitive source of InterMail account information. This information includes each end user's domain, username, password, class of service, e-mail address, and delivery information. This is the database from which all Directory caches originate.
- A **Directory cache** is a local database that exists on each host running a Directory Cache server. End user account and domain information is partitioned among the Directory caches in order to improve efficiency by eliminating bottlenecks to the Directory database. This promotes quick local access to account information when large numbers of accounts are being serviced. The Directory server routes information requests to the correct Directory Cache server.
- The **Message Store database** stores dynamic data such as which messages have been read and deleted. Information about mailboxes, folders, message headers, and message status is also maintained here.
- The **Message File system** stores the actual body of a message, along with another copy of the header. Each message exists as a single file that includes all headers, body text, and attachments. Therefore, the text of a message destined for multiple recipients is stored only once, saving valuable system resources.
- The **Configuration database** stores a complete set of system settings that control the behavior of the InterMail servers and indicate the location of required system resources. Message routing, mail blocking, mailbox creation, message bouncing, and other system functions are controlled through configuration keys. A copy of the Configuration database exists on each InterMail host. Every InterMail process reads this database at startup.

  The master Configuration database is on the same host as the Configuration server. Whenever you make changes to the Configuration database, you should make them to the master and then propagate them throughout the system.

Supporting Technology

All components of the InterMail messaging system use the following:

- **Multi-threading**—All InterMail servers are multi-threaded to enhance performance. Multi-threading allows individual InterMail processes to handle simultaneous message delivery and retrieval requests, resulting in high message throughput.

  Multi-threading also allows a server to take advantage of multiple CPUs on a single machine, thus increasing performance. Moreover, the number of threads each process can use is configurable, allowing you to fine-tune your system for the most efficient use of resources.

- **RME**—InterMail servers communicate using a remote-procedure-call protocol called Remote Method Execution (RME). RME is object-oriented, so if a server asks the MSS to retrieve a message, it replies by sending an object representing the message.

  The RME protocol is transaction-based, guaranteeing that the results of all operations are known. In addition, RME supports versioning to facilitate smooth system upgrades, allowing different versions of the various InterMail servers to communicate seamlessly while system components are upgraded one by one.

- **LDAP**—LDAP (Lightweight Directory Application Protocol) is the emerging Internet standard for remote access and integration of diverse directory services. The InterMail directory supports the LDAP V3 protocol and adopts many LDAP features that are critical for integration with other applications, including an LDAP-based data model, schema access and extension capability, replication, partitioning, and referrals.

- **Event logging and statistics**—All InterMail server processes have a common logging mechanism. This mechanism can produce standard logs that record events as they occur, and statistics logs that indicate activity over time. You can use these logs to monitor the overall performance of the system for capacity planning and fine-tuning.

- **Administration commands**—The administration utilities are command-line programs for observing and changing the behavior of the InterMail system. The execution of many commands is location-independent, and can monitor and control any component of the system. Commands exist for account, mailbox, server, and log management, as well as for system diagnostics.

2

Getting Started

This chapter assumes that the InterMail software has been installed and outlines tasks that must be performed before the system goes into production mode. It also discusses some of the basic operations associated with administering InterMail. Anyone who administers InterMail in any capacity should be familiar with the topics covered in this chapter, which include:

- Using InterMail administrative commands
- Starting up and shutting down servers
- Planning tasks that must be completed before the system goes into production mode

Using InterMail Administrative Commands

The InterMail system supports a command line interface for performing various administrative tasks, such as:

- Starting up and shutting down InterMail servers
- Creating, modifying, and retrieving account information
- Analyzing and fixing corrupted messages
- Modifying message stores (mailboxes)
- Reporting statistics on server usage

On the host where you install InterMail, a new UNIX user and group are created before installation. This user account is the InterMail user, and it is the only member of the new InterMail group. It is in this user's home directory that you install InterMail files, and it is as this user that you run all InterMail processes. To execute any administrative commands, or to administer InterMail in any way, you must log in as the InterMail user.

The utilities used to administer InterMail from the UNIX command line are in the `bin` and `lib` directories in the InterMail user's home directory. The installation process

adds the `bin` directory to the user's path, so that you can execute all InterMail commands in the `bin` directory from any directory. For example, to get a report on the InterMail servers currently running, you need only log in as the InterMail user and execute the appropriate command:

```
UNIX(r) System V Release 4.0 (paris)

login: imail
Password:

paris% imservdisplay
...
```

Throughout this manual, examples appear for executing specific InterMail commands. Before executing these commands, you must be logged in as the InterMail user.

Starting Up and Shutting Down Servers

Once you have logged in as the InterMail user and are ready to start administering the system, your first task is to start up the InterMail servers. This section explains the basics of starting up, shutting down, and restarting InterMail servers.

Note: Following installation, servers that manage the ISD and the configuration of the system are already running.

To perform an operation on a server, you must specify the server by its process name. InterMail servers and their process names are:

Server Name	Server Process Name
Message Transport Agent (MTA)	`mta`
Message Store Server (MSS)	`mss`
POP server	`popserv`
IMAP server	`imapserv`
Queue server	`imqueueserv`
Manager server	`immgrserv`
Configuration server	`imconfserv`
Directory server	`imdirserv`
Directory Cache server	`imdircacheserv`
SNMP server	`snmpd`

There are two commands that you can use to start up and shut down InterMail servers:

- `imservctrl` starts up and shuts down servers only on the local host and must be run from the `lib` directory.
- `imctrl` starts up and shuts down servers on both local and remote hosts. This command is very useful in large multi-host InterMail installations. The `imctrl` command requires that the Manager server be running.

Note: Because the `imctrl` command requires that the Manager server be running, you must start the Manager server using the `imservctrl` command before you can use the `imctrl` command for operations on the other servers.

---

imservctrl

The `imservctrl` command starts up and shuts down InterMail servers. Depending on the parameters you included, `imservctrl` can either start, restart, or shut down a single server, a list of servers, or all servers on the local host.

Note: For the `imservctrl` command to have an effect on a server, the `<host>/sysadmin/<server>_run` configuration key for that server must have a value of `on`. You can read this key either by running `imconfedit` and searching for this key with your editor, or by running `imconfget -h <hostname> -m sysadmin <serv>_run`.

Example

To stop all servers, run `imservctrl` as in the following example:

Note: When you run `imservctrl`, the full path (`lib/imservctrl`) is necessary.

```
paris% lib/imservctrl stop
imservctrl: stopping imapserv (25421)
imservctrl: stopping popserv (25584)
imservctrl: stopping mta (25481)
imservctrl: stopping mss.1 (25267)
imservctrl: stopping imdirserv (21295)
imservctrl: stopping immgrserv (21282)
imservctrl: stopping imconfserv (25351)
imservctrl: cleaning /vol1/imail/tmp ...
imservctrl: done
paris%
```

The server list is optional; when no list is specified, `imservctrl` acts on all the servers configured to run on the local machine. To start a particular server and no others, you must specify this particular server as a parameter to `imservctrl`. For example, to start the Manager server only, run `imservctrl` as follows:

```
lib/imservctrl start immgrserv
```

Note: The server operations later in this section are described using the `imctrl` command. For more information on the `imservctrl` command, see the *InterMail Mx Reference Guide*.

imctrl

The `imctrl` administration command starts up and shuts down InterMail servers on a local or remote host, allowing administrators to remotely control the operation of InterMail. When executed, `imctrl` determines the hosts affected by the specified commands, and then passes these commands to the Manager Server of each affected host. The Manager server then executes the appropriate startup or shutdown operations for the affected servers.

The following sections demonstrate using `imctrl` to start up, shut down, and restart InterMail servers.

Starting Up

To start up an InterMail server with `imctrl`, use the `start` option:

```
imctrl <host> start <server>
```

For example, to start all servers on the host `paris`, enter:

```
imctrl paris start allservers
```

To start one or more specific servers, include the server types. For example, to start the IMAP and POP servers on the host `paris`, enter:

```
imctrl paris start popserv:imapserv
```

Shutting Down

There are four ways to shut down InterMail servers:

- **Stopping:** Stopping a server causes the process to exit as soon as possible in an orderly fashion. The server shutdown interrupts client sessions, but any disconnected clients receive meaningful error or status messages.
- **Draining:** Draining a server causes the process to refuse to accept any new client connections, but does not interrupt connections that are still in process. When all transactions finish, the server then exits. For example, when a POP server drains, it does not accept any new connections from POP3 clients. The server waits for all existing client connections to close, and when no more clients are present, closes its connection to the MSS and exits. This method of shutdown is particularly useful for the POP server and MTA because the effects on these servers are the most visible to end users.

Note: Because client connections to the IMAP server are typically long-lived, the drain method of shut down is typically not practical for the IMAP server.

- **Killing:** Killing a server causes the process to exit at once. This method is equivalent to using the UNIX `kill -9` command. It causes the server process to exit immediately, with no error or notification messages to connected clients. It is recommended that you not use this method of shutdown unless absolutely necessary.
- **Exiting:** Exiting a server is similar to killing a server in that it forces the server to shut down immediately, closing all client connections. The difference between the `exit` and `kill` commands is that the `exit` command is responsive only if the process is healthy.

Stopping

To shut down a server by stopping with `imctrl`, use the `stop` option:

```
imctrl <host> stop <server>
```

For example, to stop all the servers on host `paris`, enter:

```
imctrl paris stop allservers
```

To stop a particular server, in this case `imapserv`, enter:

```
imctrl paris stop imapserv
```

Draining

To shut down a server by draining it with `imctrl`, use the `drain` option:

```
imctrl <host> drain <server>
```

For example, to drain the POP server on host `paris`, enter:

```
imctrl paris drain popserv
```

When this command is executed, the POP server on `paris` continues to service connected clients, but does not accept any new connections. Because there is no interruption to current connections, users connected to this POP server when the command is executed will be unaware of the imminent shutdown of the server.

Killing

To shut down a server by killing it with `imctrl`, use the `kill` option:

```
imctrl <host> kill <server>
```

For example, to kill the MTA on host `paris`, enter:

```
imctrl paris kill mta
```

Exiting

To shut down a server by exiting with `imctrl`, use the `exit` option:

```
imctrl <host> exit <server>
```

For example, to exit all the servers on host `paris`, enter:

```
imctrl paris exit allservers
```

To exit a particular server, include the server type:

```
imctrl paris exit imapserv
```

Restarting

Several configuration changes require the shutdown and restart of InterMail servers before the changes can take effect. To restart an InterMail server that is currently running, you can execute two separate `imctrl` operations: one to shut down the server, and a second to start it. Alternatively, you can use a restart option that allows you to accomplish both of these in a single `imctrl` execution.

The available `imctrl` restart options are `restart`, `drainStart`, `stopStart`, `exitStart`, and `killStart`. These operations are identical to the associated shutdown options described in the previous section, but cause the specified servers to restart immediately after shutdown.

Note: The `imctrl` parameter `restart` is identical to the `stopStart` option. In addition, the `exitStart` option has the same effect as the `killStart` option.

For example, to drain and then restart the POP server on host `paris`, enter:

```
imctrl paris drainStart popserv
```

To kill and restart the MTA on the host `paris`, enter:

```
imctrl paris killStart mta
```

Checking Server Status

Once the required servers have been started or stopped, you can check server status using `imservdisplay`:

```
paris% imservdisplay
Monitoring InterMail modules: httpd imapserv imcfgdbserv imconfserv
imdirserv
immgrserv mss mta popserv snmpdm.
 . . .
mta Report:
-----------
        Note: ProcFound: mta process Found as PID: 13847.
        Note: ServerPing: mta responded to version query
        Note: PortBanner: Port Banner found on port 25
        Note: PortQuitSucc: Port Quit successful on port 25
. . . .
```


After displaying the configuration and running status of each server, `imservdisplay` reports any log events. A review of these log events will quickly show you if serious problems exist in a server:

```
. . . .

        /imail/imail/log/mta.log, Severity: Erro {
                7: SmtpClientMailLoopDetected
        /imail/imail/log/mta.log, Severity: Note {
                14: AcctInvalidUser
                35: MsgTrace
                1: ProcReExecingProg
                1: ProcSetuidSucceed
                9: SmtpConnectionClosed
                9: SmtpConnectionReceived
        }

. . . .
```

Preproduction Planning

After you install InterMail, there are a number of additional tasks you must complete before considering your site ready to operate in full production mode, serving customers 24 hours a day, 7 days a week.

This section discusses a variety of preproduction tasks. The tasks fall into the following categories:

- Basic integration tasks. These tasks are broad in nature and involve your operation as a whole, not just your messaging system.
- Defining the structure of your site. These tasks are necessary before you can add account and domain information to the system.
- Tasks that pertain to InterMail directly. Completion of these tasks results in a custom configuration specifically suited to the needs of your organization.
- Methods to check some of the server operations before going online.

Integration with Existing Systems

When integrating InterMail into your existing environment, you must give thought to

- Establishing a connection to the existing provisioning system
- Creating a log-monitoring program
- Integrating SNMP with a monitoring station

Because these tasks are necessarily site-specific, they are described here in only general terms.

Creating a Log-Monitoring Program

The InterMail log files enable you to determine system usage, message flow, the number of connections to and from servers, and other pieces of vital system information. Each InterMail server generates its log file, and log rollover periods are configurable.

Log files are a comprehensive collection of data, in a format that allows flexible reporting. Although you can read InterMail logs from the directories in which they reside, you may want to consider creating a custom log-monitoring program that allows you to monitor your logs and receive event notifications in whatever way is most convenient.

For an overview of InterMail logging, see Chapter 10. For a complete listing of InterMail events, see the *InterMail Mx Reference Guide*.

Establishing a Connection to Your Provisioning System

To link InterMail accounts with related billing information, you must connect the content of the Integrated Services Directory (the central repository for InterMail account information) with your existing provisioning system.

Integrating SNMP with a Monitoring Station

SNMP (Simple Network Management Protocol) is a protocol that enables you to monitor useful network and system traffic statistics. In an InterMail system, SNMP provides information such as the number of connections to the POP server since the server was started, the total number of messages that are stored on the Message Transport Agent (MTA), and the total number of messages stored in the Message Store Server (MSS). This information is “sampled” and sent to output on the user-defined SNMP monitoring station.

The process of configuring SNMP to run on a monitoring station is explained in Chapter 10 in this manual.

Defining Backup and Recovery Policies

Establishing a clearly defined and well-tested backup and recovery program is very important. There are many considerations involved in backing up your InterMail system, including:

- Which items require backup?
- How often should backups occur?
- What strategies should you employ to make most efficient use of your backup and recovery resources?

Answers to these questions will depend on factors that are unique to each InterMail system such as available hardware and message traffic. You can minimize any potential disaster by establishing a suitable backup and recovery policy, modifying it as needed, and making certain to enforce it.

Chapter 11 discusses recommended backup and recovery procedures; however, you should view these instructions only as a starting point for developing your own site-specific instructions. They are not a substitute for backup procedures you already have in place, or for others that you need to develop.

Once you have fully installed and configured InterMail, make a complete backup of your customized system. This will provide you with a base that contains a clean installation along with all your initial configuration settings. For a description of all configuration settings, see the *InterMail Mx Reference Guide*.

Managing Site Structure

When an InterMail site is configured, domains, classes of service, and an initial set of accounts are created.

Establishing Domains

In addition to specifying the default domain that is set up during the installation process, InterMail allows you to specify other domains over which your server can claim partial or complete authority. If you want to provide mail service to users whose e-mail addresses are not within the default domain, you must identify those other domains to the system before mail processing begins. For example, if the default domain is set as `software.com` but you also expect to handle at least some mail for users in a domain called `accordance.com`, you must define the `accordance.com` domain in the Integrated Services Directory (ISD).

If you do not identify a mail domain, InterMail will be unable to deliver messages to users in that domain, and you will be unable to create accounts with addresses in that domain.

For a discussion of domains, see Chapter 3.

Creating Accounts

Like domain information, account information is in the ISD. You can use standard InterMail administrative commands to create accounts, but you will probably want to use a provisioning system of some kind to simplify this task. There are three methods for a provisioning system to interface with InterMail: C API, Perl API, or the command line interface.

Creating Class-of-Service Options

A *class of service* defines a common set of InterMail services that are available to individual users. Classes of service can be used to bundle feature sets into distinct packages that define certain attributes of an account, such as IMAP access or mailbox quotas. Classes of service are extremely useful from a marketing standpoint. A service provider may charge customers one monthly rate for e-mail accounts with a limited range of services (for example, “class of service A”), and a different rate for e-mail accounts offering a larger set of services (for example, “class of service B”).

The number of classes of service defined is at the discretion of the service provider, but is typically limited in scope. To determine the appropriate number of classes of

service and the characteristics associated with each, the requirements of all expected clients should be reviewed.

Migrating Existing Accounts

If you have a database of account information from a legacy mail system, one of your most important pre-production tasks will be to “migrate” existing account information to the Integrated Services Directory. The new InterMail system may be activated once the account data is migrated.

Migration also involves moving messages from your legacy mail server to mailboxes in the InterMail messaging system. This step consolidates all message and account data within InterMail, eliminating any reliance on the legacy mail system for access to legacy messages.

You can perform migration tasks after installing InterMail, but before bringing the system online. For a complete discussion of migration tasks, see the *InterMail Mx Migration Guide*.

Modifying Configuration Options

When you installed InterMail, each available configuration option received a value. You are probably unaware of the settings of most values, as very few of them required direct selection on your part. Instead, most options received their initial or default values (the “starter” entry for a particular configuration key when installing InterMail). This method simplifies the installation process, but does not necessarily result in the ideal configuration for your site.

The sections that follow identify a variety of configuration options that you should consider before bringing your site into full production mode. By examining and editing the values associated with each option, you will be able to implement the routing, storage, and security policies specific to your site.

For instructions on editing configuration keys, see Chapter 4. For details on specific configuration keys, see the *InterMail Mx Reference Guide*.

Determining a Security Policy

InterMail provides a number of security-related features you should thoroughly understand before going into production. It is not possible to offer a definitive set of rules on how to define your security settings since this would depend chiefly on your particular environment and security needs. However, Chapter 5 of this guide explains each security option in detail, and discusses the potential impact of different security settings on your mail service.

Once you become familiar with InterMail’s security features, you can determine how best to apply them to meet your particular needs.

Note: It is not advisable to put up a mail server on the open Internet that allows open mail relaying. This could cause senders of junk e-mail to use your server for relaying. Consider the InterMail anti-relay options described in Chapter 5 in this manual before determining and implementing your security policy.

Setting Mail Routing Rules

Mail routing rules allow you to proxy mail from an InterMail host to some other mail system. This is typically useful during the migration process, when transferring accounts from a legacy mail system to InterMail. For more information, see the *InterMail Mx Migration Guide*.

There are also a number of other mail routing policies that you can set, including the use of non-authoritative and rewrite domains, and header rewriting. You can add to these rules or change them at any time after going into production, but it may be useful to establish an initial set of rules in pre-production. For a full discussion of routing rules, see Chapter 6.

Establishing Message Quotas

Message quotas allow you to limit the amount of disk space that a mailbox can take up, as well as the maximum size of messages to accept.

There are three types of message quotas:

- System-wide quotas on mailbox size
- System-wide limits on message size
- Per-account quotas on mailbox size

Setting per-account quotas is an ongoing administrative task, rather than a pre-production issue. On the other hand, the system-wide settings are important considerations for pre-production, since the choices you make will affect your entire system right from the start. For more information about setting message quotas, see Chapter 8.

Setting Queuing Policy

When InterMail cannot deliver a message immediately, it queues it for future delivery. InterMail attempts delivery of queued mail at configurable intervals. In addition, you can also decide how long deferred mail can remain in the queue before considering it undeliverable and returning it to sender.

Before going into production, you will need to decide on a policy for how to handle queued mail. This is important because deferred mail that remains on your system for too long can also consume large amounts of disk space. In addition, too-frequent delivery attempts may compete for system resources that other processes require. On the other hand, if you bounce deferred mail too quickly, or allow too much time between delivery attempts, you may diminish the quality of service to your users.

InterMail's default settings for queued mail are usually adequate. However, you may want to change these settings to meet the specific needs of your site. For a further discussion of mail queuing, as well as procedures for changing the default settings, see Chapter 7.

Setting up the Welcome Message

If specified, a welcome message can arrive automatically in every new mailbox. In order to enable this option, before going into production, you must create the welcome message for new users. A well-written welcome message can create a positive first impression and also convey important information such as contact numbers and/or e-mail addresses for technical support, information on quota policies, and the URL for your organization's Web site.

To create a new `admin` mailbox with a welcome message, perform the following steps:

1. Create an account for the `admin` mailbox.

   This is required to run the `immsinit` administration command that creates the `admin` mailbox. The `internalId` specified when the account is created will be the `admin` mailbox name. Note that the `internalId` field is numeric only.

2. Write a welcome message in a text file. The message should include the following information:

   ```
   From: <Your ISP>
   To: <Your ISP's new customers>
   Subject: <Welcome to Your ISP>
   Message-ID: <unique id, such as "<Welcome.052699>">
   The body of the message
   ```

   Make sure that the `Message-ID` header is unique and is in angle brackets. Adding a date stamp can be useful.

3. Set the config key `/*/mss/welcomeMsgID` key to the value of the `Message-ID` header in the message.

   Make sure that you include angle brackets if they are present. For example:

   ```
   /*/mss/welcomeMsgID: [<Welcome.052699>]
   ```

   The `welcomeMsgID` key refers to the `messageid` field in the `im_message` table. If the `Message-ID` is not unique, this is the value that must be changed.

4. Set the config key `/*/mss/adminMessageStoreName` to the `internalId` specified for the account created in step 1.

5. Run the `immsinit` command:

   ```
   immsinit -host <hostname> -w <welcome message filename> -a <value /
   */mss/adminMessageStoreName key>.
   ```

This creates the `admin` mailbox and inserts the defined welcome message into it. If you do not use the `-w` flag, the default welcome message is inserted. The default welcome message has a `Message-ID` of `<Welcome>`. If you do not use the `-a` flag, `immsinit` creates the `admin` mailbox by default (you would need an existing account

with an `internalId` of `admin` for this to work). You cannot run `immsinit` if the `admin` mailbox specified already exists. The command will produce an `MsAlreadyExists` error.

Changing the Welcome Message

You can change the welcome message in two ways: by sending the new welcome message to the account that points to the admin mailbox, or by deleting the `admin` mailbox and running `immsinit` again with the new welcome message.

To send the new welcome message to the `admin` mailbox:

1. Using any mail client or by telnetting to the MTA, send the new welcome message to the account that points to the `admin` mailbox.

   Be sure that the headers described above are included in the welcome message. Ensure that the `Message-ID` header is unique. Otherwise, if there is already a message in the database with the same message ID, the MSS will change the message ID of the new message.

2. Verify that the message gets delivered properly and the message ID has not been changed.

   You can make sure it is the correct message by checking the arrival date or reviewing the message file on disk.

3. Change the `/*/mss/welcomeMsgID` key to the value of the `Message-ID` header of the new message.

To delete the `admin` mailbox:

1. Run the `imboxdelete` command.

   ```
   imboxdelete <hostname> <admin mailbox name>
   ```

2. Run `immsinit` again as described above making sure to use a unique message ID in the new welcome message.

3. Set the `welcomeMsgID` and `adminMessageStoreName` keys appropriately.

Set Up cron Jobs

Certain InterMail processes should run periodically to perform routine system monitoring or maintenance tasks. Most recommended `cron` jobs begin automatically at time of installation. However, you may wish to review the schedules for those utilities, and supplement them with additional `cron` jobs that perform tasks such as the archiving of mail-related log files.

3

Site Management

When you configure an InterMail site, you create domains, classes of service, and an initial set of accounts. You can perform these tasks using the InterManager graphical user interface (GUI), InterMail's command-line utilities, or the InterMail C API.

This chapter describes how to use the InterMail command-line utilities to set up and manage:

- Domains
- Classes of service
- Accounts

Domains

Domains are a general form of Internet addressing. All SMTP-compliant e-mail addresses include a domain to the right of the @ sign. The domains defined in the Integrated Services Directory (ISD) determine the mail domains known to your system. All InterMail accounts and any addresses in the system must be part of a known domain.

There are four types of domains:

- Local domains
- Non-authoritative domains
- Rewrite domains
- Deleted domains

Note: InterMail accounts and addresses can be associated only with local or non-authoritative domains.

Local Domains

A local domain is a domain for which InterMail claims exclusive control. If a domain (or subdomain) is defined as a local domain for your site and mail is received for an address within that domain, InterMail considers itself the ultimate destination for that mail.

Messages that are addressed to non-existent recipients within a local mail domain are considered undeliverable, and are not routed to any other host or site; typically, such messages are returned to the sender.

Non-Authoritative Domains

A non-authoritative domain (also known as a semi-local domain) is similar to a local domain, but does not define the InterMail system as the definitive destination for all mail addressed to that domain. Non-authoritative domains allow InterMail to accept mail for a domain, but relay it to another mail host, if necessary.

To facilitate required relay, all non-authoritative domains are associated with the name of a relay mail host. When mail is received for an address in a non-authoritative domain, and the recipient's address cannot be located among the accounts in the ISD, the message is relayed to the host associated with the non-authoritative domain.

Non-authoritative domains can be established to define domains for which your site is an MX backup, or used when InterMail is run in parallel with an existing mail system (such as during migration of e-mail accounts from a legacy mail system to InterMail).

Rewrite Domains

Rewrite domains are very different from local and non-authoritative domains. A rewrite domain simply defines a rule for rewriting the domain portion of the recipient address of incoming mail, and must be associated with the name of a local or non-authoritative domain.

When an incoming message is received, and the domain of the recipient address is defined as a rewrite domain, the domain portion of the address is rewritten with the local or non-authoritative domain associated with the rewrite domain. This feature allows InterMail accounts to receive messages addressed to the same user in multiple domains without requiring explicit alias addresses for each account.

Note: For more information on routing options available with rewrite domains, see Chapter 6.

Deleted Domains

The InterMail system maintains a logical record of deleted domains. These domains may appear in database records and are used to track previous modifications.

Creating Domains with imldapsh

This section describes how to create domains using the `imldapsh` administrative command.

Creating a Local Domain

To create a domain with the `imldapsh` command, use the `CreateDomain` option, abbreviated `cd`. Unless otherwise specified, `imldapsh` creates all domains as local domains. The syntax for creating a local domain is:

```
imldapsh cd <DomainName>
```

Where:

`DomainName`	Is the name of the local domain.

For example, to create a local domain for `example.com`, enter:

```
imldapsh cd example.com
```

Creating a Non-Authoritative Domain

The syntax for creating a non-authoritative domain is:

```
imldapsh cd <DomainName> nonauth <relayHost>
```

Where:

`DomainName`	Is the name of the non-authoritative domain
`relayHost`	Is the host (or fully qualified domain name) to which the system routes mail addressed to unknown users at the non-authoritative domain.

For example, to create a non-authoritative domain for `minorcorp.com`, enter:

```
imldapsh cd minorcorp.com nonauth pluto.minorcorp.com
```

In most instances, you should use a fully qualified domain name as the relay host. The only exception would be if the relay host were a “pingable” host on your local network, in which case just the host name (`pluto`) would be sufficient for the `relayHost` argument:

```
imldapsh cd minorcorp.com nonauth pluto
```

Note: Because domain identification plays a critical role in mail delivery, it is extremely important to identify domains appropriately. You should never claim a domain as local unless you are the sole provider of mail service for that domain. Conversely, you should not assert partial authority (by specifying a non-authoritative domain) if you are, in fact, entirely responsible for mail addressed to that domain.

Creating a Rewrite Domain

The syntax for creating a rewrite domain is:

```
imldapsh cd <DomainName> rewrite <RewriteValue>
```

Where:

`DomainName`	Is the name of the rewrite domain (the name to be replaced).
`RewriteValue`	Is the domain that replaces the `DomainName` value.

For example, to create a rewrite domain whose rule rewrites `minorcorp.com` to `example.com`, enter:

```
imldapsh cd minorcorp.com rewrite example.com
```

When mail arrives addressed to `<anyone>@minorcorp.com`, the system rewrites the domain portion of the recipient address to `example.com`. InterMail then attempts delivery to the rewritten address.

Classes of Service

A class of service defines a common set of InterMail account attributes that are available to end users. Each e-mail account in InterMail has a particular class of service that acts as a template for those accounts. It is possible to define most account attributes at both the account and class-of-service levels.

Classes of service can bundle features in distinct packages for consumers. For example, a service provider may charge customers one monthly rate for e-mail accounts with a limited range of services (such as simple POP3 access and low mailbox quotas) and a different rate for e-mail accounts with a larger set of allowed services (such as IMAP access, higher mailbox quotas, and security features). Classes of service allow you to manage these sets of features easily for groups of end users.

Each class of service defines a set of service options that correspond to definable account characteristics. These service options, called class-of-service attributes, specify the individual InterMail features that end users may use, and control the ability of end users to set these features themselves. There is no minimum or maximum limit to the number of attributes included in a class of service.

Sample Classes of Service

Sample classes of service illustrates three sample classes of service that a service provider might use to differentiate specific service options. Each of these classes of service has a name, a series of associated service options, and a price to end users.

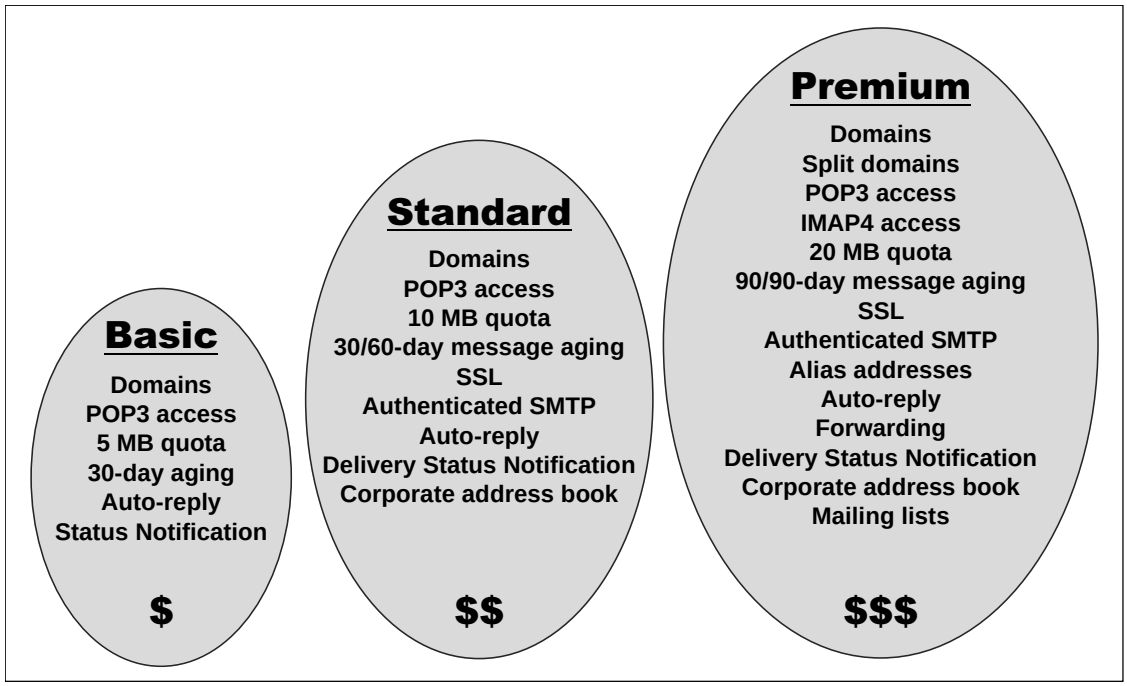

Figure 2 Sample classes of service

In this example, the service provider's customers have a choice of three account levels when purchasing e-mail service:

- The **Basic** class of service defines e-mail accounts that have only a basic set of service options, such as POP3 access, auto-reply, and delivery status notification. The mailbox of a Basic account only gets 5 MB of disk space.
- The **Standard** class of service defines e-mail accounts that include the features of Basic accounts, but with added service options that provide access to security features (such as SSL and authenticated SMTP), as well as corporate address book data. The message aging service option of the Standard class of service allows messages to remain on the server for a longer period, and the mailbox quota of 10 MB is twice that of the Basic mailbox quota.
- The **Premium** class of service defines e-mail accounts that include all of the service options of Standard accounts, but with additional service options such as IMAP4 mailbox access, alias addresses, forwarding delivery, and access to mailing lists. The message aging policy and mailbox quota of a Premium account are also more than the values for the Standard class of service.

About Creating Classes of Service

During installation, an initial class of service called `default` is created. The `default` class of service contains all the class-of-service attributes and is used for account creation if no other class of service is specified.

After creating initial domains, you are ready to create classes of service. By creating several classes of service in a tiered system, you can place users according to their account needs. For example, you may wish to have three classes of service, each designed to suit a particular user:

- A user who reads mail infrequently and does not require forwarding.
- A user who frequently reads mail; requires the ability to forward mail, send auto-reply messages, and requires more quota allocation than the first class of service.
- A user who requires, in addition to the features of the Standard class, requires IMAP access, and the ability to filter messages that appear to be junk mail.

In order to set up a class of service containing certain attributes, you must set the corresponding configuration keys. For example:

- To create a class of service that includes mail blocking as an option, set the `blockPerAccount` configuration key to `true`. If the key is set to `false`, all users have access to mail blocking, regardless of their class of service.
- To set an absolute maximum message size, set the value of the `maxMessageSizeInKb` configuration key should exceed the value set for any class of service. The `maxMessageSizeInKb` configuration key takes precedence over the maximum message size for any class of service. Therefore, any class of service limit that exceeds this value is ignored.

Note: For complete information on these configuration keys, see the *InterMail Mx Reference Guide*. For instructions on changing configuration key settings, see Chapter 4.

Creating Classes of Service with imldapsh

You can use the `imldapsh CreateCos` command to create classes of service at various locations in the Directory Information Tree (DIT). In preparation for creating a class of service, you must define the variable `imdbcAdminRoot` to specify the location in the DIT where administrative policies, which represent classes of service, can be found and created. For more information on the DIT, see the *Integrated Services Directory User Guide*.

To define the variable `imdbcAdminRoot`, use the `setenv` command as follows:

```
setenv IMDBCADMINROOT “<DN>”
```

Where:

`“<DN>”`	Is the distinguished name specifying the location in the DIT where you want to create the COS. ***Note:*** *Use quotation marks around the DN.*

For example, to create a class of service in the DIT location “`cn=site, cn=admin root`”, enter:

```
setenv IMDBCADMINROOT “cn=site, cn=admin root”
```

You are now ready to create a class of service by using the `imldapsh` command with the `CreateCos` option, abbreviated `cc`:

```
imldapsh cc <classOfService>
```

Where:

`classOfService`	Is the name of the new class of service.

For example, to create a class of service called Premium, enter:

```
imldapsh cc Premium
```

You can display a list of all existing classes of service using the `ListCosNames` option, abbreviated `lcn`:

```
imldapsh lcn
```

To display all the attributes associated with a class of service, as well as their values, use the `ShowCos` option, abbreviated `sc`:

```
imldapsh sc <classOfService>
```

For more information on the `imldapsh` command, see the *Integrated Services Directory User Guide.*

Accounts

After configuring the site, you must perform various account tasks such as adding new accounts, modifying existing accounts, and listing account information. Account information stored in the ISD has a significant effect on mail handling, given that entries in the account record control message delivery, message storage, and message retrieval.

This section describes:

- Account attributes
- Allocations

- Special-purpose accounts
- Account provisioning rules
- Basic account tasks, including creating, modifying, deleting, and listing accounts
- Account management tasks, including setting auto-reply, working with class-of-service attributes, and advanced management tasks

Account Attributes

This section discusses attributes that you must define for all accounts, including:

- General account data
- E-mail addresses
- Delivery information
- Auto-reply options
- Mailbox quota and related options
- Mail system access

General Account Data

The basic information for each InterMail account includes the name of the user, the domain associated with the account, the user's password, the type of account, and the status of the account. All accounts include information for these attributes.

- **Username**—The username of an account typically reflects the name of an individual user who will use the account. This name is the part of the account's e-mail address that precedes the `@` symbol. When the username and domain of an account are combined, they form the primary e-mail address of the account.

  For example, an account whose primary address is `susie.queue@software.com` has the username `susie.queue`.

  Usernames can be up to 64 characters in length. They can include all alphanumeric characters, as well as the underscore (_), period (.), and hyphen (-) characters. Usernames are not case-sensitive, so the usernames `John.Doe`, `john.doe`, and `JOHN.DOE` are identical. Because two accounts cannot share the same e-mail address, usernames must be unique within a domain.

- **Domain**—All InterMail accounts are associated with a specific domain. The combination of the username and the domain defines the primary e-mail address for an account.

  Note: An account can be associated with more than one domain through the use of alias addresses.

- **Password**—E-mail clients use the account password when connecting to the POP or IMAP server to retrieve mail from the account's mailbox. The password is also

required for mail delivery when secure or authenticated SMTP transmission is requested.

- **Type—**There are two types of InterMail accounts:
  - Standard
  - Administrative

  The type of account has no effect on InterMail behavior, so an administrative account has no more (or less) access to InterMail services than a standard account.

  The account type can be used in conjunction with a billing or provisioning system to indicate special handling. For example, you may create a rule in your billing system to ignore all administrative accounts when generating usage and billing information.
- **Status—**Each account has an associated status that defines its current state. There are six possible values for account status:
  - **Active** status indicates that the account is in a normal state. An active account can send and receive messages normally. This is the most common account status.
  - **Maintenance** status causes the system to queue incoming messages, and deliver them normally when the status of the account returns to Active. Requests to download mail are rejected with a message indicating that the account is in Maintenance mode and is temporarily unavailable.
  - **Suspended** status prevents access to the account's mailbox, and is typically set when payment for an account is overdue. The system returns any new mail to its sender, and rejects user requests to download mail by returning an unknown username/password error. Setting the status to Active restores normal delivery and client access to a suspended account.
  - **Locked** status is identical to Suspended status; it prevents user access to the mailbox and returns all mail sent to the account. The system supports this status type for backward compatibility only.
  - **Deleted** status completely halts mail activity for the account. The system treats mail addressed to the account as undeliverable, and rejects client requests to access the account's mailbox by returning an unknown username/password error. Unlike permanently deleting an account, setting an account's status to Deleted allows you to restore the account later by resetting its status.
  - **Proxy** status indicates that another mail system is handling mail activity for the account. The main use of this status is during migration of accounts from an existing mail system to InterMail. The system redirects incoming mail for an account in Proxy mode to the proxy MTA defined for that account. Similarly, the system redirects client requests to retrieve messages with the POP or IMAP server to the account's proxy POP or IMAP server. This redirection is transparent, so end users can specify an InterMail host as their

SMTP, and POP or IMAP server while their accounts remain on a legacy mail system.

When an account is in Proxy mode, the purpose of two account attributes is redefined. The entries for MSS host and auto-reply host define the proxy SMTP and POP servers, respectively.

Note: SMTP authentication does not work with an account is in Proxy mode.

E-Mail Addresses

All accounts have at least one e-mail address, known as the primary address. Accounts may also have additional addresses, or SMTP aliases.

- **Primary address—**The primary address is the "official" Internet e-mail address of an account. Although SMTP alias addresses are equally valid for sending mail to an account, the primary address is the only address that can be used to identify the account in operations related to the Message Store database and the ISD.

  The combination of the account's username and its domain defines its primary e-mail address. For example, an account with the username `jane.doe` and the domain `software.com` has a primary address of `jane.doe@software.com`.

- **SMTP aliases—**SMTP aliases are additional e-mail addresses for an account. The system delivers mail addressed to an SMTP alias to the account exactly as if the mail were addressed to the account's primary address. Like primary addresses, SMTP aliases must be unique throughout the system.

  Alias addresses allow individual accounts to receive mail at multiple domains, and to use multiple addressing formats. For example, an account with the primary address `john.doe@software.com` might have the following SMTP aliases:

  ```
  john@software.com
  jdoe@software.com
  ```

- **Alias limit—**Each account includes an optional limit on the maximum number of SMTP aliases. Although users may add their own alias addresses, you can control this limit.

Delivery Information

Each account includes a series of attributes that determine the account's method of mail delivery. The available types of mail delivery are local and forwarding. Accounts may use one or both of these delivery methods. Any account that uses local delivery has an associated mailbox.

- **Mailbox—**InterMail accounts typically have an associated mailbox, a storage area for mail sent to the account. The system delivers messages sent to the account to this mailbox, and the POP and IMAP servers retrieve mail from this mailbox. Mailboxes are discussed in greater detail in Chapter 8.

> ***Note:*** An account needs a mailbox only if it uses the local delivery method. Accounts that use forwarding delivery only do not require or make use of a mailbox.

- **Local delivery—**The most common method of mail handling is local delivery. When local delivery is enabled, messages are stored in the account's local mailbox. Messages can be retrieved from the mailbox with any of InterMail's standard retrieval mechanisms.
- **Forwarding—**The forwarding method of delivery is similar to the forwarding of postal mail. When mail arrives for an account that uses forwarding delivery, InterMail modifies the message's destination address and then forwards it to the new location.

  Accounts that use forwarding delivery must also have one or more associated forwarding addresses.
- **Forwarding limit—**Each account includes an optional limit on the maximum number of forwarding addresses. Although users may add their own forwarding addresses, you control this limit.
- **Mail filtering—**Each account includes an optional control on mail filtering. If filtering is enabled, mail addressed to this account is subject to system-wide filtering rules. For a description of mail filtering, see Chapter 5.

Auto-Reply Options

InterMail accounts include the ability to automatically respond to incoming mail. When mail is received by an account that has the auto-reply feature enabled, InterMail immediately sends the account's auto-reply message to the sender of the original message. The auto-reply feature can operate in one of three modes:

- **Reply mode** sends the account's auto-reply message every time the account receives mail. One common use of Reply mode is to distribute information on sales, system policies, directions, and so forth.
- **Echo mode** is the same as Reply mode, but it includes the sender's original message as an attachment to the auto-reply message (in addition to the standard response).
- **Vacation mode** sends only one copy of the auto-reply message to each sender during the defined auto-reply expiration period (by default, one week). This means that if a person sends ten messages within a week to a particular account, and the account uses the Vacation mode of auto-reply, that person receives only one copy of the account's auto-reply message. If the user sends another message to the account after the auto-reply expiration period, he or she receives a second copy of the auto-reply message.

Mailbox Quotas and Related Options

All InterMail accounts that use the local delivery method have an associated set of mailbox quotas. These quotas set limits on the number and size of messages that can be delivered to the account's mailbox. There are three different quotas for each account:

- Maximum message size limits the size of the largest message that can be delivered to the account.
- Maximum mailbox size limits the total storage size of the account's mailbox.
- Maximum number of messages limits the total number of messages that can be in the account's mailbox at any time.

If delivery of mail to an account's mailbox violates any of these quotas, the system bounces or defers the mail (depending on the value of the configuration key `bounceOnQuotaFull`).

Two additional account options control notice of over-quota or near-quota events. You may choose to send either or both of the following notices:

- A bounce notification advises the recipient that a message was bounced because it would have exceeded one of the mailbox quotas.
- A quota threshold warning advises the recipient that his or her mailbox is nearing one or more of its established quotas.

Mail System Access

A series of mail access options controls the methods available for accessing the various InterMail services. For example, the local delivery attribute of an account controls delivery of messages to a mailbox, while the POP3 service option controls whether a user can use the POP server to retrieve mail from his or her mailbox.

The following options define user access to InterMail services:

- **Login name—**All accounts that use local delivery have an associated login name. When retrieving messages with the POP or IMAP a user must supply a login name and password. An account's mailbox cannot be accessed unless the correct login information is supplied.

  Note: Although they frequently are the same, there is no requirement that an account's login name be the same as the account's username. In fact, there are security advantages to ensuring that the two are not the same.

- **POP3 access—**Controls access to standard POP3 service. When this is enabled, the user can access the POP server through the standard (non secure) POP3 port. Because POP3 is the most common method of accessing mail, systems typically enable this option for all accounts that use local delivery.

- **POP3 with SSL—**Controls POP access using Secure Socket Layer (SSL). When this is enabled, the user can access the POP server through the SSL (secure) POP port. A user must have an SSL-compliant POP3 client to use this feature.
- **IMAP4 access—**Controls access to standard IMAP service. Because IMAP access usually requires longer connection times and more storage capacity than POP, this option is usually enabled for only a limited set of accounts.
- **IMAP4 with SSL—**Controls IMAP access using SSL. When this is enabled, the user can access the IMAP server through the SSL (secure) IMAP port.
- **Authenticated SMTP—**Specifies that the user must provide authentication information when sending mail using SMTP. If this is enabled, the system accepts a message only if the user provides the appropriate username and password information. If authentication information is missing or incorrect, the system rejects the message.
- **SMTP access—**Controls users' ability to send e-mail with the standard (nonsecure) SMTP port. This option is used only when the authenticated SMTP option is enabled.
- **SMTP with SSL—**Controls user access to sending e-mail with the SSL (secure) SMTP port. This option is used only when the authenticated SMTP option is enabled.

Class-of-Service Identification

Account attributes can be inherited from a class of service. The class-of-service identification number indicates the class of service with which a particular account is affiliated.

Allocations

The allocations feature of InterMail enables you to count and limit account services used by a customer. For example, you can use allocations to limit:

- The total number of accounts that your customer can create
- The number of a particular kind of account, such as SMTP, POP, or IMAP, that your customer can create
- The number of SMTP, POP, and IMAP accounts that can use SSL encryption
- The number of domains that your customer can define

Allocations are initially set when you install the ISD. For a complete list of allocation attributes, and for instructions on customizing allocation attributes, see the *Integrated Services Directory User Guide*.

Special-Purpose Accounts

InterMail has two special-purpose accounts:

- Wildcard accounts
- Reserved accounts

Wildcard Accounts

All local domains can have an optional wildcard account. A wildcard account is an account within a local domain that receives all mail sent to non-existent addresses within the domain. This feature allows you to collect all mail sent to non-existent addresses within a particular domain in a single account.

Note: Delivery to a wildcard account occurs only when the destination address of a message does not exist as an account's primary address or as an SMTP alias.

For example, if you define the account `john.doe@software.com` as a wildcard account for the local mail domain `software.com`, then any message to a non-existent address within `software.com` will go to `john.doe@software.com`.

The most useful application of wildcard accounts is in servicing "vanity domains" for smaller companies. For example, if mail is sent to `sales@minorcorp.com`, `pres@minorcorp.com`, or `support@minorcorp.com`, adding a wildcard account for `minorcorp.com` allows these messages to be delivered regardless of the existence of an account containing one of these e-mail addresses. In this situation, InterMail handles the mail in two possible ways:

- Delivering it to an actual account if the address is valid
- Delivering it to the designated wildcard account if the address is unknown

Reserved Accounts

InterMail creates several reserved accounts during installation. These accounts are not actively used to receive and deliver mail, but instead receive special administrative mail, such as output from `cron` jobs and system notifications.

InterMail reserved accounts created are:

- `imail`—This account can receive output from `cron` jobs.
- `postmaster`—This account is responsible for sending out bounce messages. In addition, it is a well-known address defined in Internet protocols, reserved so users can be guaranteed at least one valid address at a site.
- `mailer-daemon`—This account is included for legacy purposes. Older systems and administrators who have used the mail protocol may attempt to contact InterMail using the mailer-daemon account.
- `root`—This account is similar to the `imail` account. It may also receive `cron` job and other system messages generated by InterMail.

Note: Reserved accounts are InterMail accounts and should not be confused with UNIX accounts. For example, `imail` and `root` are typical UNIX accounts as well as InterMail accounts for mail handling.

Account Provisioning Rules

InterMail imposes constraints on the account data that can be added to the ISD, and on the users who can modify specific data. These account provisioning rules take the form of:

- ISD schema checking, which ensures that all directory entries conform to the rules defined in the directory schema.
- Access control information (ACI) rules, which define which users or groups of users have permission to modify specific ISD entries.
- Permission to provision mail information, which is granted through the adminPolicy objects in the ISD schema.

For more information on these account provisioning rules, see the *Integrated Services Directory User Guide*.

Creating Accounts with imldapsh

To create an InterMail account through the command-line interface, use the `imldapsh` command with the `CreateAccount` option. The `imldapsh` command enables you to create accounts in various administrative realms in the DIT. In preparation for creating an account, you must define the variable `imdbcAdminRoot`. to specify the location in the DIT of the administrative realm in which you want to create the account. For more information on the DIT, see the *Integrated Services Directory User Guide.*

To define the variable `imdbcAdminRoot`, use the `setenv` command, as follows:

```
setenv IMDBCADMINROOT “<DN>”
```

Where:

`“<DN>”`	Is the distinguished name specifying the administrative realm in which you want to create the account. ***Note:*** *Use quotation marks around the DN.*

For example, to create an account associated with the site licensing InterMail, enter:

```
setenv IMDBCADMINROOT “cn=site, cn=admin root”
```

You are now ready to create an account using the `imldapsh` command with the `CreateAccount` option, abbreviated `ca`:

```
imldapsh ca <pSMTPAddress> <delivery-host> <internal-ID> <POPAddress>
<Password> <Hashing_scheme> <Domain> <Account_status> <Account_type>
<COS>
```

Note: For detailed information on the syntax of `imldapsh`, see the *Integrated Services Directory User Guide*.

For example:

```
imldapsh ca john.doe paris 12345 jdoe rosebud clear
software.com A S Basic
```

This command creates an account with the following attributes:

- The SMTP address `john.doe` in the domain `software.com` (that is, the e-mail address `john.doe@software.com`)
- A mailbox on the host `paris`
- The internal ID number `12345`
- The POP/IMAP login name `jdoe`
- The password `rosebud`
- No password hashing scheme
- An account status of Active (`A`) and an account type of Standard (`S`)
- The class of service `Basic`

To display all of the account attributes associated with an existing account, enter:

```
imldapsh ga <username> <domain>
```

Creating Accounts in Batch Mode

You can create, modify, and delete multiple InterMail accounts using batch provisioning. This commits changes to the Directory database in groups, speeding up the process and reducing network traffic. The following InterMail features support batch provisioning:

- The `imldapsh` directory management utility
- A batch-mode command-line option, `-m`, that extends the standard LDAP directory administration commands `ldapadd`, `ldapmodify`, and `ldapdelete`.
- The configuration keys `ldapBatchTimeoutMS` and `ldapMaxBatchOperations`
- Batch C API functions that conform to the industry-standard LDAP protocol

Using imldapsh

To use `imldapsh` in batch mode:

1. Create an input file with the `imldapsh` command lines you want to run, but without the word `imldapsh`. For example:.

```
CreateAccount smtp00001 paris 1 poplogin00001 pass00001 clear
software.com A S Basic
CreateAccount smtp00002 paris 2 poplogin00002 pass00002 clear
software.com A S Basic
CreateAccount smtp00003 paris 3 poplogin00003 pass00003 clear
software.com A S Basic
CreateAccount smtp00004 paris 4 poplogin00004 pass00004 clear
software.com A S Basic
CreateAccount smtp00005 paris 5 poplogin00005 pass00005 clear
software.com A S Basic
CreateAccount smtp00006 paris 6 poplogin00006 pass00006 clear
software.com A S Basic
CreateAccount smtp00007 paris 7 poplogin00007 pass00007 clear
software.com A S Basic
"input" [New file]
```

> ***Note:*** Before using `CreateAccount`, you must define the `imdbcAdminRoot` variable as described in "Creating Accounts with imldapsh" on page 37.

2. Run `imldapsh` in batch mode by typing:

   ```
   imldapsh < <inputFileName>
   ```

   Where `<inputFileName>` is the name of the file you created in step 1.

Note: It is not necessary for all commands in the file to be `CreateAccount`; different `imldapsh` commands can be specified in the same file.

Using ldapadd

To use the `ldapadd`, `ldapmodify`, and `ldapdelete` administrative commands in batch mode, include the `-m` option in the command line, as follows:

```
ldapadd -m <other options>
```

For example:

```
$ ldapadd -m -D cn=root -w secret -p 51455 -f ldiffile.txt
```

For more information on these utilities and their other options, see the *Integrated Services Directory User Guide*.

Using Configuration Keys

The configuration keys `ldapBatchTimeoutMS` and `ldapMaxBatchOperations` enable you to control when batch provisioning begins and ends.

Use the `ldapBatchTimeoutMS` key to set the time period, in milliseconds, for which you want the server to wait before committing the requests it has received. This key is

used with batch mode as a safeguard against interrupted streams of requests. Possible values range from 1 to 86400, with a default of 20.

Use the `ldapMaxBatchOperations` key to set the maximum number of operations that you want batched together before an automatic commit is done. This key is used as a safeguard against overloading of the server. Possible values range from 1 to 1000, with a default of 15.

For more information on these configuration keys, see the *Integrated Services Directory User Guide.*

Using C API Functions

The LDAP batch C API functions extend the industry-standard LDAP protocol as specified in RFCs 2251 and 1823. To view those RFCs, go to http://www.ietf.org/rfc/rfc2251.txt and -/rfc1823.txt. For a complete discussion of the batch API functions, see the *InterMail Mx Reference Guide.*

Modifying Accounts

You can modify account information with `imldapsh` by using the `ModifyAccountSmtp` option, abbreviated `mas`:

```
imldapsh mas <username> <domain> <newUsername> <newDomain>
```

For example, to modify the SMTP address and domain information for `john.doe`, enter:

```
imldapsh mas john.doe software.com jdoe minorcorp.com
```

This example changes the primary e-mail address of an account from `john.doe@software.com` to `jdoe@minorcorp.com`.

To display the class-of-service attributes associated with the account, use the `GetAccountCos` option, abbreviated `gac`. The syntax for this command is:

```
imldapsh gac <username> <domain>
```

This command displays the names of all the class-of-service attributes for the account, their values (if any) at both the class-of-service and account levels, and the attribute value that currently applies to the account. For example:

```
imldapsh gac jdoe minorcorp.com
```

To define a class-of-service attribute value for an account, use the `SetAccountCos` option, abbreviated `sac`. The attribute value that you set using this command overrides the class-of-service value for that attribute. The syntax for this command is:

```
imldapsh sac <username> <domain> <attribute> <value> …
```


Deleting Accounts

To delete accounts, use the `imldapsh` command with the `DeleteAccount` option, abbreviated `da`:

```
imldapsh da <username> <domain>
```

For example, to delete the account for `john.doe` from the domain `minorcorp.com`, enter:

```
imldapsh da john.doe minorcorp.com
```

Note: Deleting an account with `imldapsh` does not delete the associated mailbox.

Listing Available Accounts

You can use the following command-line utilities to list account information:

- `imbillreport`
- `imaccountreport`
- `imldapsh ListAccounts`

Using imbillreport

You can use the `imbillreport` administrative utility to generate raw billing information from the ISD and store it temporarily in the file system. This billing information includes:

- Realm billing record
  - Total number of domains
  - Number of organizations and organizational units
  - Classes of service
  - Maximum number of accounts
  - Number of allocated mailboxes
  - Number of accounts used
  - Number of accounts of particular types used, such as the number of accounts with POP3 access
- Person billing record
  - First name/last name
  - SMTP address
  - Mailbox size
  - Class of service
  - Account status
  - Billing ID

- Number of SMTP aliases
- Mail delivery host

For a complete list of billing record information generated by this utility, see the *InterMail Mx Reference Guide*.

Note: The site administrator must run `imbillreport` for organizations at the provider level and below, since they do not have access to the ISD.

To process your billing information, use your own billing utility.

To run `imbillreport`, enter:

```
imbillreport <HostName> <PortNumber> <UserName> <Password> <RealmDN>
<OutputDirectory>
```

For detailed syntax information for this utility, see the *InterMail Mx Reference Guide*.

Using imaccountreport

You can use the `imaccountreport` administrative utility to generate a report that summarizes the number and type of accounts created after a specified date. You can print the number of active accounts, the number of suspended accounts, and the total number of accounts of all types created after this date.

To run `imaccountreport`, enter:

```
imaccountreport [-date <date>] [-detail <on|off>] [-U <user>] [-W
<passwd>] [-H <host>] [-help] [Options ...]
```

For detailed syntax information for this utility, see the *Integrated Services Directory User Guide*.

Using imldlapsh ListAccounts

The `imldapsh ListAccounts` command produces a list of account information, such as:

- `Local Delivery`—`P` for POP, `I` for IMAP, `W` for Web, and `N` for No Local Delivery
- `ForwardFlag`—`N` for No Forwarding, `Y` for Forwarding
- `AutoReplyMode`—`N` for No Auto-reply, `E` for Echo Mode, `V` for Vacation Mode, `R` for Reply
- `AutoReplyHost`—The name of the auto-reply host

For example:

```
……
Addr:           zack.clark@minorcorp.com
Pass:           clark
Pass-Type:      C
Type:           S
Status:         A
Host:           paris
IntID:          915896904
Local Delivery: P
ForwardFlag:    N
AutoReplyMode:  N
AutoReplyHost:  paris
……
```

Additionally, you can use a typical UNIX search utility such as `grep` with the `ListAccounts` option, abbreviated `la`. For example, enter the following to list all accounts that have the name `bill` in the account record (including the SMTP address, login/password, or the domain name):

```
imldapsh la | grep bill
```

Setting Auto-Reply for a User

The `imreplyctrl` command allows you to set auto-reply information for individual end users. To use `imreplyctrl` to set an auto-reply message:

1. Create a plain text file for the auto-reply message; for example, `message.txt`.
2. Set message.txt as the autoreply message for the user. For example:

   ```
   imreplyctrl vacation broc.lee paris message.txt
   ```

   Ensure that the user for whom you are adding auto-reply information is associated with a class of service that has the necessary attributes (such as `perm_vacation`, `perm_echo`, `pref_replymode`). If necessary, add the class-of-service attribute by using `imldapsh sac` for that user.

For more information on `imreplyctrl`, see the *InterMail Mx Reference Guide*.

Working with Class-of-Service Attributes

There may be occasions when modifications to class-of-service attributes must be made at the account level. This need may arise, for example, when a particular class of service does not completely fulfill the needs of all users and extra privileges need to be established on an account-by-account basis.

To list the class-of-service attributes and values for an account, enter:

```
imldapsh GetAccountCos <username> <domain>
```

To modify individual class-of-service attributes, enter:

```
imldapsh SetAccountCos <username> <domain> <attribute> <value> …
```

For more information on the `imldapsh` command, see the *Integrated Services Directory User Guide*.

Advanced Account Management

In account administration, many account tasks are typically performed simultaneously. For example, a user may need to be created, assigned a class of service, and placed in an organization at the same time.

There are many ways to develop user applications using the various interfaces. This section provides suggestions for using various interfaces.

If the task you wish to complete will be repeated later, consider building a front-end to the appropriate InterMail interface that can both accept user input and that is generic so that it can be reused. For example, if you prefer or require Web access, construct a Web form with text input areas. This form can then invoke a `cgi` script that will read the text areas (user data) and populate the appropriate interface.

Another possibility is to write a shell script that defines user input and subsequently use this input when creating a C API source file, or invoking a command-line utility. For example:

```
#! /usr/bin/csh

# batch loading utility for adding multiple users in InterMail
# Creates an account, assigns the user to the Premium Class of Service,
# then increases the maximum quotas for this user.
#
# Usage: superpremiumuser <Username> <Internal-ID>
#
# <Username> Name of user (this will also be Login and Password)
# <Internal-ID> The unique Internal ID (check existing IDs for
conflict)

set smtpaddr = $1
set popname = $1
set password = $1
set InternalID = $2

# set fixed values for this script

set domain = software.com
set msshost = sbs-idsun4
if ($#argv == 2) goto makeuser

echo "Usage: createusers <Username> <Internal-ID>"
exit 1

makeuser:
imldapsh ca $smtpaddr $msshost $InternalID $popname $password clear
$domain A S premium
imldapsh sac $smtpaddr $domain pref_quotatotkb 10000000
imldapsh sac $smtpaddr $domain pref_quotamsgkb 1000000
```

The `Superpremiumuser` script in the example:

- Creates a new user
- Assigns the user to the Premium class of service

- Modifies two class-of-service attributes, `pref_quotatotkb` and `pref_quotamsgkb`, to allow the user to have greater account quotas than typical members of the Premium class of service.

After creating this script and adding execute permission, invoke it:

```
Superpremiumuser joebob 1284738
```

In this example, `joebob` is the value for the SMTP address (which will be used as the login and password), and `1284738` is the internal ID.

4

System Configuration

This chapter discusses the tasks related to viewing and updating the Configuration database, and includes instructions for making configuration changes.

The topics covered in this chapter are:

- InterMail Configuration Structure
- Modifying Configuration Data

InterMail Configuration Structure

The Configuration database contains configuration values for each server. This configuration data resides in the `config.db` file, and controls all InterMail components. The Configuration server manages the content of this database by responding to server and application requests for updated configuration information.

When you change configuration data, the Configuration server sends the modified configuration data to each server. Each server responds by indicating the impact of the new changes. For example, you may have to restart a server to activate your changes.

Configuration Keys

The Configuration database contains configuration keys that represent a specific aspect of InterMail server functionality or behavior.

Every entry in the Configuration database has the following syntax:

```
/<host>/<server>/<keyName>: [<value>]
```

Where:

`host`	Defines the host to which the key applies. Use the wildcard character (*) to define entries in the Configuration database that apply to all hosts.

`server`	Defines the specific InterMail server to which the key applies. Use `common` in place of a specific server to indicate that the entry applies to all servers.
`keyName`	Specifies the name of the configuration key. For example, MTA configuration key names can have more than one hierarchical level.
`value`	Defines the value assigned to the specified configuration key. Configuration keys can take single or multiple values. All configuration key values appear between square brackets.

Configuration Key Hierarchy

You can define the same configuration key in several different ways, depending on the combination of server and host. For example, one entry can define a key value for all servers, while a second entry defines a different value for the same key for only one particular server.

The InterMail configuration hierarchy goes from specific to general. InterMail servers go through the following steps to determine the correct value of a configuration key:

1. The server looks for a Configuration database entry that is host and server specific. For example:

   `/paris/mta/<key>`

2. If there are no entries that are host and server specific, the server looks for keys that apply to a specific host but all servers. For example:

   `/paris/common/<key>`

3. If there are no entries that are host specific, the server looks for keys that apply to all hosts but a specific server. For example:

   `/*/mta/<key>`

4. If there are no entries that are server specific, the server looks for keys that apply to all hosts and servers. For example:

   `/*/common/<key>`

5. If there are no entries for the key in the Configuration database, the server uses the key's default value, if one exists.

The configuration hierarchy allows you to specify system-wide configuration values, but define specific exceptions to those policies on a per-server basis. The order in which the configuration key entries are specified has no impact on the hierarchy.

Modifying Configuration Data

The administration utility `imconfedit` enables you to edit and view the Configuration database in read-only format. Before changes are applied to the Configuration database, `imconfedit` displays a list of the changes and their impact on InterMail servers.

You can modify Configuration database information by:

1. Running `imconfedit`.
2. Editing configuration keys.
3. Confirming your modifications.

Running imconfedit

To run `imconfedit`, enter the command name at the prompt:

```
imconfedit
```

The utility contacts the Configuration server to confirm that it is running and reports its status to the user:

```
% imconfedit
imconfserv is running on paris
```

If the Configuration server is running, `imconfedit` retrieves a copy of the Configuration database and opens it in a text editor determined by the editor environment variable, usually `vi`. The database appears as a long series of entries:

```
/*/common/runDir: [/var/tmp]
/*/common/commonGroup: [imail]
/*/common/commonUser: [imail]
/*/common/logDir: [log]
/*/common/spoolDir: [spool]
...
```

Once the Configuration database appears in the text editor, you can edit its contents to change current configuration settings.

Editing Configuration Keys

To edit the value of an existing entry:

1. Find the entry in the Configuration database that you want to change.
2. Delete the value of the key inside the brackets, and enter a new value.
3. Save your changes and close the text editor.

Note: When making changes to the Configuration database, you should always have at your disposal the comprehensive list of configuration keys given in the *InterMail Mx Reference Guide*. This list includes the name, possible values, and other important characteristics of every configuration key.

The following tips can simplify the task of editing the Configuration database:

- Search for all occurrences of a particular key before modifying any single value anywhere in the configuration hierarchy. A single key may appear multiple times in the Configuration database.
- Keys that signify Boolean options, such as keys that enable or disable a configuration option, can have any of four values: `yes`, `no`, `true`, and `false`. Consistently use only one Boolean value pair, `yes/no` or `true/false`, for these keys.
- If you want a key to assume its default value, enter this value for the key under `/*/common`, rather than deleting the key from the Configuration database. Deleted keys do not appear in subsequent `imconfedit` operations, so you will not be able to see the key (or its value).
- Do not edit configuration keys with the configuration path `/<host>/sysadmin`. The InterMail system sets and uses these keys; they should not be modified manually.
- When adding a second value to a key that can accommodate multiple values, place the new value (enclosed in square brackets) on its own line beneath the current value. For example:

```
/*/mta/Error-Actions/BadReturn: [hold]
                                [log]
```

- When editing a configuration key whose value spans multiple lines (such as an MTA error message), enter each line of the value (enclosed in square brackets) on its own line under the key. For example:

```
/*/mta/SMTP-Accept/Help/helo: [Usage: HELO domain]
  []
  [HELO announces the domain name of the sending host.]
  [This is required at the start of every SMTP session.]
```


Committing Configuration Modifications

To commit your configuration modifications:

1. Exit the `config.db` file.

   The `imconfedit` utility displays a review of the configuration changes that you have made and prompts you as to how to proceed:

```
The changes you have made are:
-------------------------------------------------------------------
12c12
< /*/common/dirRmeConnections: [40]
---
> /*/common/dirRmeConnections: [39]
-------------------------------------------------------------------

Do you want to assess the changes now (Re-edit/Quit) [Proceed] ?
```

2. Enter one of the following:
   - `R` to returns to the text editor for further editing.
   - `Q` to cancel all changes and terminates `imconfedit`.
   - `P` to commit your changes to the Configuration database. `imconfedit` queries the InterMail servers about the impact of these changes on their operation, and the results of this query appear:

```
"
*                              Impacts of Changes
:
-------------------------------------------------------------------
*
*******************************************************************
********
*                           Servers Needing Re-starting
*
*******************************************************************
********
* imapserv on paris (dirRmeConnections): requires a re-start
* mss on paris (dirRmeConnections): requires a re-start
* mta on paris (dirRmeConnections): requires a re-start
* popserv on paris (dirRmeConnections): requires a re-start
*
*******************************************************************
********
*                         Parms Not Yet (if ever) Fetched
*
*******************************************************************
********
*
Do you want to install the changes now (Re-edit/Quit) [Proceed] ?
```

3. Enter one of the following:
   - `R`—Returns you to the text editor for further editing.
   - `Q`—Cancels all changes and terminates `imconfedit`.

– `P`—Commits your changes and propagates them to all InterMail hosts.

If you decide to commit your configuration changes, `imconfedit` reports the outcome of the changes. If the configuration changes require that one or more servers restart, `imconfedit` also prompts you as to whether the affected servers should immediately restart:

```
---- The changes have been made on the config server ----
Do you want to re-start the servers now? (Y/N) y
```

Although `imconfedit` can restart servers, this is typically a manual operation for you to perform at an off-peak time or during a maintenance window.

Viewing Configuration Information

In addition to allowing you to edit the Configuration database, `imconfedit` allows you to view configuration information in read-only format.

To view the Configuration database without making changes, enter:

```
imconfedit –viewonly
```

When `imconfedit` runs with this flag, `imconfedit` retrieves a copy of the Configuration database and opens it in a text editor. You can use the editor's search functions to locate the values of individual configuration keys.

5
Security

The InterMail system offers security features that help control the flow of unwanted message traffic through your system, and prevent the viewing and retrieval of mail by unauthorized individuals.

To combat these problems, InterMail can:

- Hold messages suspected of being junk mail so that an administrator can evaluate them
- Terminate client connections from sites or users who abuse the system
- Filter incoming messages for potentially unwanted transactions
- Restrict specified remote sites and users from sending mail to or through the system
- Provide safeguards to prevent the guessing of passwords and e-mail addresses by unscrupulous individuals
- Verify the identity of both the client and the server before allowing transactions

Security Features Overview

InterMail has a number of features designed to restrict use of the system by those who attempt to appropriate its resources to transmit messages where InterMail itself is neither the source nor final destination of such messages. This type of transmission is called message relay. InterMail allows you to determine when message relay is and is not permitted.

Additional security features allow you to intercept suspected junk mail and place it in a specified directory, where it is held temporarily. This process is called sidelining. You can check sidelined mail periodically and if you determine that a given message is junk mail, you can delete it. Alternatively, if you determine that the message is legitimate, you can send it on to its intended recipient.

InterMail also offers password protection and other features to prevent unauthorized individuals from using the system to send, view, or retrieve mail that is not intended for them.

The following table lists each InterMail security feature and its function. The rest of this chapter discusses these features in detail.

Feature	Function
Connection dropping	Used to immediately terminate connections from specific hosts.
Mail blocking	Used to prevent specific users or systems from sending any mail to an InterMail site.
SMTP authentication	Used to prevent individuals from forging addresses (tactic often used by junk e-mailers to hide their true identity).
Relay prevention	Used to control how senders may use InterMail as a relay point for messages.
Message sidelining	Used to place suspected “junk mail” into a holding directory, for later evaluation.
Mail filtering	Provides a series of specific rules that apply to all incoming mail. These rules allow users to extend and refine relay prevention, mail blocking, and message sidelining features.
Per-user mail filtering	Allows you to set customizable mail filtering rules on a per-user basis. This functionality extends the system-level filtering.
Password protection	Used to prevent individuals from retrieving mail for accounts other than their own by discovering account passwords.
LDAP and RME port protection	Used to specify IP addresses that are allowed to connect to the LDAP and RME ports, thereby setting up firewall-like protection for these ports.
Secure socket layer (SSL) authentication	Used to perform a special type of encrypted authentication for message transactions.
POP/IMAP access control by location	Used to limit user access to the POP or IMAP server, based on the location of the user.
Blocking RCPT TO: harvesting	Provides a mechanism to dynamically detect and block `RCPT TO:` harvesters.

Connection Dropping

Connection dropping is a tool for blocking unwanted mail. This feature allows the Message Transport Agent (MTA) to terminate immediately any client connections made by a specific host, or by a host that is sending mail to a large number of recipients.

The main use of connection dropping is to combat denial-of-service attacks on your system. A denial-of-service attack typically involves a program that opens up many client connections to a server for the purpose of disrupting normal activity on that server. If your site has been the victim of an SMTP denial-of-service attack, you can use connection dropping to terminate any future SMTP connections from the site that launched the attack.

InterMail's connection dropping features also allow the MTA to terminate a connection if the client is sending a single message to more than a specific number of users. This allows you to combat the sending of junk e-mail, which typically has hundreds or thousands of addressed recipients.

When InterMail drops a connection, the MTA sends the client a 421 error response code to indicate that it closed the connection.

Note: Like mail blocking, connection dropping is an extreme measure that can disrupt the flow of legitimate mail to your site, so it should be used with caution. For a description of a less drastic method for intercepting junk mail, see "Message Sidelining" on page 73.

Configuration Options

The following configuration keys control connection dropping options:

`dropConnections`	A key to enable (or disable) connection dropping. When the value of this key is `true`, the system compares the IP address of every connecting client with the list of IP addresses specified by `dropTheseIPs`. In addition, if there is a recipient limit defined by `dropMaxMessageRCPTs`, it compares the number of message recipients against this limit. If the connection is in violation of either of these policies, the server immediately terminates the connection.
`dropTheseIPs`	A list of IP addresses of hosts to drop. When the value of `dropConnections` is `true`, the system compares the IP address of each connecting client with the values in this list. If there is a match, it immediately terminates the connection. Use a zero (`0`) as a wildcard to specify all hosts within a network.

`dropMaxMessageRCPTs`	The maximum number of recipient addresses that a message can have before it terminates the client connection. Set the value of this key to `0` to disable connection dropping based on number of recipients.
`dropRcptsReplyText`	The text returned with the 421 error code to dropped clients. The default value for this message is "`Service unavailable.`"

Sample Scenarios

The examples in this section illustrate use of the InterMail connection dropping features, with step-by-step instructions for creating typical configurations.

Combating SMTP Denial-of-Service Attacks

If your site has been the target of an SMTP denial-of-service attack, or you know of a site that is commonly responsible for such attacks, create a connection dropping policy that terminates all client connections made from the attacking site. To set this type of connection dropping policy, perform the following steps:

1. Determine the IP address of the system responsible for the attack, which appears in the MTA log files. Identify any other connections that this site has made to ensure that dropping all connections from this site will not block legitimate mail.
2. Execute the `imconfedit` administrative command to edit the Configuration database (as described in Chapter 6).
3. Locate the key `/*/mta/dropConnections` in the Configuration database. If the value of this key is `false`, change its value to `true`. If the key does not exist in the Configuration database, add it as a new configuration entry. For example:

   ```
   /paris/mta/dropConnections: [true]
   ```

4. Locate the configuration key `/*/mta/dropTheseIPs`. If this key does not already exist, add it as a new entry. Specify the IP address of the offending system as a value of this key, using a zero (`0`) as a wildcard to define all systems within a network. This key can have multiple values, so add as many IP addresses as necessary, with each enclosed in square brackets. For example:

   ```
   /paris/mta/dropTheseIPs: [10.3.21.0][21.5.117.4][21.5.117.5]
   ```

5. Save your changes, committing the new connection dropping policies to the Configuration database.
6. Carefully monitor the logs of each MTA to determine the effects of the new policies. Because connection dropping affects all connections from the listed systems, this kind of policy may prevent your users from receiving legitimate mail.

Dropping Connections from Distributors of Junk E-Mail

The mail blocking features discussed in Blocking Options combat the distribution of junk e-mail best. Unlike connection dropping, mail blocking allows the MTA to log data about rejected messages, which provides valuable information on the effectiveness of the blocking policies. However, only connection dropping allows you to stop mail transmissions based on the number of message recipients.

To drop MTA client connections that transmit junk e-mail, make the following changes to the Configuration database:

1. Locate the key `/*/mta/dropConnections` in the Configuration database. If the value of this key is `false`, change its value to `true`. If the key does not exist in the Configuration database, add it as a new configuration entry. For example:

   ```
   /paris/mta/dropConnections: [true]
   ```

2. Locate the key `/*/mta/dropMaxMessageRCPTs` in the Configuration database, setting its value to the maximum number of recipients allowed for a single message before the system considers the sender a distributor of junk e-mail. If the key does not exist in the Configuration database, add it as a new configuration entry. For example:

   ```
   /paris/mta/dropMaxMessageRCPTs: [1000]
   ```

3. Carefully monitor your MTA logs and the behavior of your system after setting a connection dropping policy. These policies can inadvertently prevent your users from receiving legitimate mail.

Mail Blocking

Mail blocking prevents specific users and/or systems from sending e-mail—any e-mail—to your site. Unlike relay restrictions (described in “Relay Prevention” on page 64), mail blocking is an extreme measure, so it is typically used only after a particular sender has flooded a site with unwanted e-mail. You should be careful how you use mail blocking, so as not to stop legitimate mail as well.

Blocking Options

InterMail blocks messages during their initial transmission to the MTA. When a client connects to the system, the MTA consults the Configuration database to determine the current mail blocking policies. If blocking is on, and the source of the message is a blocked user or system, the MTA blocks the message.

When blocking mail, the MTA reads the headers of the message—but not the body—before rejecting the message. It stops transmission of the message by returning an SMTP error code to the connected client to indicate that message transmission failed. The client is then responsible for taking the appropriate action, which typically includes alerting the sender that delivery failed. The MTA also logs information about the sender and the rejected message, which allows you to review the effects of your

mail blocking policy (for example, you may find that your site is inadvertently blocking mail from legitimate senders).

Note: This is an important difference between mail blocking and connection dropping. Because connection dropping occurs the moment that a server connection is made, it does not allow for logging of information regarding the messages that would have been sent in that transaction.

Blocking Criteria

InterMail allows you to block mail according to any of the following criteria:

- **IP address of the sending system—**This allows you to block all mail sent by a particular system or network. Blocking by IP address is similar to connection dropping (described in "Connection Dropping" on page 55), but allows the MTA to log information on each rejected message.
- **Sender's e-mail address—**This blocking method rejects a message if its sender address (given by the `MAIL FROM` command) matches a list of blocked addresses.
- **Sender's domain—**This option is similar to blocking by e-mail address, but blocks messages based on the domain portion of the sender's return address (given by the `MAIL FROM` command). This allows you to block any sender whose return address includes a particular domain. .
- **Sender's username—**This option is also similar to blocking by e-mail address, but blocks messages based on the username portion of the `MAIL FROM` address, the local portion of the address, given before the `@` character. This allows you to block mail from users who send mail from multiple domains using the same username. .

Note: Because it is easy to forge e-mail addresses, blocking by IP address is the most secure of these methods.

System-Wide Blocking vs. Per-Account Blocking

The InterMail mail blocking feature can operate in two modes: on a system-wide basis, or on a per-account basis.

System-wide mail blocking blocks messages based solely on their sender information. In this mode, the system rejects all incoming mail that matches any of the blocking criteria, regardless of the recipient address. This is the default blocking behavior.

Per-account mail blocking, in contrast, allows you to override the system's mail blocking policies for individual e-mail accounts and thereby block or allow messages based on both the sender and the recipient. If the recipient account has not enabled blocking, the system delivers the message normally, regardless of whether it is blocking the sender. For instance, you can define policies to block all mail sent from a particular host, but still allow certain end users to receive mail from that host.

Note: Per-account mail blocking simply allows accounts to accept or ignore the entire set of administrator-defined blocking policies. It is not possible to define new blocking criteria—such as additional domains and addresses to block— on a per-account basis.

Configuration Options

Configuration keys set mail blocking options to enable particular blocking methods—by IP address, sender e-mail address, sender domain, or sender username—and to specify a list of senders to be blocked. These blocking methods are independent of one another, and you can use as many or as few as you want.

Blocking Mode

A single configuration key defines the mode of mail blocking:

`blockPerAccount`	Key that determines whether the blocking policy is system-wide (blocking policies apply to all incoming messages), or per-account (blocking policies apply only for accounts that have this option enabled). Setting this key to `true` sets the blocking mode to per-account. By default, this key is `false` (for system-wide blocking).

Blocking by E-Mail Address

Three configuration keys define mail blocking based on the e-mail address of the sender:

`blockAddresses`	Key to enable (or disable) mail blocking by e-mail address. When the value of this key is `true`, the system checks the `MAIL FROM` address of each incoming message against the list of addresses defined in `blockTheseAddresses`. If the address matches a listed address, it blocks the message.
`blockLocalNoAcct`	Key to verify the existence of local sender addresses. This option prevents users from fraudulently including one of your domains in their e-mail addresses, which might allow them to avoid mail blocking or other policies. When this key is `true`, it checks the domain of the `MAIL FROM` address against the list of local mail domains. If this address includes a local mail domain, it asks the ISD to verify the existence of the sender address. If the address does not exist, it blocks the message.

`blockTheseAddresses`	A list of e-mail addresses to be blocked. Addresses should include both a username and domain name (for example, `make-money-fast@scamnet.com`).

Blocking by Domain

Two configuration keys define mail blocking by the domain of the sender:

`blockDomains`	Key to enable (or disable) mail blocking by sender domain. When this key is `true`, the system checks the domain of the `MAIL FROM` address of each incoming message against the list of domains defined in `blockTheseDomains`. If the address includes a listed domain, it blocks the message.
`blockTheseDomains`	A list of sender domains to be blocked (`scamnet.com`, for example). Domains can include an optional wildcard (`*`) to specify all subdomains within a domain (for example, `*.scamnet.com`).

Blocking by IP Address

Two configuration keys define mail blocking based on the client IP address:

`blockConnections`	Key to enable (or disable) mail blocking by client IP address. When this key is `true`, the system compares the IP address of a connected client with the list of IP addresses defined in `blockTheseIPs`. If the client IP address matches a listed address, it blocks all messages sent by the client.
`blockTheseIPs`	A list of IP addresses to be blocked. IP addresses can include a zero (`0`) as a wildcard to specify all systems within a network (for example, `10.3.21.0`). You can also enter an IP address as a subnet to specify all systems within a range of IP addresses (for example, `10.3.21.16/24`).

Blocking by Username

Two configuration keys define mail blocking based on the username of the sender:

`blockUsers`	Key to enable (or disable) mail blocking by sender username. When this key is `true`, the system checks the username portion of the `MAIL FROM` address of all incoming messages against the list of usernames defined in `blockTheseUsers`. If the address includes a listed username, it blocks the message.
`blockTheseUsers`	A list of usernames to be blocked (for example, `make-money-fast`).

Mail Blocking Responses

When blocking mail, the MTA must notify the connected client that it did not accept the message. The following configuration keys define this response:

`blockReplyCode`	The three-digit SMTP error code that goes to the client when a message from it is blocked. By default, this value is 550, the standard SMTP code to indicate that the client operation was not successful.
`blockReplyText`	The error text to be returned with the `blockReplyCode` value. This text informs the client of the nature of the message failure. By default, this message is "`You are not allowed to send mail to <recipient>.`"

Sample Scenarios

This section contains examples of mail blocking operations, with instructions for using the available blocking-related configuration keys. Before defining mail blocking policies, it is important that you understand the way that these configuration keys work together to block mail.

Blocking All E-Mail from a Particular System

The most typical usage of mail blocking is to prevent specific sites from sending unsolicited commercial e-mail to your users. The easiest method of stopping such messages is to block all mail from the offending sites. To set this type of blocking policy, perform the following steps:

1. Determine the IP address of the mail server that is responsible for sending unwanted e-mail to your site. This address appears in the MTA log files, as well as in the `Received:` headers of the unwanted messages. Identify any other messages from this site to ensure that blocking this host will not reject legitimate mail.

2. Execute the `imconfedit` administrative command to edit the Configuration database (as described in Chapter 3).
3. Locate the key `/*/mta/blockConnections` in the Configuration database. If the value of this key is `false`, change its value to `true`. If the key does not exist in the Configuration database, add it as a new configuration key entry. For example:

   ```
   /paris/mta/blockConnections: [true]
   ```

4. Locate the configuration key `/*/mta/blockTheseIPs`. If this key does not already exist, add it as a new entry. Specify the IP address of the offending system as a value of this key, using a zero (`0`) as a wildcard to define all systems within a network. This key can have multiple values, so add as many IP addresses as necessary, with each enclosed in square brackets. For example:

   ```
   /paris/mta/blockTheseIPs: [10.3.21.0][21.5.117.4][21.5.117.5]
   ```

5. Save your changes, committing the new blocking policies to the Configuration database.
6. Carefully monitor your MTA to determine the effects of the new blocking policy.

Caution! Because blocking by IP address affects all mail sent from blocked systems, this kind of policy may prevent your users from receiving legitimate mail.

Blocking Mail from Non-Existent Local Addresses

A common issue for Internet service providers is the problem of users who send e-mail from non-existent addresses within the service provider's local domains. For example, a user may use your service to distribute junk e-mail using a forged address. In this case, responses from the recipients of the junk e-mail message would arrive at the non-existent address. This causes your mail system to handle undeliverable mail, which drains system resources.

To prevent the use of non-existent return addresses, make the following changes to the Configuration database:

- Locate the key `/*/mta/blockLocalNoAcct`. If this key is set to `false`, change it to `true`. If the key does not exist in the Configuration database, add it as a new configuration entry. For example:

  ```
  /*/mta/blockLocalNoAcct: [true]
  ```

Blocking Mail from All Users in a Domain

Some distributors of commercial e-mail consistently use addresses from their own domains as the return addresses of junk e-mail. To stop delivering messages from such groups to users at your site, you can block mail according to the domain name of the return address.

To set this type of blocking policy, make the following changes to the Configuration database:

1. Locate the key `/*/mta/blockDomains` in the Configuration database. If the value of this key is `false`, change its value to `true`. If the key does not exist in the Configuration database, add it as a new configuration entry. For example:

   ```
   /paris/mta/blockDomains: [true]
   ```

2. Locate the configuration key `/*/mta/blockTheseDomains`. If this key does not already exist, add it as a new entry. Enter the domain name associated with the offending addresses as a value of this key, using a wildcard (`*`) to specify all subdomains within the domain. This key can have multiple values, so add as many domain names as you need, with each enclosed in square brackets. For example:

   ```
   /paris/mta/blockTheseDomains: [*.cyberspammers.com][junknet.net]
   ```

After you commit these changes to the Configuration database, the MTA rejects any message whose `MAIL FROM` address includes a blocked domain.

SMTP Authentication

An issue related to junk e-mail is the forgery of sender addresses. By using forged addresses, senders of junk e-mail can hide their identities. If senders forge their return addresses to include one of your local domains, they may also bypass some of the other security mechanisms described in this chapter.

To combat address forgery, InterMail supports SMTP authentication. When enabled, this authentication mechanism requires senders to transmit a username and password at the beginning of the SMTP client session. The client session may continue only if the given authentication data matches that of an existing account. When a sender transmits messages, InterMail compares the sender address of the messages— in both the `MAIL FROM` command and the `From:` header—to the addresses associated with the account. If the sender address is not valid for the account (that is, the address is a forgery), it rejects the message.

Note: SMTP authentication requires that the user's e-mail client support the `AUTH_LOGIN` command.

Related Class-of-Service Options

SMTP authentication is set for users on a class-of-service level. The following class-of-service options relate to SMTP authentication. You can enable or disable each option for each class of service.

- **Authenticated SMTP—**This option specifies that the user must provide authentication information when sending mail using SMTP. With this option enabled, if the user submits mail to an MTA, the MTA accepts the message only if

the user provides the appropriate username and password information. If authentication information is withheld or incorrect, the MTA rejects the message.

- **SMTP with SSL—**With the Authenticated SMTP option enabled, this option controls the user's ability to send e-mail by way of the SSL (secure) SMTP port.
- **SMTP—**With the Authenticated SMTP option enabled, this option controls the user's ability to send e-mail by way of the standard (non-secure) SMTP port.

Configuration Options

The following configuration keys affect SMTP authentication:

`CheckAuthentication`	Key to enable (or disable) checks for per-account SMTP authentication. If enabled, the MTA checks each incoming `MAIL FROM` address, as well as the address in the `From:` header line, to see if the SMTP authentication is set for the sender's account in the Integrated Services Directory. If it is, the MTA requires the sender to authenticate using the `AUTH_LOGIN` command.
`RequireSecureAuth`	Key to require users to have a secure connection. If the value of this key is `true`, the MTA allows users to transmit the `AUTH_LOGIN` command only if they have a connection to the SSL port. If `false`, the MTA allows `AUTH_LOGIN` regardless of whether SSL is on.

Relay Prevention

Relay prevention allows you to protect the MTA from third-party relay, a tactic that distributors of "junk" e-mail commonly use.

Relay prevention has implications both for controlling mail traffic and for general system security, since excessive use of system resources devoted to third-party relay can seriously degrade system performance, denying efficient service to the system's legitimate users. Because of this, defining policies that prevent certain types of relay is an important step in preparing your site for full production.

Message Relaying Basics

Message relay occurs every time that a message that arrives at an MTA has a destination on some other MTA. Users relay mail whenever they send a message to another user whose e-mail account is at a different site. eWhen an MTA receives such a message, it must relay (transfer) the message to the appropriate mail server.

Third-Party Relay

In principle, there is nothing wrong with mail relaying; in fact, relaying is one of the primary functions of an MTA, and without it, e-mail could never be routed across the Internet. However, distributors of junk e-mail have abused this functionality by using mail servers at other sites to relay huge volumes of junk mail to users throughout the Internet. This message volume can consume an MTA host's resources, disrupt normal mail operation for hours, and cause junk mail recipients to blame the unwanted messages on the site that relayed them. This type of relay—in which neither the sender nor the recipients of the mail are related to the MTA that relays it—is called third-party relay.

Third-party relay allows distributors of junk e-mail to avoid the cost of the hardware and software necessary to support their level of activity. It effectively allows them to distribute messages at little or no cost to themselves, with the expense borne by the sites that relay and receive the messages.

Relay Strategies

Because relay is both a necessity and a potential liability to an InterMail system, it is crucial to think relay policies through carefully so that you can prevent exploitation of your MTA while still maintaining full service for your users.

InterMail permits a variety of anti-relay options, from no relay restrictions at all (in which case anyone can use an MTA to relay mail) to full relay restrictions (in which case no one can ever use an MTA to relay mail to another system). Having no relay restrictions at all can open an MTA to abuse by any junk mail sender that knows about its vulnerability. On the other hand, preventing any and all relay would mean that users on an InterMail system could only send mail to other users on the same system.

These are extreme examples at either end of the spectrum; however, a given service provider's unique needs may mandate one of these strategies. In most cases, though, a strategy that falls somewhere between these two extremes will be appropriate.

Firewalls

One common relay strategy is to have one MTA that runs behind a network firewall and accepts messages from users within the network, while a second MTA runs outside the firewall and accepts messages from the Internet.

When an MTA outside the firewall receives messages for local users, it routes them to the MTA inside the firewall. In this case, the MTA outside the firewall is neither the source nor the final destination of the messages that it receives. Technically, this MTA performs third-party relaying to the MTA, which is inside the firewall. Configuring the firewall MTA to deny all third-party relays would mean that it could not route messages to local users inside the firewall. Local users could thus never receive mail from the Internet.

In general, it is desirable to deny almost all third-party relay for MTAs that run outside of a network firewall, with limited exceptions that allow for delivery of mail to specific domains, or from specific sources. For MTAs that are behind the firewall,

relay prevention is typically necessary only if you want to prevent users within your network from sending mail to particular domains.

Anti-Relay Options

InterMail includes a number of options for controlling how mail may be relayed through an MTA. These options support four basic relay policies:

- **Completely unrestricted—**This is the default system behavior, and allows relay in all cases regardless of the sender or the destination.
- **Unrestricted with exceptions—**This policy maintains a mostly open relay policy, with restrictions that prevent relay only for specific hosts and/or users. This is the most commonly used relay policy.
- **Completely restricted—**Because this policy restricts all relay, it causes the MTA to accept only messages to addresses of users in local mail domains, or to other specific domains.
- **Restricted with exceptions—**This option allows you to maintain a mostly closed system, with relay allowed only under certain circumstances.

You can define these relay policies for InterMail as a whole or for each MTA independently, depending upon your own requirements.

Restricted vs. Prevented

Restricting relay in InterMail does not necessarily mean preventing relay. In InterMail terminology, “restricted” means that there is a suspicion that a message may be junk e-mail, and that it is a candidate for denial. Mail that is restricted may still be deliverable to its destination domain.

For example, you may decide to restrict all relay mail, and then allow delivery only to one domain. In this case, relay is actually allowed, but only for mail addressed to the designated domain.

In general, relay is restricted in order to limit its use to specific parties. Relay is denied because of a combination of both its source and final destination.

When determining whether to allow potential relay, InterMail uses the following logic:

1. Is this message addressed to a user in a local mail domain? If yes, accept the message; if not, proceed to step 2.
2. May the source of this message (a system and/or user) relay mail through the system, as defined by current policies? If yes, accept the message and send it to the destination domain; if not, proceed to step 3.
3. Is the destination of this message a domain that may receive restricted relay mail, as the current relay policies define? If yes, accept the message and send it to the destination domain; if not, reject the message.

Relay Source Policies

The principal method of preventing mail relay in InterMail is by restricting the systems and users who may relay. This can be defined in four ways:

- **IP address—**This restricts any system whose IP address does not appear in a list of allowable addresses (or which appears in a list of denied address) from sending relay mail. IP addresses can be specified with a wildcard character, which allows (or restricts) relay from systems in a range of IP addresses.
- **E-mail address—**This restricts relay from specific senders. Relay is allowed if the `MAIL FROM` address is in the list of allowed e-mail addresses (or is not in the list of restricted e-mail addresses). When granting relay privileges by e-mail address, you can optionally choose to verify that each sender address exists in the Integrated Services Directory before allowing the relay.
- **Domain**—Similar to e-mail address restriction, restricting by domain also uses the `MAIL FROM` address of the message to determine relay privileges. Relay is allowed if the domain of the `MAIL FROM` address is in the list of allowed domains (or is not in the list of restricted domains). You can specify domains with a wildcard character to include all subdomains within a particular domain..
- **Username—**Also similar to restricting by e-mail address, this option determines relay privileges based on the username portion of the `MAIL FROM` address (the local portion of the address before the `@` character). This allows you to restrict relay by the same username from multiple domains. .

Note: Because it is easy to forge e-mail addresses and domain names, restricting by IP address is by far the most secure of these methods. Restrictions should thus be defined by IP address whenever possible.

Relay Destination Policies

If relay policies restrict the source of a message from relaying, the fate of that message then depends on its intended destination address. If the destination domain is one that can receive restricted relay mail, normal delivery of that message occurs. If the domain is among those that cannot receive restricted mail, the system rejects the message.

As with relay source restrictions, you can define allowable destination domains in multiple ways:

- **All domains (with specific exceptions)**—This allows delivery of restricted relay to occur unless the destination domain appears on a deny list.
- **No domains (with specific exceptions)**—This prevents delivery unless the destination domain appears on an allow list.

Because allowing delivery of restricted mail is an exception to the relay restriction rules, the most common delivery policy is to deny delivery except for messages

addressed to particular domains (for example, a domain for which an InterMail site is an MX backup).

Note: A restricted message addressed to multiple recipients is deliverable only to those recipients in domains that may receive relayed mail. A restricted message is not deliverable to recipients in domains that may not receive relayed mail. The sender will receive notification that the denied recipients did not receive the message.

Configuration Options

A combination of configuration keys defines InterMail relay policies. Most of these configuration keys fall into one of three categories:

- Keys that define general relay policy.
- Keys that define restrictions on the sources of relay mail.
- Keys that define allowable destinations for restricted relay mail.

Additional configuration keys specify the response to clients who cannot relay mail.

General Relay Policy

The following configuration keys define general relay policy. These keys determine the overall InterMail anti-relay strategy, as well as the behavior of other relay-related keys:

`relayMaxRCPTs`	Maximum number of recipients that a message may have before relay checks are performed. This key allows you to exempt messages from your anti-relay policies if they are addressed to a few recipients. For example, if this value is set to 3 and a user attempts to relay a message addressed to 2 recipients, the relay is allowed regardless of other relay policies. To cause all messages to go through relay checks, set the value to `0`. ***Note:*** *If a message has more recipients than* `relayMaxRCPTs`, *but* `relaySourcePolicy` *is set to* `allowAll`, *relay is allowed.*

`relaySourcePolicy`	Definition of the overall relay policy. You can define this policy to allow all relay (`allowAll`), allow relay generally but deny relay from specific users/hosts/domains (`denyListed`), restrict relay generally but allow relay from specific users/hosts/domains (`allowListed`), and restrict all relay (`denyAll`). When defining relay restrictions, you should always start by setting the value of this key, or by verifying its value. To prevent all third-party relays, set this key to `denyAll`.

Defining Relay Sources

The following configuration keys define restrictions on the sources of relay. These keys define either the users/hosts/domains that you allow to relay (when `relaySourcePolicy` is set to `allowListed`) or the users/hosts/domains that you restrict from relaying (when `relaySourcePolicy` is set to `denyListed`):

`relayLocalDomainsOk`	Option to include local mail domains in the list of domains specified by `relaySourceDomainList`. When this key is set to `true`, all messages whose return address includes a local mail domain can be relayed without restriction. ***Note:*** *This option applies only if* `relaySourcePolicy` *is set to* `allowListed` *(restricting relay except for listed hosts, domains, and users).*
`relayLocalMustExist`	Option to verify local senders before relay is allowed. When the value of this key is set to `true`, and the return address of a message includes a local mail domain, InterMail confirms the existence of the sender in the ISD. If the address exists, the system allows relay; if the address does not exist, the system restricts relay.
`relayNullRestricted`	Option for restricting messages that have a null (<>) return address. This option applies only if `relaySourcePolicy` allows relay except from specified hosts, domains, and users. The possible settings are `true` and `false`.
`relaySourceDomainList`	A list of source domains. The relay policy defined by `relaySourcePolicy` will apply to the domains in this list. When the `MAIL FROM` address includes a domain in this list, the value of `relaySourcePolicy` determines whether the message restricted or allowed.

`relaySourceLocalIPList`	A list of local IP addresses (on the receiving MTA) to which the relay policy defined by `relaySourcePolicy` applies. When an InterMail MTA receives a message on an IP address that appears in this list, the value of `relaySourcePolicy` for this host determines whether the message is restricted or allowed. For example, suppose your MTA has two Ethernet interfaces: one has the IP address `10.18.5.1`, and the other has the IP address `10.18.5.2`. You set up your network routers so that systems from your own private network can connect only to `10.18.5.1`, whereas systems from the Internet can connect only to `10.18.5.2`. By creating a `relaySourceLocalIPList` entry for the IP address `10.18.5.1`, you can specify that only users on your own private network—that is, users who connect to the listed IP address—can relay through this MTA.
`relaySourceRemoteIPList`	A list of remote IP addresses to which the relay policy defined by `relaySourcePolicy` applies. When a message arrives from a host whose IP address appears on this list, the value of `relaySourcePolicy` determines whether the message is restricted or allowed.

Defining Relay Destinations

The system allows relay mail if it satisfies the defined source policies regardless of the mail's destination domain. However, delivery of restricted relay mail can be allowed or denied on a destination-by-destination basis. The following keys define the domains that are permitted to receive messages that relay source policies restrict:

`relayDestAllowList`	A list of domains to which messages can be relayed, regardless of restrictions on the source of the messages. Including domains in this list implies that unlisted domains will not receive mail that your relay source policies restrict.
`relayDestDenyList`	Destination domains for which you do not allow restricted relay mail. Any message from a restricted source (according to your relay source policies) cannot be relayed to these domains. Including domains in this list implies that all other destination domains receive relay mail regardless of source restrictions.

When defining allowable destination domains, it is not necessary to specify local mail domains in `relayDestAllowList`. The assumption is that this list includes local mail domains.

Note: If you want to disallow delivery of relay mail to a local mail domain, include the local mail domain in the list of domains that `relayDestDenyList` defines.

Responses to Denied Relay

When InterMail does not allow relay, the MTA must notify the connected client that it did not accept the message. The following configuration keys define this response:

`relayReplyCode`	The three-digit SMTP error code sent to the client when the system denies a relayed message. By default, this value is 550, the standard SMTP code to indicate that the client operation was not successful.
`relayReplyText`	The error text to be returned to the client with the `relayReplyCode` value. This text informs the client of the nature of the message failure. By default, this message is: `Relaying to <domain> is not allowed.`

Sample Scenarios

This section contains some typical relay prevention scenarios, with instructions for setting the required configuration keys. Although this section does not discuss all relay prevention strategies, the examples illustrate the interaction among the many configuration keys involved in relay prevention. An understanding of these keys, and how they work together, is a prerequisite for creating effective InterMail relay prevention policies.

Preventing Third-Party Relay

To prevent third-party relay, you should define configuration options for your MTA to establish the following policies:

- Mail received from within your own network (as defined by IP address) is not rejected. This allows users connected to your network to relay mail through the MTA without restriction.
- All mail received from systems outside of your network is restricted.
- If the restricted mail is addressed to an account within your local mail domains, the system allows delivery of this mail. This allows normal delivery of messages addressed to your users from anywhere throughout the Internet.
- All other restricted relay mail—which, by definition, is both sent from and addressed to users outside your network—is rejected.

To define these policies, modify the Configuration database as follows:

1. Create a new `relaySourcePolicy` configuration key entry for the host on which your system is running, and set the value of this key to `allowListed`. This sets a relay policy for the MTA that is restricted except for specified systems or domains. For example:

   ```
   /paris/mta/relaySourcePolicy: [allowListed]
   ```

2. Create a new `relaySourceLocalIPList` configuration key entry for this host, setting its value to a list of the IP addresses of your network. This specifies that the only systems allowed to relay mail freely through your MTA are systems within your network. Use a `0` (zero) as a wildcard to specify a range of IP addresses. For example:

   ```
   /paris/mta/relaySourceLocalIPList:
   [10.2.3.0][10.3.21.0][127.0.0.1]
   ```

When you have committed these changes to the Configuration database, the MTA rejects all third-party relay attempts made by users outside of your network.

Note: The MTA does not have to be restarted after modification of the configuration keys for these changes to take effect.

Allowing Delivery of Restricted Relay Mail

By default, the system denies restricted relay mail regardless of its destination domain. However, you may want to allow delivery of restricted mail. For example, if your site is an MX backup for another domain, you should allow mail to be relayed to that domain.

To allow delivery of restricted relay mail to one or more domains, set the following options in the Configuration database:

1. If you have not already done so, set up your relay source restrictions using `relaySourcePolicy` and its related configuration keys (described in the previous example).

2. Create a new `relayDestAllowList` configuration key entry as follows:

   ```
   /*/mta/relayDestAllowList:
   ```

3. As the value for the new configuration key, enter the list of destination domains that should receive restricted relay mail. Remember to enclose each domain name between square brackets. Use the asterisk character (`*`) as a wildcard to specify all subdomains of a particular domain. For example:

   ```
   /paris/mta/relayDestAllowList: [minicorp.com][*.megacorp.com]
   ```

The values in this example specify that the MTA handles messages addressed to users in the domain `minicorp.com`, and to any subdomain in `megacorp.com`, even if the relay source policies restrict these messages.

Message Sidelining

A less drastic alternative to mail blocking and connection dropping is message sidelining, which allows you to "hold" messages that are likely to be unsolicited commercial e-mail. Unlike mail blocking, which rejects e-mail based on the source of the mail, and connection dropping, which simply terminates connections, sidelining intercepts specific messages based on certain characteristics. This allows you to set specific conditions which, if present, may indicate that a message is junk mail.

Once a message is sidelined, it can be evaluated (either manually or with a filtering agent) to determine whether it is actually junk mail. If it is junk mail, it can be discarded. If not, it can be sent on to its intended recipients.

There are two message characteristics that can sideline mail:

- **Total number of recipients—**Because senders typically address junk e-mail to hundreds or thousands of recipients, any message sent to such a large number of recipients may be suspect. Sidelining by the total number of recipients allows you to review the contents of mass mailings before distribution.
- **Null return address—**Mail servers often send automatic responses—such as bounce notifications, auto-replies, and so on—using the null return address (<>). These automatic notifications are important to mail server operation, so the system never blocks mail from the null address. However, this provides a loophole for junk e-mailers, who can use the null return address as the `MAIL FROM` address of their messages, thereby avoiding address and domain blocking rules. For this reason, the InterMail mail sidelining feature includes an option for sidelining messages from the null address to more than one recipient. Because legitimate automatic responses always use only one user address, this allows sidelining of potential junk e-mail while still permitting legitimate notifications to go through.

If message sidelining is turned on, and an incoming message satisfies either of these sidelining criteria, the message goes to the `sideline` directory. You can inspect the messages in this directory to determine whether to deliver them. If you consider a message legitimate, you can use the `immsgprocess` administrative command to reintroduce it into the system. If you consider a message unwanted, you can simply delete it.

Note: The system does not process the mail in the `sideline` directory automatically. If you choose to sideline potential junk e-mail, be sure to review and handle the sidelined messages periodically.

Configuration Options

The following configuration keys define message sidelining policies:

`sidelineMessages`	Enables (or disables) message sidelining. If the value of this key is `true`, messages that violate the `sidelineNumRcpts` or `sidelineNullToMany` values move to the sideline directory.
`sidelineNullToMany`	Sidelines all messages sent from the null address (<>) to more than one recipient.
`sidelineNumRcpts`	Sidelines a message with this number of recipients. If the value is zero (the default value), this feature is disabled.
`sidelineNumRcptsPerConnection`	Defines the maximum number of recipients allowed over a given connection before message sidelining begins. If the value is zero (the default value), there is no limit.

Viewing Sidelined Mail

With sidelining enabled, if an incoming message violates one of the sidelining policies, the system moves the Header, Control, and Body files of the message to the `sideline` directory. For example, suppose the system sidelines a message with the following ID:

```
19990114235158898.AAA250@oslo.software.com
```

In this case the following files move to the `$INTERMAIL/queue/sidelined` directory:

```
19990114235158898.AAA250@oslo.software.com-Control
19990114235158898.AAA250@oslo.software.com-Header
19990114235158898.AAA250@oslo.software.com-Body
```

To review the contents of a message, simply use a text editor to open the Header or Body file. Once you have reviewed a sidelined message, you should either delete the mail or reprocess it to allow normal delivery. To delete mail, you use normal operating system commands to delete the Header, Control, and Body files of the message from the `sideline` directory. For example, to remove the message files shown in the previous example, enter:

```
rm 19990114235158898.AAA250@oslo.software.com-*
```

Note: For an introduction to Control, Header, and Body files, refer to the discussion of message files in Chapter 7.

Processing Sidelined Mail

Reprocessing a message requires you to use the `immsgprocess` administrative command. When executed, `immsgprocess` moves the Control file of the specified message to the `deferred` directory, where the MTA processes it normally. Meanwhile, the Body and Header files of the message move to appropriate buckets in the `messages` directory.

For example, to reprocess the sidelined message shown in the previous example, you would execute the following command:

```
immsgprocess 19990114235158898.AAA250@oslo.software.com-Control
```

Note: You can specify the Control, Body, or Header file name with `immsgprocess` to reprocess a message.

The `Ok-To-Sideline` header assists in processing sidelined mail. The `Ok-To-Sideline` header is found in the Control file for each message and can have a value of `0` or `1`. A value of `0` indicates that sidelining tests have already been performed and the message either passes the tests or was successfully deposited in the `sideline` directory. A value of `1` indicates that sidelining tests still need to be performed, either because those tests have not yet been executed, or because the message failed the tests but remains in the `deferred` directory because the `sideline` directory was for some reason unavailable.

Use of this header is important for two reasons. First, it uses the system resources efficiently by ensuring that messages are tested against sideline criteria only once. Second, it facilitates processing of sidelined mail that you have reviewed, approved for delivery, and re-submitted for processing via the `immsgprocess` command. The header value of `1` bypasses subsequent sidelining tests, which would deposit the message right back in the `sideline` directory even after review and approval.

Mail Filters

InterMail allows you to extend its relay prevention, mail blocking, and message sidelining features by using custom mail filters. These global filters run against all incoming mail. Based on a message's content, the filters can specify that the message should be rejected, bounced, sidelined, forwarded, deleted, or else delivered normally.

Filter actions can depend on the presence or absence of certain headers, the sender, the recipients, or any other text contained in the headers or body of the message. You can test message content by searching for exact string matches, by using simple patterns, or by using complex regular expressions.

The configuration key `incomingMailFilter` defines mail filters in the Configuration database. The MTA can have only one associated `incomingMailFilter` key, which defines all of the filtering criteria it uses.

For a full description of the `incomingMailFilter` configuration key, see the *InterMail Mx Reference Guide*.

Filter Syntax

The SIEVE filtering language is the basis for InterMail mail filtering. The general syntax of a mail filter is:

```
if <test> <action> [else <action>];
```

Where:

`test`	Specifies a Boolean expression that is `true` or `false`, based on a characteristic of the message. See "Tests" on page 77 for more information.
`action`	Defines an action taken against the message by the MTA. See "Actions" on page 79 for more information.

Filter statements follow the `if-else` syntax that is common to many programming languages, and can have any number of levels. Curly braces ({}) can delimit blocks of syntax. This allows you to construct highly complex filtering criteria, such as:

```
if <test> {
   if <test> <action>;
   else if <test> <action>;
   else if <test> <action>;
   else <action>;
}
else {
   <action>;
}
```

The tests within filter statements can include the following logical operators:

`ANY-OF ( <test>, ...)`	Defines a logical OR which means that if any of the specified tests is true, the entire expression is true.
`ALL-OF ( <test>, ...)`	Defines a logical AND which means that each of the specified tests must be true. If just one of them is false, the entire expression is false.
`NOT <test>`	Defines a logical NOT which means that if the specified test is false, the entire expression is true.
`TRUE`	Is always true.
`FALSE`	Is always false.

Note: Filter keywords are not case-sensitive; they appear in this section in all capital letters for readability purposes only. This is not a requirement when defining filters.

Example

```
if ANY-OF ( <test>, <test>, <test> ) {
   if ALL-OF ( <test>, <test> ) <action>;
   else if NOT <test> <action>;
   else if TRUE <action>;
}
```

Tests

Each test in a filter is an expression that yields a value of `true` or `false`, based on one or more comparisons. There are two general types of tests:

- **Numeric expressions,** which determine action by the number of recipients or the size of the message
- **String expressions,** which search the specific areas of the message for one or more string values

Numeric Expressions

The syntax of a numeric expression is:

```
<message-element> <number-operator> <number>
```

Where:

`message-element`	Specifies an attribute of the message expressed as a number. The possible values are: `RECIPIENTS.COUNT`, the number of recipients `SIZE`, the total size of the message, including attachments
`number-operator`	Specifies a comparative number operator. The possible values are: `OVER`, is greater than `UNDER`, is less than `IS`, equals `IS-NOT`, does not equal `>`, is greater than `>=`, is greater than or equal to `<`, is less than `<=`, is less than or equal to ***Note:*** *The operators* `OVER` *and* `>` *are equivalent, as are the operators* `UNDER` *and* `<`*. Which you use is solely a matter of preference. However, the* `=` *can be used with* `>` *or* `<` *but not with* `OVER` *or* `UNDER`*.*

`number`	Specifies an integer value, with an optional unit of measure. Because the `SIZE` element is in bytes, specifying a unit of measure with a number value provides a convenient method of specifying kilobytes, megabytes, or gigabytes. The possible values are: `K`kilobytes; multiplies value by 1,024 (`1K`, `2K`, and so on) `M`megabytes; multiplies value by 1,048,576 (`1M`, `2M`, and so on) `G`gigabytes; multiplies value by 1,073,741,824 (`1G`, `2G`, and so on)

Example

```
if RECIPIENTS.COUNT >= 100 <action>;
else if SIZE OVER 1M <action>;
```

String Expressions

The general formats of a string expression are:

```
<message-element> <string-operator>[-NOCASE] “<string>”
<message-element> <string-operator>[-NOCASE] ( “<string>”, ... )
```

Where:

`message-element`	Specifies an attribute of the message that expressed as a string. The possible values are: `HEADER “<header>”`, the value of the specified header `HEADER (“<hdr>”, ...)`, the value of the specified headers `SENDER`, the full sender address `SENDER.DOMAIN`, the domain of the sender address `SENDER.LOCAL-PART`, the username of the sender address `RECIPIENTS`, all recipients of the message `BODY`, the message body, including attachments `BODY.TOP(#)`, the first n lines of the body
`string-operator [-NOCASE]`	Specifies a comparative string operator. The possible values are: `CONTAINS`, contains the specified string `IS`, is identical to the specified string `IS-NOT`, is not identical to the specified string `HAS-PREFIX`, begins with the specified string `HAS-SUFFIX`, ends with the specified string `MATCHES`, contains the specified string pattern `CONTAINS-RE`, contains the specified regular expression Any of these operators used with the `-NOCASE` suffix (for example, `CONTAINS-NOCASE`) indicates a case-insensitive query.

`string` — Specifies the text string with which the message element is to be compared, enclosed in double quotes. In the case of the operators `CONTAINS`, `IS`, `IS-NOT`, `HAS-PREFIX`, and `HAS-SUFFIX`, the value of this parameter is a literal string. In the case of `MATCHES`, the value is a character pattern that can include the wildcards `*` and `?` to find partial matches. In the case of `CONTAINS-RE`, the value is evaluated as a regular expression.

Example

```
if BODY.TOP(10) CONTAINS “MLM” <action>;
else if HEADER(“Subject:”) HAS-PREFIX “make.money.fast” <action>;
else if SENDER.DOMAIN MATCHES-NOCASE “*.software.com” <action>;
```

An additional string test uses the keyword `EXISTS` to check for the presence of one or more message headers:

```
EXISTS (“<header>”, “<header>”, ...)
```

If all of the specified headers exist in the message, the `EXISTS` expression is true. If any of the headers are not present, the expression is false. For example:

```
if EXISTS (“To:”, “Subject:”, “From:”, “Date:”) <actions>;
```

Actions

Mail filters can specify that a message is to be deleted, bounced, sidelined, forwarded, or delivered normally. The following keywords define actions in mail filter expressions:

`KEEP`	Delivers the message normally.
`DISCARD`	Deletes the message.
`TOSS`	Deletes the message.
`STOP`	Stops execution immediately. If no action has run, InterMail delivers the message normally. Otherwise, the filter completes the previously executed actions.
`FORWARD "<address>"`	Forwards the message to the specified address. A filter can include multiple `FORWARD` actions. If it specifies `FORWARD` along with `KEEP`, it both forwards and delivers the mail normally.
`SIDELINE "<reason>"`	Sidelines the message. (See “Message Sidelining” on page 73 for information on sidelined mail.)

REJECT ["<reason>"]	Rejects the message. This typically means that the sending client bounces the message back to the sender. The text entered for the "reason" passes back to the client in the SMTP response to the DATA command.
BOUNCE ["<reason>"]	Operates in two different modes, depending on the value of the configuration key blockPerAccount. If the value of this key is false, BOUNCE operates identically to REJECT, and rejects the message. However, if the value of this key is true, the MTA checks account information for each recipient to determine whether the account is using the MTA Filtering class-of-service option. If filtering is turned on for the recipient, it rejects the message for that recipient. If the account does not use mail filtering, it delivers the message normally for that recipient. If the system rejects the mail, the bounce message returned to the sender is: An incoming mail filter has determined that this message should not be delivered. <reason>

Any filter actions other than normal delivery (KEEP) become part of the log, and MTA statistics files reflect the number of times that each filter action has occurred.

Sample Filters

The following examples illustrate simple mail filters. Because InterMail sites often have an extensive list of criteria for handling mail, mail filters are typically longer than those given here. However, the principles and syntax are the same.

Example 1

The following filter uses a string expression to test for the presence of a particular domain in recipient addresses:

```
if RECIPIENTS matches "*@minorcorp.com" BOUNCE;
```

If an incoming message includes a recipient address in the domain minorcorp.com, the MTA bounces the message.

Example 2

The following filter uses a variety of criteria to operate on messages:

```
if size >= 100k {
   if SENDER.DOMAIN IS-NOCASE ("software.com") KEEP;
   else if HEADER("Subject:") IS-NOCASE "Make Money" TOSS;
   else if RECIPIENTS.COUNT >= 100 TOSS;
   else SIDELINE "Too large";
}
```


This filter carries out the following tests:

1. The first line checks the size of the message:

   ```
   if size >= 100k
   ```

   If the size of a message is less than 100 KB, the rest of the filter does not apply to that message, and the message is handled normally. However, if its size does exceed (or equal) this value, the filter continues with the next test.

2. The second line checks the domain portion of the sender's address:

   ```
   if SENDER.DOMAIN IS-NOCASE ("software.com") KEEP;
   ```

   If the sender's address includes the domain `software.com`, the message is delivered normally (`KEEP`). If not, the filter continues with the next test.

3. The third line checks the value of a particular header:

   ```
   else if HEADER("Subject:") IS-NOCASE "Make Money" TOSS;
   ```

   If the message includes the header `Subject:`, and the value of this header is `Make Money` (regardless of case), the message is deleted from the system (`TOSS`). If not, the filter continues with the next test.

4. The fourth line checks the value of a particular header:

   ```
   else if RECIPIENTS.COUNT >= 100 TOSS;
   ```

   If the number of recipients of the message is greater than or equal to 100, then the system deletes the message (`TOSS`). If not, the filter continues with the next test

5. The fifth line specifies that if the message has passed through steps 1 through 4 without being delivered or tossed, it is to be sidelined:

   ```
   else SIDELINE "Too large";
   ```

Note: One benefit of this filter is that it limits the number of messages for you to verify, since some messages that might otherwise have been sidelined are delivered or tossed in the first four steps. This demonstrates how the use of filters can greatly extend the capabilities of message sidelining, relay prevention, and so forth.

Verifying Filters

Because filters appear in the Configuration database, adding mail filters to InterMail requires `imconfedit`. However, `imconfedit` performs no validation checks on mail filters, so it is important that you verify your filters before using them. The administrative command `imfiltercheck` provides this verification.

The usage of this command is:

```
imfiltercheck { -v <filter-file> ... | -vi |-vc <key> |
                -e  [<filter-file>|-c <key>] <message-file> ... |
                -es [<filter-file>|-c <key>] <header-file><body-file>
                |-ei [<filter-file>|-c <key>] }
```

Where:

`-v`	Validates that one or more filters specified in files are valid.
`-vc`	Validates that a filter specified in a configuration key is valid.
`-vi`	Validates that a filter taken from standard input is valid.
`-e`	Executes the given filter on a message defined in a single file.
`-es`	Executes the given filter on a message defined in a Body and Header file.
`-ei`	Executes the given filter on a message taken from standard input.
`filter-file`	Specifies the file that contains the filter to be validated.
`key`	Specifies the configuration key that contains the filter to be validated. The value of this parameter is typically `incomingMailFilter`.
`message-file`	Specifies the message on which the filter is to be executed.

Examples

- To validate a filter defined in one or more files, use the `-v` option:

  ```
  imfiltercheck -v filter1, filter2
  ```

  This example checks the mail filters contained in the files `filter1` and `filter2` to determine if the definition is correct. If they are not correct, it reports the line number and position of the first error.

- To validate a filter already defined in the Configuration Database, use the `-vc` option:

  ```
  imfiltercheck -vc incomingMailFilter
  ```

  This example checks to determine if the definition of the mail filter in the `incomingMailFilter` configuration key for the current host is correct.

- To validate a filter that is taken from standard input, use the `-vi` option:

  ```
  imfiltercheck -vi
  ```

- To execute a filter on one or more existing messages that are defined in single files, use the `-e` option:

  ```
  imfiltercheck -e filter1 test-message1, test-message2
  ```

  This example executes the filter defined in the file `filter1` on the messages defined in the files `test-message1` and `test-message2`.

- To execute a filter on an existing message defined in a header and body file, use the `-es` option:

  ```
  imfiltercheck -es filter1
  19990114235158898.AAA250@oslo.software.com-Header
  19990114235158898.AAA250@oslo.software.com-Body
  ```

  This example executes the filter defined in the file `filter1` on the messages defined in the given Header and Body files.

- To execute a filter on a message taken from standard input, use the `-ei` option:

  ```
  imfiltercheck -ei filter1
  ```

  This example executes the filter defined in the file `filter1` on a message that comes from standard input.

Per-User Mail Filtering

InterMail allows you to set customizable mail filtering rules on a per-user basis. This functionality extends the system-level filtering. Since per-user filters are specified on a per-account level, you have great flexibility when configuring spam prevention methods.

For example, John Doe can have a rule to block all mail from a known spammer; Julie Williams can have all mail from her friend, Brock Lee, sent to her “Friends” folder; and Marge Harding can have no filters at all, allowing mail to be delivered as normal.

Per-user filters are run inside the MTA, just before the message is delivered to the Message Store database. These filters are run only on messages destined for delivery to a local mailbox.

Extensions to the SIEVE Language

To accommodate per-user filtering, the SIEVE filtering language was extended to include four new actions:

- Write-header
- Sendmessage
- Fileinto
- Forward

Write-header

The syntax of this action is:

```
Write-header <"quoted-string"> [,] ["quoted string"]
```

This action takes the current message being delivered to the user and changes the header portion of the message. The quoted string contains the header attribute value pair (with a “:” as a delimiter). If the attribute exists, the old value is replaced with the new value.

Note: The command `write-header` does not affect forwarded messages. Only the header changes following the `HEADER` keyword are made.

Example

If you want to add an X-Header to the message with the value “sieve,” do the following:

```
write-header "X-Header: sieve";
```

Alternatively, if you want to add two headers to the message, do the following:

```
write-header "X-Header: sieve" , "X-JunkMail: true";
```

Sendmessage

The syntax for this action is:

```
Sendmessage <"recipient’s address"> [,] ["recipient’s address"] DATA
\<"quoted string"> [,] ["quoted string"]
```

This action sends a new message to the list of recipients specified. The header and body of the message are specified as a list of quoted strings following the `DATA` keyword.

Example

If you want to send a new message to John Doe, do the following:

```
sendmessage "Brock.Lee@software.com" DATA

"To: John.Doe@software.com",
                                                    "From:
Marge.Harding@accordance.com",
                                    "Subject: Hi",
                                    "",     # mark end of header
                                    "Hello, Marge Harding. How are
you?",
                                    " –John Doe";
```

Fileinto

The syntax for this action is:

```
Fileinto <"foldername">
```

This action files the current message in the user's folder. If the folder does not exist, it is instantly created. This command implies that “keep” was called.

For example, if the users want to file the current message into the “jokes” folder, they must do the following:

```
fileinto "jokes";
```

Alternatively, if they want to file the current message into the “funny” folder (inside the joke folder), they must do the following:

```
fileinto "jokes/funny";
```


Forward

The syntax for this action is:

```
forward <"recipient's address"> [,] ["another recipient's address"]
\[HEADER "quoted string" [,] ["quoted string"]]
```

In this version of InterMail the `forward` action allows you to specify multiple recipients, as well as change the header on the forwarded message.

Example

If you want to forward the current message to `john.doe@software.com` without header changes, do the following:

```
forward "john.doe@software.com"
```

If you want to forward the current message to John Doe and Marge Harding, with one header change, do the following:

```
forward "john.doe@software.com" , "Marge.Harding@accordance.com"
HEADER "subject: This is my forwarded message";
```

If you want to forward the current message to John Doe, change the header sent to Marge Harding, and file the original in the default Inbox folder, do the following:

```
forward "john.doe@software.com",;
forward "marge.harding@accordance.com" HEADER "subject: This is a
forwarded message";
keep;
```

Setting Per-User Mail Filters

You can configure per-user mail filters using any of the following:

- The `pref_mtaFilterPerUser` class-of-service attribute
- The `imfilterctrl` command line utility
- The C APIs

Setting Filters Through Class-of-Service Options

You can enable or disable class-of-service attributes by using `pref_mtaFilterPerUser`. This is necessary for turning SIEVE rules on or off on a per-user basis or on a class-of-service basis.

Example

```
imdbcontrol sac John.Doe software.com pref_mtaFilterPerUser 1
```

Note: If `pref_mtaFilterPerUser` is on (set to `1`), an extra call is made to the Message Store database to retrieve the user's SIEVE filter text, even if the filter text file is empty. This may cause a small performance issue. It is recommended that `pref_mtaFilterPerUser` be disabled when no filters are set.

Setting Filters on the Command Line

The `imfilterctrl` utility is used to set per-user mail filters in the Message Store database. To execute `imfilterctrl`, you must be logged in as the InterMail user on any InterMail host. The `imfilterctrl` command is used to set, retrieve, or remove filter rules for an individual account.

Syntax

```
imfilterctrl <get|set|clear> <address> <domain>
[- |{<filterfile>|<filter-file to stdout>}]
```

Where:

`get`	Retrieves per-user filtering information for a specified user.
`set`	Sets a new per-user rule for a specified user.
`clear`	Removes all rules for a specified user.
`address`	Is the SMTP address of the specified user.
`domain`	Is the domain name of the specified user.
`filter-file`	Is a text file containing filtering rules.
`filter-file to stdout`	Is the file to which filter data is to be written.

Example

```
imfilterctrl set Mario.Garcia software.com sievefilename.txt
```

Setting Per-User Filters with the C API

Three functions in the C API allow you to add and remove per-user filters from the Message Store database:

- `IM_ValidateMtaFilter`
- `IM_UpdateMtaFilter`
- `IM_ReadMtaFilter`

IM_ValidateMtaFilter

```
int  IM_ValidateMtaFilter(IM_Account*, IM_Reply*, IM_Error*);
```

This function compiles and validates the filter text. This is useful in verifying that the SIEVE filter compiles.

Return values are:

`IM_SUCCESS`	The filter is valid.
`IM_FAILURE`	There was a parser error. Possible values of error->number.
`IM_ERROR_FILTER_PARSER`	There was a SIEVE compiler error. Error messages are contained in error->string (same output as `imfiltercheck -e`).
`IM_ERROR_MISSING_ARG`	The routine had a missing argument.

IM_UpdateMtaFilter

```
int  IM_UpdateMtaFilter(IM_Account*, IM_Reply*, IM_Error*);
```

This function stores the filter text for a given user. This function also validates the `filterText` and returns an error if the test cannot be parsed by SIEVE.

Return values are:

`IM_SUCCESS`	The filter is valid and is stored in the user's mailbox.
`IM_FAILURE`	There was an error. Possible values of error->number.
`IM_ERROR_FILTER_PARSER`	There was a SIEVE compiler error. Error messages contained in error->string (same output as `imfiltercheck -e`).
`IM_ERROR_MISSING_ARG`	The routine had a missing argument.
`IM_ERROR_INVALID_ARG`	One of the arguments was invalid.

IM_ReadMtaFilter

```
int  IM_ReadMtaFilter(IM_Account*, IM_Reply*, IM_Error*);
```

This function reads the user SIEVE filter text and places it in the `filterText` structure.

Return values are:

`IM_SUCCESS`	The filter is valid and is stored in the user's mailbox.
`IM_FAILURE`	There was an error. Possible values of error->number.

`IM_ERROR_FILTER_PARSER`	There was a SIEVE compiler error. Error messages contained in error->string (same output as `imfiltercheck -e`).
`IM_ERROR_INVALID_ARG`	One of the arguments was invalid (same as `IM_CreateReply`).

Sample Filters

The following are examples of per-user SIEVE filtering methods.

Example 1

To bounce all messages from “johnDoe@spam.org”:

```
if header "from" contains "johnDoe@spam.org" {
  bounce "This message was bounced because we think it’s spam";
}
```

Example 2

To silently drop all mail from “junkmail.com”:

```
if header "from" contains "junkmail.com" {
  discard;
}
```

Example 3

To forward a message to a pager and deliver the message to an inbox:

```
if header "subject" contains "urgent" {
  forward "mypager@skytel.com";
}
```

Example 4

To forward a message to a fax server and not deliver the message to an inbox:

```
if any-of (header "to" contains "fax", header "cc" contains "fax") {
  forward "myfax@server.com";
}
```

Example 5

To file all messages that have no `from` or `date` field in a folder called “spam”:

```
if not exists ("From" "Date") {
  fileinto "spam";
}
```

Password Protection

One common abuse of e-mail systems involves users who retrieve e-mail from accounts other than their own. They can do this by guessing the target account's password, often sending dozens or hundreds of authentication attempts before discovering the password value. Once someone has successfully guessed the password of an account, they can access mail sent to that account using any e-mail client.

InterMail allows you to combat this type of improper mail access with a series of options that deter users from guessing passwords. These options apply to both the POP and IMAP servers, so failed authentication attempts with either of these servers have the same effect. InterMail's password protection features include the following methods of deterring password guesses:

- **Limited number of authentication attempts**—When a user submits more than a certain number of incorrect passwords, the client connection terminates.
- **Increasing delays after incorrect passwords are given**—If user authentication fails because of an incorrect password, the client must wait for a defined number of seconds before a subsequent authentication attempt is possible. With each failed attempt, the delay time increases.
- **Tracking of sources of incorrect passwords**—The system tracks hosts responsible for sending failed login attempts for a specific period of time. This allows the system to enforce increasing password delays on single users who attempt to guess the passwords of multiple accounts during the same transaction.
- **Logging of password failures**—Several logging options allow you to record information about failed authentication attempts. This information alerts you to incidents of password guessing from particular systems or against particular accounts.

Formula for Authentication Delays

If an authentication attempt fails, InterMail imposes a delay in response to subsequent authentication attempts. This delay calculation depends on the following factors:

- The number of failed password attempts for this account
- The number of failed password attempts for any account from the same client IP address
- The authentication delay value specified by the `badPasswordDelay` configuration key

The formula for calculating the delay is:

$$\left(\begin{array}{c}\text{number of failures on an account}\\ + \\ \text{failures by a particular IP address}\end{array}\right) * \begin{array}{l}\text{number of seconds defined}\\ \text{by } \texttt{badPasswordDelay}\end{array} = \text{delay}$$

For example, suppose a user at IP address `29.82.172.24` attempts to guess the password of John Doe's account, and fails. The user is immediately notified that the authentication failed, and promptly attempts another guess. When the second authentication attempt also fails, InterMail delays notifying the user of the failure for a certain number of seconds. Assuming that the `badPasswordDelay` value is 5 seconds (the default), the delay after the second authentication attempt is 10 seconds:

(1 failure on John Doe's account + 1 failure from [29.82.172.24]) * 5 seconds = 10 seconds

When the user tries and fails a third time to guess the account's password, the delay for notification is 20 seconds:

(2 failures on John Doe's account + 2 failures from `[29.82.172.24]`) * 5 seconds = 20 seconds

Finally, the user gives up trying to guess the password of John Doe's account, and instead attempts to access Susie Queue's account. When the user tries and fails to guess the password of this account, InterMail delays the client notification for 20 seconds, even though the user has not previously attempted to guess the password of this account:

(1 failure on Susie Queue's account + 3 failures from `[29.82.172.24]`) * 5 seconds = 20 seconds

These delay times continue to increase until they reach the configurable maximum delay time. The delay time resets to zero if a user gives a valid password, or if the window for tracking specific IP addresses and accounts (also configurable) has expired.

Configuration Options

The following configuration keys control password protection policies on both the POP and IMAP servers:

`badPasswordDelay`	The number of seconds of delay of notification of a failed authentication attempt. With each failed attempt, this number of seconds adds to the delay time up to the limit defined by the value of `maxBadPasswordDelay`.

`badPasswordWindow`	The number of minutes for which users and IP addresses are to be tracked after an incorrect password. After this number of minutes have elapsed, InterMail resets the number of failed password attempts made against a particular account or from a particular system (both are part of calculating password delay time).
`maxBadPassword`	The number of failed authentication attempts allowed before the client connection is dropped.
`maxBadPasswordAddrs`	Each time a failed authentication occurs, InterMail stores the IP address of the client that issued the incorrect login information. This key controls the maximum size of this IP address list. By default, the size of this list is 10,240. When the list reaches this limit, the system logs the event and resets the list size to 0.
`maxBadPasswordDelay`	The maximum delay time for authentication attempts. If the delay time reaches this limit, there is no further increase in delay time with subsequent failed authentication attempts.
`maxBadPasswordUsers`	Each time a failed authentication occurs, InterMail stores the login name of the account for which the incorrect password was given. This key controls the maximum size of this login name list. By default, the size of this list is 10,240. When the list reaches this limit, the system logs the event and resets the list size to 0.
`maxPasswordFailures`	Specifies the number of authentication failures on an account or by an IP address before connections from that account or address are dropped. When a user submits the number of failed login attempts specified by this key during the number of minutes specified by `badPasswordWindow`, the system records an `AcctBadPswdMaxFailures` event in the POP or IMAP server log.

LDAP and RME Port Protection

Through its implementation of the MSS and Directory server, InterMail contains LDAP and RME ports. In order to protect these ports from being potential denial-of-service attack points, InterMail has several controls that allow you to specify privileged IP addresses or netmasks. If these controls are used, only the specified IP addresses are allowed to connect to the LDAP and RME ports, thereby setting up firewall-like protection for these ports. The `ldapAccessList` and `rmeAccessList` configuration keys control this mechanism:

`ldapAccessList`	A list of IP addresses or IP masks which are allowed to connect to the LDAP port. If a client connects from an address, which is not in this list, the connection is refused and a `NioConnNotAllowed` error is logged. By default, access is only allowed from the local machine.
`rmeAccessList`	A list of IP addresses or IP masks which are allowed to connect to RME ports (such as the MSS port or the Directory RME port). If a client connects from an address that is not in this list, the connection is refused and a `NioConnNotAllowed` error is logged. By default, access is only allowed from the local machine.

Note: The event message logged when an illegal connection is attempted is the existing `NioConnNotAllowed` message, which specifies the IP address of the host attempting to connect, along with the port type being either `ldapAccessList` or `rmeAccessList`.

Secure Socket Layer (SSL) Authentication

InterMail incorporates the Secure Socket Layer (SSL) into the MTA, POP, and IMAP servers. This allows both client and server to verify authentication in mail transactions by operating on alternate secure ports.

SSL Data Flow

SSL operates on a low-level TCP layer and performs authentication before any message transactions occur between client and server. SSL involves encryption, allowing SSL-capable clients to access a special key/certificate pair on the server in order to secure transactions and prevent unauthorized “listening” and authentication. The basic order of events with SSL in InterMail appears in SSL data flow.

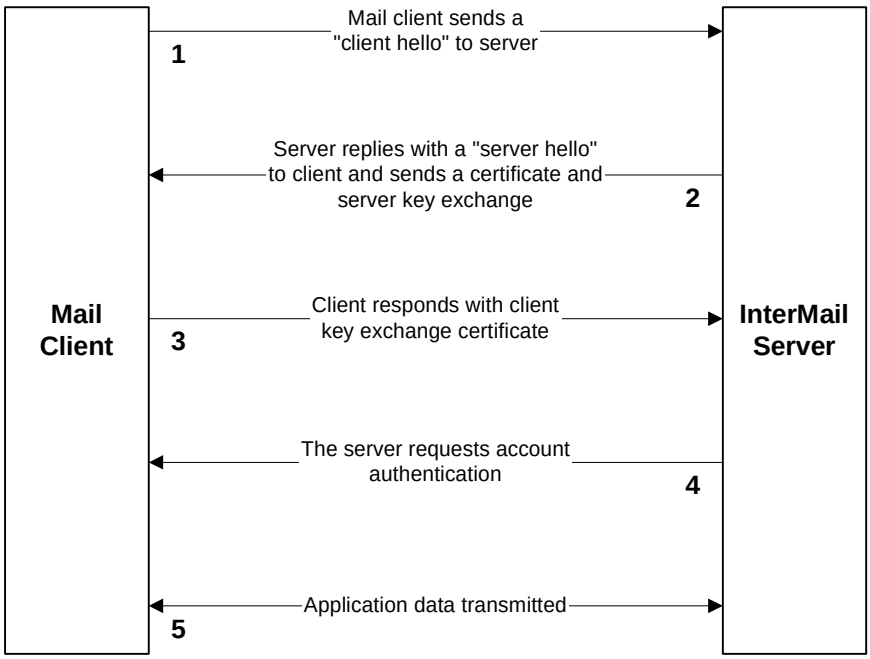

Figure 3 SSL data flow

1. An SSL-enabled e-mail client contacts InterMail on an alternate POP/IMAP port specifically used for SSL transactions.
2. InterMail sends a “Server Hello” message to greet the client and initiate the certificate/key exchange.
3. The client either responds with a message containing the certificate or sends a “no certificate” message and then verifies the exchange.
4. The server requests account authentication and finishes the exchange process.
5. SSL transmits and protects application data.

Connections with SSL Clients

The configuration of SSL allows the assignment of one additional SMTP port, one additional POP port, and one additional IMAP port for secure SSL sessions. When the system detects an SSL-enabled client, the SSL handshake procedure (verification) runs on the defined SSL ports.

To enable SSL:

- Use the `sslPop3Port`, `sslIMAPPort`, and/or `sslSMTPPort` configuration keys. These are the SSL-defined ports that will accept connections from SSL clients. By default, these keys are 995 for POP SSL, 993 for IMAP SSL, and 465 for SMTP SSL. Do not change these port assignments, since SSL clients will expect these assignments.

- Use the `sslTrustedCertPathAndFile` configuration key to specify the file containing the certificate. As shown in SSL data flow, the SSL process involves certificate exchange by client and server. Therefore, a certificate must exist in the InterMail system.
- Set up a file containing an encrypted private key (defined in `sslCertChainPathandFile`) and the password to decrypt this private key (defined in `sslCertPassword`) before SSL operation.
- Finally, you must make certain that the `pref_popssl`, `pref_imapssl`, and `pref_smtpssl` class of service attributes are set.

Note: When a client that is not SSL-enabled connects to InterMail, the ports defined by the `pop3Port`, `IMAPPort`, and `SMTPPort` configuration keys accept the connection instead of the SSL ports.

Transport Layer Security

InterMail provides an additional extension to the SMTP server based on SSL—Transport Layer Security (TLS). TLS provides private, authenticated communication over the Internet to help protect messages from eavesdroppers and attackers. TLS allows an SMTP server/client pair to activate SSL on the normally nonsecure SMTP port.

There are several conditions to satisfy in order to enable TLS in an SMTP session.

- First, as with SSL in general, the connected client must be SSL-compliant.
- Second, the negotiation of data shown in SSL data flow must occur successfully.
- Lastly, the MTA configuration keys `SMTPPort` and `sslSMTPPort` must specify the same port number (typically, port 25).

Configuration Options

The following configuration keys define the behavior of InterMail SSL features:

`sslCacheAgeSeconds`	The lifetime in seconds of an SSL session cache entry. The system deletes entries older than the number of seconds specified.
`sslCacheBucketLen`	The maximum number of entries in each bucket of the SSL session cache. If a bucket reaches the maximum number, the system removes the first entries to make room for new entries.
`sslCacheBucketNum`	The number of buckets in the SSL session cache.

`sslCertChainPathAndFile`	The name of a file containing a PKCS 5 password-encrypted, formatted private key, followed by DER formatted certificates defining the private key and certificate chain for the POP and IMAP servers. The last certificate in the file is the root certificate. "_____Begin" and "_____End" PEM syntax delimits the encrypted private key and certificates. If this configuration key does not exist, or if there are errors reading the file, the system disables POP and IMAP server operation on the secure port.
`sslCertPassword`	The password used to decrypt the server private key specified in `sslCertChainPathAndFile`.
`sslIMAPPort`	The port for secure IMAP server operation. If the key does not exist, secure IMAP server operation is disabled. If this key has a valid, unused port number, the IMAP server operates in secure mode on the specified port.
`sslPop3Port`	The port for secure POP server operation. If the key does not exist, secure POP server operation is disabled. If this key has a valid, unused port number, the POP server operates in secure mode on the specified port.
`sslSMTPPort`	The port for secure SMTP operation. If the key does not exist, secure SMTP operation is disabled. If this key has a valid, unused port number, the MTA operates in secure mode on the specified port.
`sslUseSessionCache`	Option for using the SSL session cache.

Controlling POP/IMAP Access by Location

It is desirable for Internet Service Providers (ISPs) to limit user access to their POP and IMAP servers, based on user location. By programming their routers to block access to all unknown users (for example, anyone connecting from the Internet), ISPs can completely deny all unknown access.

However, there are times when an ISP may want to grant users access to their POP and IMAP servers, even if retrieval is being attempted from an unknown location. InterMail offers a feature that supports this. Using specific paths through the ISP's routers, traffic coming from an unknown location will arrive at one interface, while traffic coming from a trusted location (the internal network) will arrive at a different interface. The InterMail system can identify trusted and unknown locations by the interface through which the connection is made.

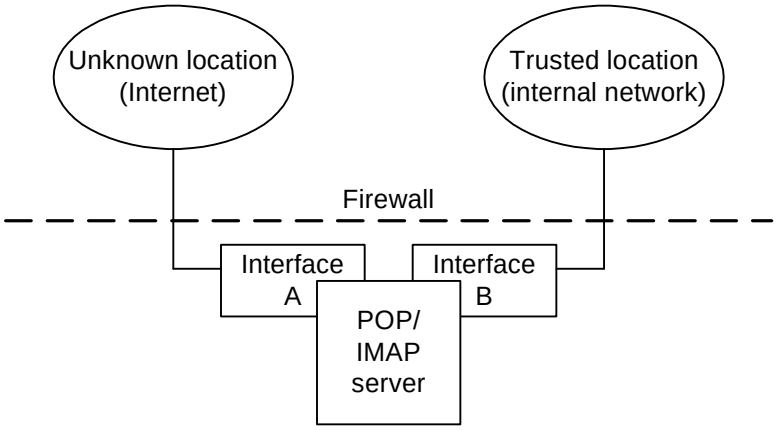

Figure 4 Trusted locations vs. unknown locations

Configuring Trusted Interfaces

In order to limit user connections to the ISP's internal network, each server host must have a list of interfaces that are considered trusted. The `/*/common/trustedInterfaces` configuration key allows the ISP to list a set of trusted interface addresses. Values are specified in the form of IP addresses. For example:

```
/uranus/common/trustedInterfaces:
                    [10.2.7.49]
                    [10.2.7.50]
                    [10.2.5.51]
```

Any IP address that does not appear in the `/*/common/trustedInterfaces` configuration key is considered untrusted.

Controlling User Access to Designated Interfaces

Class-of-service options give the ISP further flexibility when choosing the locations from which users can access their mailboxes. Users can be granted permission to connect to the ISP's server from any location, from trusted locations only, or not at all.

The following class-of-service attributes specify the login permissions of an account:

- `popaccess` defines access restrictions for standard POP retrieval.
- `popsslaccess` defines access restrictions for POP retrieval with SSL.
- `imapaccess` defines access restrictions for IMAP.

These class-of-service attributes can have the following values:

`all`	Users can log in from any location
`trusted`	Users can log in only from interfaces that are specified in the `/*/common/trustedInterfaces` configuration key.
`none`	Users cannot log in from any location.

The SSL option for the POP server is listed independently, allowing specified access for encrypted locations.

For example, suppose that John Doe has the following account settings:

- For `popaccess: trusted`
- For `popsslaccess: all`
- For `imapaccess: none`
- John Doe is allowed standard POP access from trusted locations only, POP with SSL access from any location, and no IMAP access.

Note: These options can be set on either on a class-of-service level or an account level. If they are set on both levels, the account setting takes precedence.

When a user with a "trusted" account setting tries to connect to a corresponding POP or IMAP service, the server compares the server host interface on which the connection arrived to the list of trusted IP addresses. If the IP address is in the list, and the user's access restrictions allow entry from a trusted interface, access is allowed.

Blocking RCPT TO: Harvesting

One of the tools used by senders of junk e-mail is a program to guess valid e-mail addresses. Spammers can do this by connecting to your MTA and running `RCPT TO: <username>` commands. If the MTA accepts the address, the spammer saves that as a valid address for later spamming.

InterMail provides a mechanism to dynamically detect and block `RCPT TO:` harvesters. Using this feature you can track IP addresses that are the source of bad `RCPT TO:` commands, on a per-MTA basis. To use this feature, set the `trackRcptHarvesters` configuration key to `true`. However, this key has no effect if the `/*/mta/verifyRCPTs` configuration key is set to `false`.

An IP address is labeled as a `RCPT TO:` harvester if:

- The number of bad `RCPT TO:` commands from an IP address exceeds the number specified in the `rcptHarvesterCount` configuration key.
- The number of bad commands occur within the window of time specified in the `rcptPotentialHarvesterTTLMinutes` configuration key.

If an IP address is labeled as a `RCPT TO:` harvester, the following action is taken:

- The current connection from that IP address is dropped.
- All further connections from that IP address are dropped for the period specified in the `rcptHarvesterTTLMinutes` configuration key.

This action is equivalent to dynamically adding that IP address to the `dropTheseIPs` list for a specified period.

You must be cautious while implementing this feature or you may end up blocking legitimate mail. Set the `rcptHarvesterCount` and `rcptPotentialHarvesterTTLMinutes` configuration keys with care so that you are reasonably sure before labeling an IP as a harvester.

Configuration Options

The following configuration keys define this feature:

`rcptHarvesterCount`	Indicates the number of bad `RCPT TO:` commands from an IP within a specified time that causes that IP to be labeled as a `RCPT TO:` harvester.
`rcptHarvesterTTLMinutes`	Specifies the number of minutes for which connections should be blocked from an IP that is identified as a source of `RCPT TO:` harvesting.
`rcptMaxHarvesters`	Specifies the maximum number of harvesters or potential harvesters that are tracked at one time.
`rcptPotentialHarvesterTTLMinutes`	Specifies the number of minutes that an IP should be tracked as a potential harvester after one bad `RCPT TO:` is received from that IP. ***Note:*** *Since entries are expired every* `rcptPotentialHarvesterTTLMinutes` *minutes, an entry could, on an average, actually live 1.5 times the number specified in* `rcptPotentialHarvesterTTLMinutes`.
`trackRcptHarvesters`	Indicates whether `RCPT TO:` harvesters should be tracked and blocked. This key has no effect if the `/*/mta/verifyRCPTs` configuration key is set to `false`.

6

Mail Routing

InterMail offers various options to control the direction of mail flow and ensure that mail reaches its intended destination. You can use these options to correct incomplete or wrong recipient addresses and domain names and to handle mail for unknown users.

This chapter covers:

- How envelopes and message headers are addressed
- How address completion works
- Setting up and disabling wildcard accounts
- Rewriting headers and domains for incoming mail
- Rerouting outgoing mail, and rewriting headers and domains for outgoing mail
- Enabling delivery status notifications (DSNs)

Note: For information on proxying, a special form of mail routing that is used primarily during migration from a Post.Office system or Sendmail system to an InterMail system, see the *InterMail Mx Migration Guide*.

Overview

In routine delivery, mail addressed to a known user at a known domain is delivered to its intended recipient exactly as addressed. For example, if a message arrives at the Message Transport Agent (MTA) addressed to `john.doe@minorcorp.com`, the InterMail system checks its directory to see whether `minorcorp.com` is a domain it knows about and whether `john.doe` is a known user at `minorcorp.com`.

The system then does one of the following:

- If both conditions are met, it delivers the message as addressed.

- If `minorcorp.com` is not a domain known to InterMail, the system does a domain name system (DNS) lookup to find a remote domain by that name and then attempts to deliver the message there.
- If `minorcorp.com` is a domain known to InterMail, but `john.doe` does not have an account there, the system bounces the message back to its sender with a note saying that John Doe is not a known user at `minorcorp.com`.

This is InterMail's behavior under routine conditions; however, there are a number of circumstances under which messages that could otherwise be delivered successfully bounce or are misrouted if InterMail tries to deliver them exactly as addressed by their senders. For example, without special routing instructions, a message whose address is incomplete may bounce or be delivered incorrectly.

Envelope and Message Addressing

In order to understand the subject of mail routing, it is first necessary to understand how an InterMail message is addressed. If you are already familiar with this subject, you can skip ahead to "Address Completion" on page 101.

An InterMail message is similar to postal mail in that it consists of both an envelope and the actual message. Both envelopes and messages contain addresses, but their functions are very different.

In the envelope, there are only two types of "addresses": the `RCPT TO:` addresses which are the addresses of the intended recipients, and the `MAIL FROM:` address, which is the address of the message's sender. The InterMail system uses these addresses to deliver messages to their proper recipients, although the person reading the message never sees them.

Note: In order for the InterMail system to deliver a message, the envelope's `RCPT TO:` address must be SMTP-compliant (for example, `john.doe@minorcorp.com` or `john.doe@rome.minorcorp.com`).

The address lines that a reader sees in a message are "headers." These include a `To:` header, which contains the recipient's address, and a `From:` header, which contains the sender's address. A message may also contain several additional headers, including the `Reply To:` header, a `Cc:` header, and a number of others. The headers in a message do not route mail. Their chief purpose is to give the reader important information about who sent the message, who else received a copy, and the subject of the message.

Figure 5 shows how the addresses and headers on an e-mail envelope and message might look if they were postal mail.

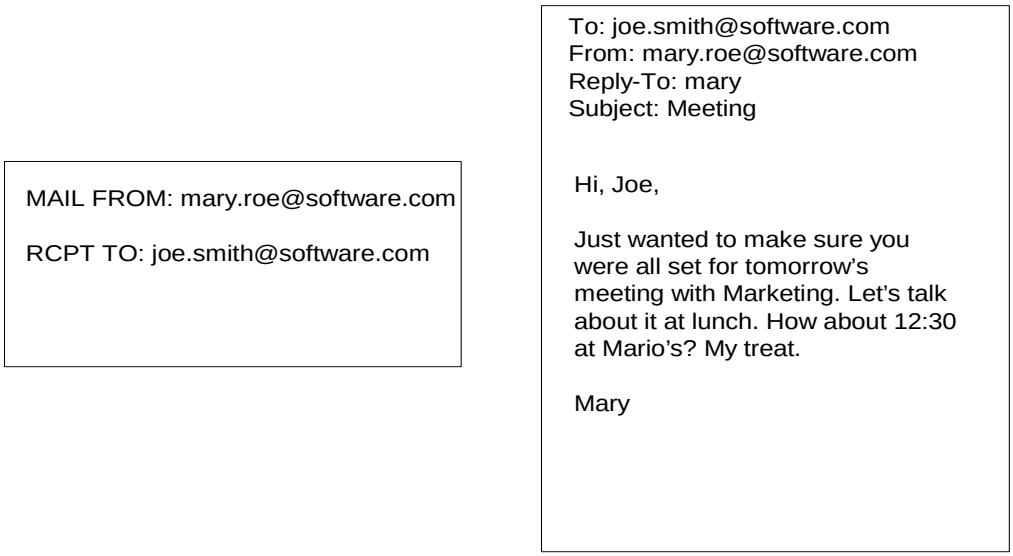

Figure 5 Envelope addresses and message headers

Address Completion

InterMail provides configuration options for completing two types of incomplete addresses and domains:

- Addresses with only partial domain information (incomplete information to the right of the `@` sign). An example of this type of incomplete address is `john.doe@rome`, which consists of just a user and host name.
- Addresses that contain no domain information (omitting information to the right of the `@` sign). An example of this type of incomplete address is `mary` or `mary@`, which consists of just a username.

The ability to complete addresses automatically offers users within the same network the convenience of addressing messages to each other in shorthand form. For example, if Mary Roe and Joe Smith are both users in the `minorcorp.com` domain, Mary can send Joe a message at `joe.smith` instead of having to enter Joe's full address, `joe.smith@minorcorp.com`.

Note: It is the `RCPT TO:` address in the message envelope that is used to deliver the message, not the address in the message header.

Recipient Addresses with No Domains

If an address does not include a domain at all, the system creates an SMTP-compliant address by appending the default domain defined in the `defaultDomain` configuration key. This is the most common type of address completion.

If the value for `defaultDomain` is defined as `minorcorp.com`, then the address `john.doe` or `john.doe@` automatically changes to:

```
john.doe@minorcorp.com
```

As a backup method for address completion, in the rare case that the `defaultDomain` key is missing or its value is undefined (null), InterMail attempts to complete the address by concatenating the values of the `confServHost` and `domainName` configuration keys as follows:

```
<any_user>@confServHost.domainName
```

Where:

`any_user`	Is the username of the intended recipient, such as `john.doe`.
`confServHost`	Is the name of the host on which the Configuration server is running.
`domainName`	Is the domain name used to complete addresses with partial domains.

For example, if the values for `confServHost` and `domainName` are `london` and `megacorp.com`, respectively, InterMail completes John Doe's address as follows:

```
john.doe@london.megacorp.com
```

Of course, there is no guarantee that this address exists or that mail for John Doe will arrive at this address.

Recipient Addresses with Partial Domains

To complete a recipient address that contains only a partial domain, such as `mary.roe@rome`, InterMail uses the value defined in the `domainName` configuration key.

Therefore, for example, if the value for `domainName` is `megacorp.com`, a message addressed to `mary.roe@rome` automatically changes to:

```
mary.roe@rome.megacorp.com
```

Non-Sender Addresses

The `completionMethod` configuration key specifies a completion method for any incomplete addresses other than the envelope's `MAIL FROM:` address:

- The `default` setting completes an incomplete address using the values specified in `defaultDomain` or `domainName`.

- The `bounce` setting does not complete incomplete addresses. Instead, it bounces and returns to its sender any message with an address in any of the following forms:

  ```
  john.doe
  john.doe@
  john.doe@london
  ```

- The `sender` setting completes an incomplete address only if the `MAIL FROM:` envelope address contains a known domain (that is, a domain that the Integrated Services Directory (ISD) specifies as local, non-authoritative, or rewrite). For a discussion of domain types, see Chapter 3.
- If the domain in the `MAIL FROM:` address is known, InterMail completes the incomplete address using the domain name that appears in the `MAIL FROM:` address.

Note: The `sender` option is probably the most useful for an InterMail system that serves multiple domains.

The `sender` option is most useful because it permits users in a domain to perform "lazy" message addressing. For instance, for a mail address in the `software.com` domain, if the `completionMethod` key is set to `sender`, you can send mail to `software.com` domain members without specifying domain and the system will automatically append `@software.com` to the address.

This is particularly useful when a mail provider is handling outsourced mail for many organizations that have their own vanity domain names (which are then configured as local domains within the ISD). Users within each domain (who are apt to be sending primarily to other users in their own organization) can use "lazy addressing" and the addresses will be completed with their own domain.

Sender Addresses

The `canonicalize` configuration key completes an envelope's `MAIL FROM:` address if that address is incomplete. If `canonicalize` is set to `true`, the system completes the `MAIL FROM:` address using the value set in `defaultDomain` or `domainName`, as specified in "Address Completion" on page 101. If `canonicalize` is set to `false`, the system does not complete the address.

Wildcard Accounts

Each local domain can have an optional wildcard account, which is simply a normal e-mail account that receives all mail to non-existent addresses in the domain. This feature allows you to collect in a single account all mail for a particular domain whose destination address does not exist as an account primary address or as an SMTP alias.

For example, supposed that `accordance.com` has no wildcard account. If a customer sends a message to `accordance.com` using an intuitive address such as

sales@accordance.com or techsupport@accordance.com and that address does not exist, the system considers the message undeliverable. In contrast, if accordance.com had a wildcard account, such a message would be delivered there.

Note: You can enable or disable wildcard delivery for an existing local domain at any time. However, a domain can only have one wildcard account at any time.

Setting Up a Wildcard Account

To set up a wildcard account for a local domain:

1. If it does not already exist, create the account to which you want mail for unknown users to go.

   Note: If the volume of mail for unknown users is likely to be light, you may want to make the wildcard account the same as the existing account of the person who will be checking this mail. In contrast, if the volume of mail for unknown users is likely to be heavy, you may want to create a separate account as the wildcard account.

2. Enter:

   ```
   imdbcontrol SetWildcardAccount <DomainName> <PrimarySMTPAdddress>
   ```

Where:

DomainName	Is the name of the local domain for which you are creating the wildcard account.
PrimarySMTPAddress	Is the fully qualified address of the existing e-mail account that will be the wildcard account for this domain.

For example, assume that John Doe, whose existing account is john.doe@accordance.com, has the job of handling mail for unknown users sent to the software.com domain. To create a wildcard account for John, you would enter:

```
imdbcontrol SetWildcardAccount software.com john.doe@accordance.com
```

Note: The wildcard account for a domain may be displayed with the imdbcontrol ListDomains command.

Disabling a Wildcard Account

You can disable a wildcard account at any time. Because a domain can have only one wildcard account at a time, in order to create a new wildcard account you must first disable the existing wildcard account. To disable a wildcard account, enter:

```
imdbcontrol UnsetWildcardAccount <DomainName>
```

Where:

`DomainName`	Is the name of the local domain for which you are disabling wildcard delivery

For example, to disable wildcard delivery for the `software.com` domain, you would enter:

```
imdbcontrol UnsetWildcardAccount software.com
```

Rewriting Incoming Mail

Every message, whether addressed to a local user or destined for a remote recipient, is initially treated as incoming mail when it arrives at the Message Transport Agent (MTA). Only after applying the rules for incoming mail does the system complete additional checks to determine whether the message's final destination is local or remote.

You can set up InterMail to do both domain and header rewriting for incoming mail:

- You use header rewriting to clean up the addresses that readers see in messages, and to hide proprietary origination addresses when necessary.
- You use domain rewriting, which modifies the domain portion of envelope addresses, to reroute mail destined for one domain to another domain.

Header Rewriting

Header rewriting for incoming mail affects message headers only, leaving the `RCPT TO:` and `MAIL FROM:` envelope addresses untouched. No matter which incoming headers the system rewrites, mail is still delivered to the unaltered `RCPT TO:` address specified in the message envelope.

To set up header rewriting, you must specify:

1. Which headers you want rewritten
2. What method you want used to rewrite the headers
3. Under what conditions you want headers rewritten
4. Whether you want original header information saved

Specifying the Headers to Be Rewritten

The first step in rewriting incoming headers is to specify which headers you want eligible for rewriting. There are six header types:

```
To:
Cc:
Bcc:
Reply-To:
From:
Sender:
```

To make headers eligible for rewriting, use `imconfedit` to set the `rewriteHeaderList` configuration key as follows:

```
/*/mta/rewriteHeaderList:     [header_type]
                              [header_type]
```

Example:

```
/*/mta/rewriteHeaderList:     [To:]
                              [Cc:]
                              [From:]
```

Note: A null value in `rewriteHeaderList` means that there will be no header rewriting for incoming messages, regardless of any other settings.

Selecting a Rewriting Method

Once you have selected the headers that will be eligible for rewriting, you are ready to select a rewriting method for these headers. You can choose to rewrite eligible headers using either or both of the following:

- The primary SMTP addresses of local users. This method can be used only for users with accounts (and therefore with SMTP addresses) in the ISD.

  Example:

  Suppose a message for Mary Roe in the `megacorp.com` domain arrives addressed to her alias address:

  `mary@megacorp.com`

  With this method of rewriting, the system rewrite the message's `To:` header with Mary Roe's primary SMTP address, so that it reads:

  `mary.roe@megacorp.com`

- Rules specified in a rewrite domain. This method can be used only if there is an applicable rewrite domain in the ISD.

  Example:

  Suppose you have a rewrite domain whose rule specifies the rewriting of all mail addressed to any user in the `minorcorp.com` domain to that same user in the `megacorp.com` domain, and suppose that a message arrives addressed to:

  `john.doe@minorcorp.com`

With this method of rewriting, the system rewrites the message's `To:` header as follows:

```
john.doe@megacorp.com
```

If you choose to use both of these methods, InterMail implements them as follows:

1. Checks whether there is an account for the user to whom the message is addressed and does one of the following:
   - If there is an account for this user, rewrites eligible headers with the user's primary SMTP account and does no further rewriting.
   - If there is no account for this user, goes on to Step 2.
2. Checks for an applicable rewrite domain in the ISD and does one of the following:
   - If a rewrite domain exists, rewrites the domain portion of eligible headers according to the rule specified in the rewrite domain.
   - If no rewrite domain exists, does no rewriting.

Note: The system may rewrite individual headers within the same message differently. For example, the `To:` header might pass the test for primary rewriting, the `Cc:` header might fail that test but pass the test for domain rewriting. and the `Bcc:` header might fail both tests and not be rewritten at all.

To rewrite eligible headers, you use `imconfedit`:

- To rewrite headers with the primary SMTP addresses of local users, set the value of the `rewritePrimary` configuration key to `true`.
- To rewrite headers using rules defined in a rewrite domain, set the value of the `rewriteDomains` configuration key to `true`.

Note: If the values for `rewritePrimary` and `rewriteDomains` are both `false`, no header rewriting occurs, regardless of which headers `rewriteHeaderList` specifies.

Specifying the Conditions for Header Rewriting

After choosing a method for rewriting eligible headers, you are ready to specify the circumstances under which eligible headers are to be rewritten.

To specify the conditions for header rewriting, you use `imconfedit`:

- To rewrite eligible headers only for messages that have passed through fewer than a specified number of mail servers, set the value of the `rewriteMaxMtaHops` configuration key. The assumption here is that if the message has already passed through the value set in `rewriteMaxMtaHops` (number of MTAs), header rewriting has probably already been performed for the message. For example, if

you want headers rewritten only for messages that have been processed by 2 mail servers or fewer, set the key as follows:

```
/*/mta/rewriteMaxMtaHops: [2]
```

- To rewrite eligible headers only for messages that originate from a sender in a known domain that InterMail hosts, set the value of the `rewriteOnlyLocal` configuration key to `true`. If this key is set to `false`, rewriting of eligible headers occurs regardless of whether InterMail knows the sender.

Saving the Original Header Information

You can also specify whether, after header rewriting, you want the original headers (in `X-Original` format) saved as part of the message.

To save the original headers in the message, use `imconfedit` to set the value of the `rewriteSaveOrig` configuration key to `true`. If the value of this key is `false`, the system will not save the original headers.

Example:

At this point, an example may be useful to illustrate how these header rewriting features work together.

Suppose that you want to rewrite only the `To:` headers for incoming mail. Furthermore, assume that you want to rewrite these headers with the primary SMTP addresses of users, that you want to rewrite eligible headers only when message senders are local, and that you do not want to save the original headers as part of each message.

To achieve these results, you use `imconfedit` to do the following:

1. Set the value of the `rewriteHeaderList` configuration key to `[To:]`.
2. Set the value of the `rewritePrimary` configuration key to `[true]`.
3. Set the value of the `rewriteOnlyLocal` configuration key to `[true]`.
4. Set the value of the `rewriteSaveOrig` configuration key to `[false]`.

Header Rewriting Process Flow

Figure 6 shows the flow of the header rewriting process. The numbered steps indicate the sequence in which InterMail performs the various checks required to rewrite incoming mail headers.

Note: The diagram and steps are carried forward throughout this chapter to show how InterMail's various mail routing features are connected.

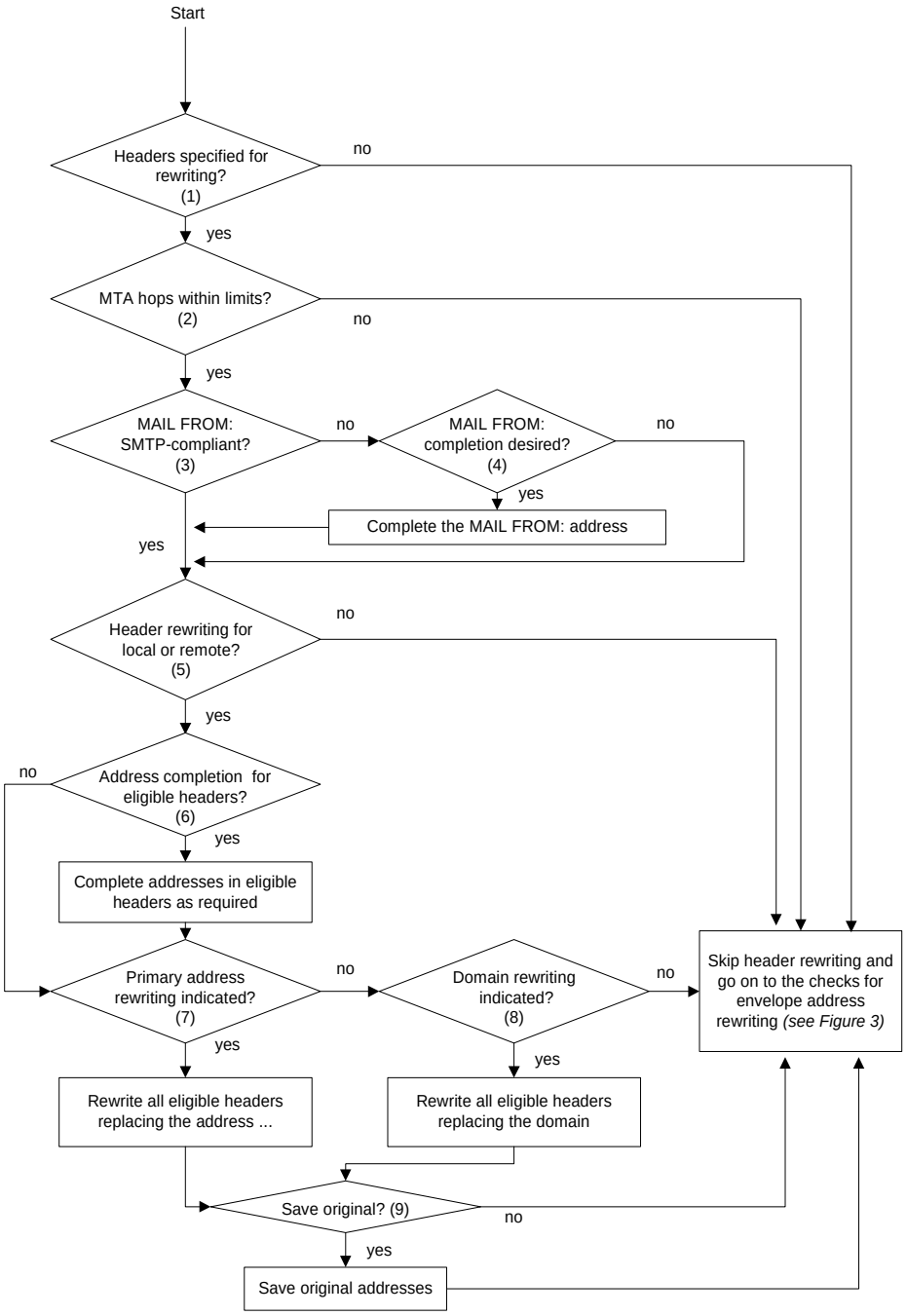

Figure 6 Header rewriting

In this process, the InterMail system:

1. Checks the `rewriteHeaderList` configuration key.
   - If the `rewriteHeaderList` configuration key specifies headers, considers those headers eligible for rewriting and goes on to step 2.
   - If the value of the `rewriteHeaderList` configuration key is null, will not rewrite headers, and skips to step 10.
2. Compares the number of received lines in the message header with the value of the `rewriteMaxMtaHops` key.
   - If the number of received lines is less than or equal to the value of the `rewriteMaxMtaHops` key, goes on to step 3.
   - If the number of received lines exceeds the value of the `rewriteMaxMtaHops` key, will not rewrite headers, and skips to step 10.
3. Checks the `MAIL FROM:` address on the envelope for SMTP-compliant formatting.
   - If the address is SMTP-compliant, continues with the header rewriting operation (step 5).
   - If the address is incomplete, goes to step 4 to determine whether to complete address.
4. Checks the `canonicalize` configuration key.
   - If the value of the `canonicalize` key is `false`, does not perform address completion, and goes on to step 5.
   - If the value of the `canonicalize` key is `true`, completes the `MAIL FROM:` address (typically with the value in the `defaultDomain` configuration key) and then goes on to step 5.
5. Checks the `rewriteOnlyLocal` configuration key.
   - If the value of the `rewriteOnlyLocal` key is `false`, continues with the header rewriting operation (step 6).
   - If the value of the `rewriteOnlyLocal` key is `true`, and the domain in the `MAIL FROM:` address on the message envelope matches one of the known domains specified in the ISD, continues with the header rewriting operation (step 6).
   - If the value of the `rewriteOnlyLocal` key is set to `true`, but the domain in the `MAIL FROM:` address on the message envelope does not match one of the known domains specified in the ISD, does not perform header rewriting, and skips to step 10 (Figure 7).
6. Checks whether addresses in eligible headers (those listed in the `rewriteHeaderList` key) require completion.
   - If the addresses in eligible headers are complete, goes to step 7.

- If the addresses in eligible headers are incomplete, completes them (typically with the domain identified in the `defaultDomain` configuration key), then goes to step 7.

Note: The addresses in some eligible headers may require completion, whereas others may not.

7. Checks the `rewritePrimary` configuration key.
   - If the value of the `rewritePrimary` key is `false`, goes on to step 8.
   - If the value of the `rewritePrimary` key is `true`, examines all eligible message headers (those listed in the `rewriteHeaderList` key) for primary address rewriting, and then:
   - If the address in an eligible header matches a primary address or SMTP alias address in any account in the ISD, replaces the address in the eligible header with the primary address for the matching account, and skips to step 9.
   - If the address in an eligible header does not match any address in any account in the ISD, goes on to step 8.

Note: Some eligible headers may pass this test for rewriting, whereas others may fail this test and undergo further rewriting tests.

8. Checks the `rewriteDomains` configuration key.
   - If the value of the `rewriteDomains` key is `false`, does not perform domain writing, and skips to step 10.
   - If the value of the `rewriteDomains` key is `true`, examines all eligible message headers (those listed in the `rewriteHeaderList` key) for domain rewriting, and then:
   - If the address in an eligible header includes a domain that matches a rewrite domain established in the ISD, overwrites the domain in the eligible header with the replacement value for the rewrite domain, and goes on to step 9.
   - If the address in an eligible header includes a domain that does not match any of the rewrite domains established in the ISD, does not perform domain rewriting, and skips to step 10.

Note: The process considers various headers individually. As a result, it may rewrite some and not rewrite others.

9. Check the `rewriteSaveOrig` configuration key. This key determines whether or not to save a record of the original header information.
   - If the value of the `rewriteSaveOrig` key is `false`, does not save any of the original header information, and goes on to step 10.

- If the value of the `rewriteSaveOrig` key is `true`, adds a new `X-Header` that contains the original header information, and goes on to step 10.

All incoming mail is subject to checks for both header and domain rewriting. Once the checks for header rewriting finish, the checks for domain rewriting begin, as discussed in the next section.

Domain Rewriting Process Flow

Domain rewriting involves no configuration keys; it happens automatically, whenever a rule specified in one of the ISD's rewrite domains matches the domain portion of an incoming message's `RCPT TO:` address. When such a match occurs, the system rewrites the domain portion of the envelope's `RCPT TO:` address and reroutes the message to the recipient in the new domain. For a discussion of rewrite domains, see Chapter 3.

Figure 7 shows the flow of the domain rewriting process, which starts with step 10, after the checks for header rewriting.

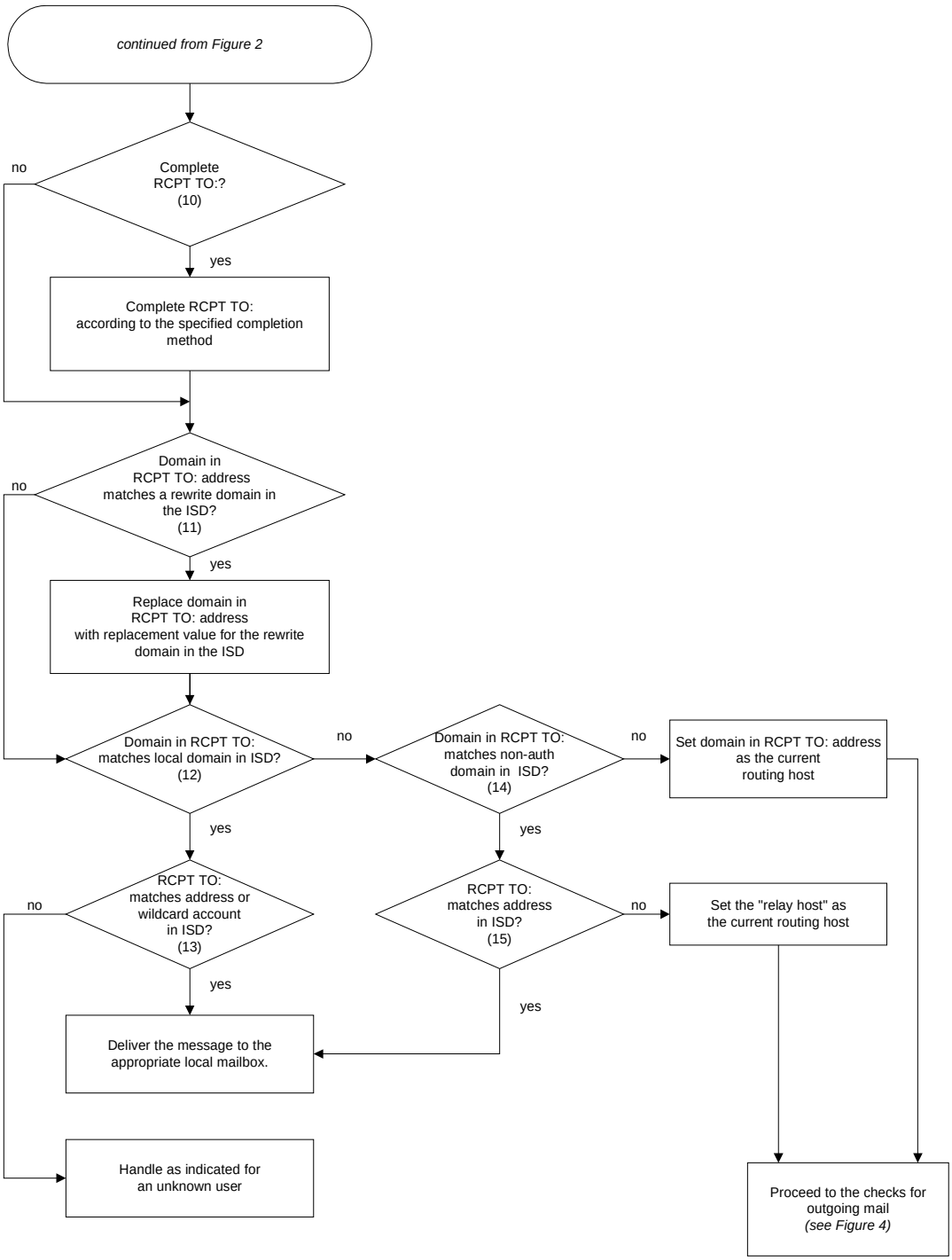

Figure 7 Domain rewriting for incoming mail

After the header rewriting process, the InterMail system:

10. Checks the `RCPT TO:` envelope address to see if it is complete.
    - If the address is complete, goes to step 11.
    - If the address is not complete, completes it (typically with the value in the `defaultDomain` configuration key), and goes to step 11.
11. Checks the complete `RCPT TO:` address on the envelope against the rewrite domains identified in the ISD.
    - If the domain in the address matches a rewrite domain established in the ISD, overwrites the domain in the address with the replacement value indicated for the rewrite domain, and goes to step 12.
    - If the domain in the address on the envelope does not match any of the rewrite domains established in the ISD, does not perform envelope address rewriting, and goes to step 12.

Note: At this point the `RCPT TO:` addresses for all envelopes destined for local delivery include a domain that is either local or non-authoritative.

---

12. Checks the `RCPT TO:` address on the envelope against the local domains identified in the ISD.
    - If the domain in the address on the envelope matches a local domain established in the ISD, looks up the address in the ISD, and goes to step 13.
    - If the domain in the address on the envelope does not match any of the local domains established in the ISD, skips to step 14.
13. Checks the `RCPT TO:` address on the envelope against the account addresses in the ISD.
    - If the address on the envelope is an exact match for an address in the ISD, delivers the message to the appropriate local mailbox (the mailbox associated with the account with the matching address).
    - If the address on the envelope does not match any of the addresses in the ISD, but there is a wildcard account specified for the domain in the address, delivers the message to the wildcard account.
    - If the address on the envelope does not match any of the addresses in the ISD, and there is no wildcard account specified for the domain in the address, considers the recipient an unknown user, and handles the message according to the instructions in the `Error-Action/acctInvalidUser` configuration key.
14. Checks the `RCPT TO:` address on the envelope against the non-authoritative domains identified in the ISD.
    - If the domain in the address matches a non-authoritative domain in the ISD, goes to step 15 in “Rerouting and Rewriting Process Flow” on page 120.

- If the domain in the address does not match any of the non-authoritative domains in the ISD, sets the routing host using the value of the domain in the `RCPT TO:` address, and skips to step 16 (Figure 8) in "Rerouting and Rewriting Process Flow" on page 120.

15. Checks the `RCPT TO:` address on the envelope against all addresses stored in the ISD (primary or alias).
    - If the address matches an address in the ISD, delivers the message to the local mailbox associated with that account.
    - If the address does not match any of the addresses in the ISD, sets the routing host to the value of the current relay host sets the routing host to the value of the current relay host associated with the non-authoritative domain, and skips to step 16 (Figure 8).

Note: The system establishes the routing host when it has determined that the message is outgoing mail (that is, mail destined for delivery to an external mail host). The routing host identifies the external host this message should pass to and is independent of the domain contained in the `RCPT TO:` address on the message envelope.

At this point all local delivery is complete, and the system considers any remaining messages to be outgoing mail. Outgoing mail has a separate sequence of routing and rewriting checks, as discussed in the next section.

Rerouting and Rewriting Outgoing Mail

Outgoing mail is mail whose final destination is a remote mail server. You can set up InterMail to reroute outgoing mail and to rewrite headers and domains for outgoing mail.

The Mail Routing Table

Normally the system routes outgoing mail by using the MX records in DNS or by delivering a message to the hostname specified in the envelope address. Alternatively, the SMTP mail routing table offers another routing option that can be used to override normal delivery.

Entry Syntax

A single entry in the mail routing table consists of one required element and two optional elements, each with a different routing or rewriting function for outgoing mail:

- The required routing host element (`<rtg.host>:<new.rtg.host>`) routes mail from one host to another, without changing envelope addresses or message headers.

Note: <rtg.host> is a "host" as defined for a RCPT TO: route in RFC 821. It may actually be a mail domain such as software.com. Likewise, <new.rtg.host> may also be a mail domain.

Also, if the routing component of the address (again, according to the RFC 821 addressing model) is present, the mail routing table checks will not be performed. For example, in RFC 821, if route information was in the form of @ONE,@TWO:JOE@THREE, (where ONE, TWO, and THREE are "hosts", or mail domains), then when InterMail received a message that was to be relayed and there was route information before the actual mailbox name (in the above example, ONE and TWO) in the RCPT TO: address, it would bypass the mail routing table and use the routing information in the RCPT TO: address.

For instance, if the value of mailRoutingTable were [software.com:example.com] and the DNS MX record for example.com pointed to smtp.example.com, mail addressed to joe.smith@software.com would actually be relayed to smtp.example.com.

- The optional header rewriting element ([header-rewrite]) specifies header rewriting for outgoing mail.
- The optional domain rewriting element ([rewrite-domain=<new.domain>]) specifies domain rewriting for outgoing mail.

Note: Domain rewriting for outgoing mail has nothing to do with rewrite domains in the ISD, which govern rewrites for incoming mail. Only the mail routing table specifies domain rewrites for outgoing mail.

Entries in the mailRoutingTable configuration key have the following syntax:

```
<rtg.host>:<new.rtg.host> [header-rewrite] [rewrite
domain=<new.domain>]
```

Where:

<rtg.host>	Specifies the initial destination (the routing host (or mail domain) to which the outgoing message initially goes), or "default". If "default" is specified, all outgoing messages (that have not matched an entry in mailRoutingTable prior to this entry) will be routed to <new.rtg.host>. "Default" should be specified only in the case where it is desired that all outgoing mail be processed through some sort of gateway or firewall.

`<new.rtg.host>`	Specifies the destination to which the outgoing message will be rerouted.
`header-rewrite`	Optionally, requests header rewriting for outgoing mail. This affects only message headers.
`rewrite-domain`	Optionally, requests domain rewriting.
`<new.domain>`	If the domain is being rewritten, specifies the new domain name that will replace the original in the `RCPT TO:` address.

Example:

The following entry specifies that any message that the system would normally deliver to the host `rome.megacorp.com` will instead go to a machine called `gateway.com`:

```
/*/mta/mailRoutingTable: [rome.megacorp.com:gateway.com]
```

Entry Order

The mail routing table can have multiple entries, which are read sequentially from top to bottom. For example:

```
/*/mta/mailRoutingTable: [rome.megacorp.com:gateway.com]
[*.megacorp.com:internal_server.com][default:firewall.com]
```

Each of the above entries has a different effect:

- The first entry affects only messages destined for the host `rome.megacorp.com`. It reroutes these messages to another host called `gateway.com`.
- The second entry uses a wildcard (`*`) for pattern matching. It reroutes all mail destined for `<any_host>.megacorp.com` to `internal_server.com`. Thus, for example, mail sent to `venice.megacorp.com` or `paris.megacorp.com` will go to `internal_server.com`.

  Note: This entry would not reroute mail sent to just `megacorp.com`; the "`*.`" portion of the entry means that some host name must appear before `megacorp.com`.

- The third entry uses `default` to reroute any message for any host to the host `firewall.com`. There is no pattern matching of any kind.

Taken together, these entries give InterMail the following rerouting instructions:

1. First, reroute all mail destined specifically for `rome.megacorp.com` to `gateway.com`.
2. Next, reroute messages destined for any other named `megacorp.com` host (that is, `*.megacorp.com`) to `internal_server.com`.

3. Finally, reroute mail destined for any other host (regardless of the host name) to `firewall.com`.

Because the system reads entries in the mail routing table from top to bottom and uses the first possible match it finds, it is important for you to organize the entries from most specific to most general. Table entries in a different sequence may yield undesired results.

For example, suppose the `RCPT TO:` address in an outgoing message is `joe@megacorp.com` (the routing host portion being `megacorp.com`). To determine how to route this message, the InterMail system starts by looking at the first entry in the mail routing table. Since the address in the message does not match `rome.megacorp.com`, the system ignores this entry and moves down to the next entry in the table.

The address in the message does not match the host name in the second table entry either—there is nothing in `joe@megacorp.com` to substitute for the wildcard (`*`) in `*.megacorp.com`—so the system also ignores this line and moves on to the final entry in the mail routing table.

The final entry does create a successful match, since `default` specifies any possible host. InterMail therefore reroutes the message for `joe@megacorp.com` to the `firewall.com` host machine.

In contrast, suppose that the order of the entries in the table were changed:

```
/*/mta/mailRoutingTable: [*.megacorp.com:internal_server.com]
[rome.megacorp.com:gateway.com][default:firewall.com]
```

In this case, messages destined for `rome.megacorp.com` will never be rerouted to `gateway.com`. Because `*.megacorp.com` is the first entry in the table, the system reads that entry first, and all messages with matches for `*.megacorp.com`, including `rome.megacorp.com`, go to `internal_server.com`.

Similarly, if the first entry in the table were `[default:firewall.com]`, the system would read the `default` entry first, reroute every outgoing message to `firewall.com`, and never read any further entries in the table.

Note: Setting the `mailRoutingHost` configuration key has the same effect as specifying a `default` entry in the mail routing table. If both the `mailRoutingHost` key is set and a `default` entry exists in the mail routing table, the `default` entry in the mail routing table will be used

---

If the initial domain does not match any of the entries in the mail routing table and the `mailRoutingHost` configuration key is not set, DNS will be used to determine the delivery host for the message.

Header Rewriting

Rerouting mail changes its destination from one host to another but does nothing to rewrite message headers or the domain portion of envelope addresses. To rewrite message headers for outgoing mail, you must complete two additional steps:

- Add the `header-rewrite` option to each mail routing table entry for which you want header rewriting.
- Specify the headers that are to be eligible for rewriting.

Note: Rewriting of an outgoing header can occur only if the eligible header is for an account that matches an address in the ISD and there is a forwarding address for that account.

Adding the Header-Rewriting Option

In the mail routing table, you add the `header-rewrite` option after the routing element in each entry for which you want header rewriting. For example, if you want the eligible headers rewritten for all messages rerouted from `rome.megacorp.com` to `gateway.com`, you would enter the following line in the mail routing table:

```
/*/mta/mailRoutingTable: [rome.megacorp.com:gateway.com
header-rewrite]
```

If you want headers rewritten without rerouting mail, simply create an entry in the mail routing table that routes mail from a named host to itself, and include the `header-rewrite` option. For example:

```
/*/mta/mailRoutingTable: [gateway.com:gateway.com header-rewrite]
```

Specifying the Headers to Be Eligible for Rewriting

You can make any of the following headers eligible for rewriting:

```
To:
Cc:
Bcc:
Reply-To:
From:
Sender:
```

To make headers eligible for rewriting, use `imconfedit` to set the `rewriteGatewayHeaderList` configuration key as follows:

```
/*/mta/rewriteGatewayHeaderList:[header_type]
                               [header_type]
```

Example:

```
/*/mta/rewriteGatewayHeaderList:  [To:]
                                  [Cc:]
                                  [From:]
```

The specified headers are now eligible for rewriting for mail routing table entries that contain the `header-rewrite` element.

Domain Rewriting

Using domain rewriting for outgoing mail is similar to using the ISD rewrite domains for incoming mail. However, in the case of outgoing messages, InterMail does not

check the ISD for rewrite domain values. Instead, you specify the rewrite domain values in the mail routing table.

Domain rewriting for outgoing mail changes the `RCPT TO:` address in the envelope, but it does not override the rerouting instructions specified in the `<rtg.host>:<new.rtg.host>` portion of the mail routing table entry. Domain rewriting also has no effect on either an envelope's `MAIL FROM:` address or the message headers.

To specify domain rewriting for outgoing mail, you add the `[rewrite-domain=<new.domain>]` option to each mail routing table entry for which you want domain rewriting.

For example, to rewrite `RCPT TO:` addresses to include a different domain name for outgoing messages being rerouted from `rome.megacorp.com` to `gateway.com`, you would enter the following line in the mail routing table:

```
/*/mta/mailRoutingTable: [rome.megacorp.com:gateway.com
rewrite-domain=megacorp.com]
```

To specify domain rewriting for outgoing mail without rerouting this mail to another host, simply create an entry in the mail routing table that routes mail from a named host to itself, and include the domain-rewriting option. For example:

```
/*/mta/mailRoutingTable:
[minorcorp.com:minorcorp.com rewrite-domain=megacorp.com]
```

Rerouting and Rewriting Example

Suppose that you want to reroute all outbound messages for the `minorcorp.com` host to the `megacorp.com` host, while simultaneously rewriting the `To:`, `Cc:`, and `From:` message headers and the domain portion of the envelopes' `RCPT TO:` addresses.

To achieve these results, you use `imconfedit` to do the following:

1. Create the following entry in the mail routing table:

   ```
   /*/mta/mailRoutingTable
   [minorcorp.com:megacorp.com header-rewrite
   rewrite-domain=megacorp.com]
   ```

2. Set the `rewriteGatewayHeaderList` configuration key as follows to make the `To:`, `Cc:`, and `From:` outbound message headers eligible for rewriting:

   ```
   /*/mta/rewriteGatewayHeaderList: [To:]
                                    [Cc:]
                                    [From:]
   ```

Rerouting and Rewriting Process Flow

Figure 8 shows the flow of the various checks required for rerouting and rewriting headers for outgoing mail. This process starts with step 16, after the checks for domain rewriting for incoming mail.

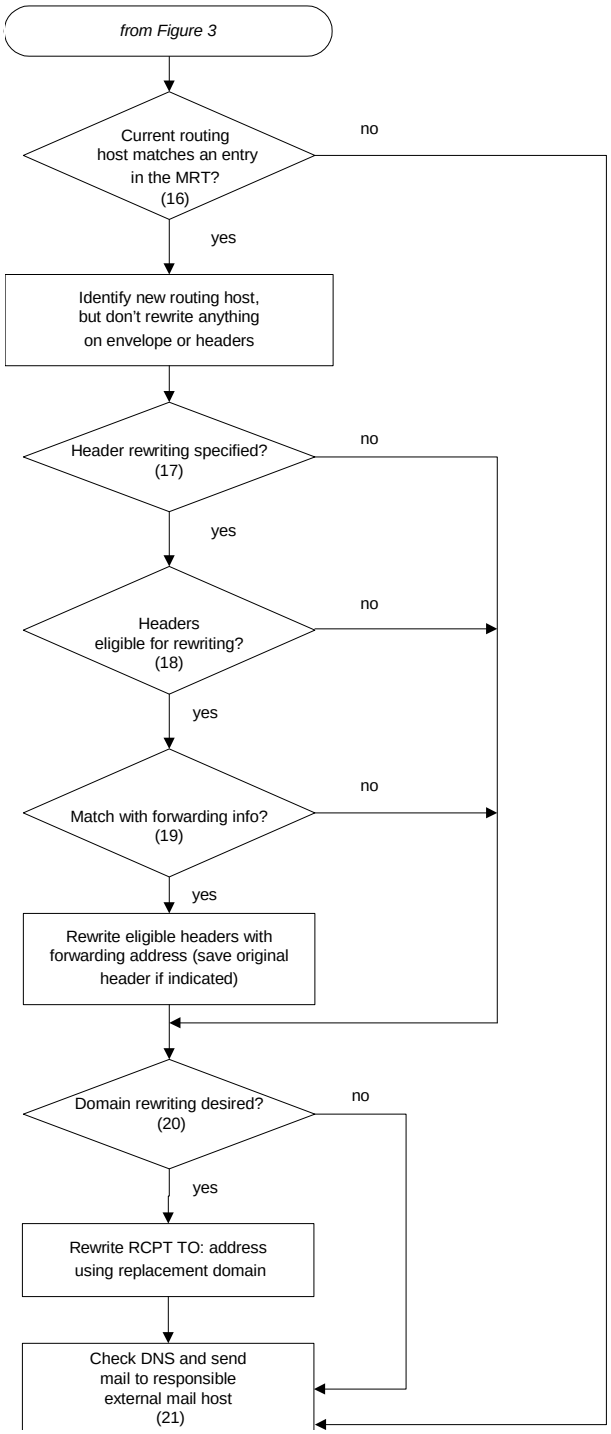

Figure 8 Rerouting and rewriting for outgoing mail

After the domain rewriting process, the InterMail system:

16. Checks the routing host against the `<rtg.host>` portion of sequential entries in the mail routing table.
    - If it does not find a match, does not rewrite the outgoing message, and skips to step 21.
    - If if finds a match, changes the destination of the message to the value in the `<new.rtg.host>` portion of the mail routing table entry, and goes on to step 17.

    Note: The system reads the entries in the mail routing from top to bottom and uses the first matching entry.

17. Checks the selected entry in the mail routing table to see if it includes the `header-rewrite` option.
    - If the entry does include the `header-rewrite` option, goes on to step 18.
    - If the entry does not include the `header-rewrite` option, skips to step 20.
18. Checks message headers against the entries in the `rewriteGatewayHeaderList` configuration key, which define the outgoing mail headers that are eligible for header rewriting.
    - If none of the headers in the message are listed in the `rewriteGatewayHeaderList` key, skips to step 20.
    - If one or more of the headers in the message are listed in the `rewriteGatewayHeaderList` key, goes on to step 19.
19. Checks the address in each message header that is eligible for rewriting against the address entries in the ISD.
    - If there is no match for a header, does not rewrite that header, and goes on to step 20.

- If the address in an eligible header matches an address in the ISD, examines the associated account record for forwarding information, and then checks for a forwarding address:
    - If the associated account includes a forwarding address, replaces the address in the eligible header with the forwarding address in the ISD account record; if the `rewriteSaveOrig` configuration key is `true`, records the original header information in a new `X-Header`; and goes to step 20.
    - If the associated account does not include a forwarding address, does not rewrite the address in the eligible header, and goes to step 20.

    Note: The system considers the various headers individually. As a result, it may rewrite some but not others.

20. Checks the selected entry in the mail routing table to see whether to perform domain rewriting.
    - If the entry does not include the `rewrite-domain` option, skips to step 21.
    - If the entry includes the `rewrite-domain` option, replaces the domain in the `RCPT TO:` address on the message envelope with the value in the `<new.domain>` portion of the mail routing table entry, and goes on to step 21.
21. Requests the DNS records associated with the appropriate external destination, and hands the message off to the external mail host identified.

Delivery Status Notification (DSN)

InterMail supports delivery status notification (DSN). A DSN is a special notice delivered to e-mail senders on other systems that also support DSN. Such a notice can advise senders of either of the following:

- Mail delivery was successful.
- Mail delivery has failed or is delayed.

The system does not send DSNs on a per-user basis. Rather, it sends them automatically for all InterMail accounts, and senders receive them if their mail clients and service providers are DSN-enabled.

Because DSNs are far more detailed than ordinary bounce notices, they can help companies that maintain large mailing lists keep their lists up-to-date. For example, they identify the result of every transaction involved in a delivery attempt to each recipient. Knowing exactly where a delivery attempt failed can help a list administrator determine whether the problem was due to an expired e-mail address.

You enable or disable DSNs for particular conditions. For example, you might enable DSNs for messages that are not delivered because of bad delivery information.

You enable DSNs by using `imconfedit` to set the value of one or more of the following `Error-Action/mtaMessage` keys:

This value:	Causes the system to:
`return`	Send a DSN to the requesting client
`log`	Log the error
`hold`	Disable DSN for this condition

Example:

```
/*/mta/Error-Actions/mtaMessageDelivered: [return]
                                          [log]
```

You can enable DSNs for the following conditions:

To enable DSNs for a message that:	Set a value of `return` in this configuration key:
Has bad delivery information	`Error-Actions/mtaMessageDelivered`
Has been deferred and queued longer than the configured limit	`Error-Actions/mtaMessageQueuedTooLong`
Reached its destination and was forwarded to at least two mailboxes or recipients	`Error-Actions/mtaMessageExpanded`
Was rejected by the receiving SMTP server	`Error-Actions/mtaMessageRejected`
Was relayed to a machine that does not handle DSNs	`Error-Actions/mtaMessageRelayed`
Is larger than the size the SMTP server can accept	`Error-Actions/mtaMessageTooLarge`

7

Managing Mail in Process

Mail in process is mail that a Message Transport Agent (MTA) has received but not yet delivered to its intended recipients. While mail is in process, temporary message storage may be necessary.

This chapter explores the concept of mail in process and covers the following topics:

- Why temporary storage of mail is necessary
- The location and organization of temporary mail storage
- Automatic and manual handling of deferred mail
- Mail queuing options
- Mail throttling and how to implement it

Note: For a discussion of persistent mail storage and related mailbox management, see Chapter 8.

Types of Mail in Process

The MTA examines all incoming mail to determine how to handle it (such as by delivery, forwarding, or automatic response). For efficiency reasons, a significant percentage of incoming mail is processed entirely in memory. However, some circumstances require that mail be stored temporarily.

Temporary disk storage of mail in process may be necessary for any of the following reasons:

Temporarily Stored Mail

- A message exceeds configured limits on size, number of recipients, or the amount of time required to deliver it.

Local Deferred Mail

- The Message Store Server (MSS) or Integrated Services Directory (ISD) is temporarily unavailable.
- A message is received for an account whose mailbox is over quota and the configuration key `/<host>/mta/bounceOnQuotaFull` is set to `false`.
- Throttling of mail delivery to the message stores is in effect.

Remote Deferred Mail

- A remote mail host is temporarily unavailable.

Sidelined Mail

- A message is sidelined as suspected junk mail and requires your examination before you decide whether to deliver or bounce it. This is due to a system-wide or user SIEVE filter.

Mail Held Due to Errors

- Something in the message itself causes an error that prevents delivery.
- Any condition where the `mta/Error-Action` is set to `hold` will cause the message to be treated as mail having an error.

The following sections explores each of these areas in detail.

Temporarily Stored Mail

If a message is large, addressed to a large number of recipients, or takes longer than a configured time to deliver, mail may be temporarily stored before delivery.

By writing such messages to disk while delivery is still in process, the MTA can safely signal successful receipt to the sending server before delivery is fully complete, thus reducing the possibility of servers' timing out.

There are three options that control when the system saves mail in process to disk. You can specify:

- The number of seconds the MTA should wait for message delivery. If this time period is exceeded, the message is written to disk.
- The maximum size (in kilobytes) a message can be without being written to disk.
- The maximum number of recipients a message can have without being written to disk.

If a message exceeds any one of these limits, the system saves it to disk during processing.

Maximum Delivery Time

The `timeoutServerDelivery` configuration key controls the maximum in-memory delivery time, specifying the number of seconds the MTA will hold a message in RAM for message delivery before writing a message to disk.

If the time required to deliver a message exceeds the limit defined in this key, the system saves the message to disk while the delivery process continues without interruption from memory.

For example, if the value of the `timeoutServerDelivery` key is 3, a client must wait no more than 3 seconds for delivery. If delivery cannot be completed in that time, the system stores the message temporarily and signals acceptance of the message to the sending server.

If the value of the `timeoutServerDelivery` configuration key is set too high, users may perceive a slow response time.

If the value of the `timeoutServerDelivery` configuration key is set too low, excessive disk input and output may negatively affect server throughput. Increasing the number of disks used for the spool area may alleviate this problem.

Excessive message delivery times may be caused by either slow processing on the MTA, or slow response from one of more MSS servers in the system.

Note: The maximum recommended value for `timeoutServerDelivery` is 5 seconds.

Maximum Message Size

The `maxDirectKB` configuration key sets the maximum size in kilobytes for a message in memory. The system secures any message larger than this size to disk, signals acceptance of the message to the sending client, and continues delivery normally from memory.

The value of this setting is a function of the amount of RAM on the MTA. If the value of the `maxDirectKB` configuration key is set too high, excessive memory swapping may occur.

Maximum Number of Recipients

The `maxDirectDelivery` key limits the number of recipients for a message that can be delivered directly from memory. The MTA writes any message with more than this number of recipients to disk before attempting delivery.

For example, if the value of the `maxDirectDelivery` key is 20, the system saves to disk any message addressed to more than 20 recipients, although the actual delivery process continues from memory.

Figure 9 shows how InterMail processes temporarily stored mail.

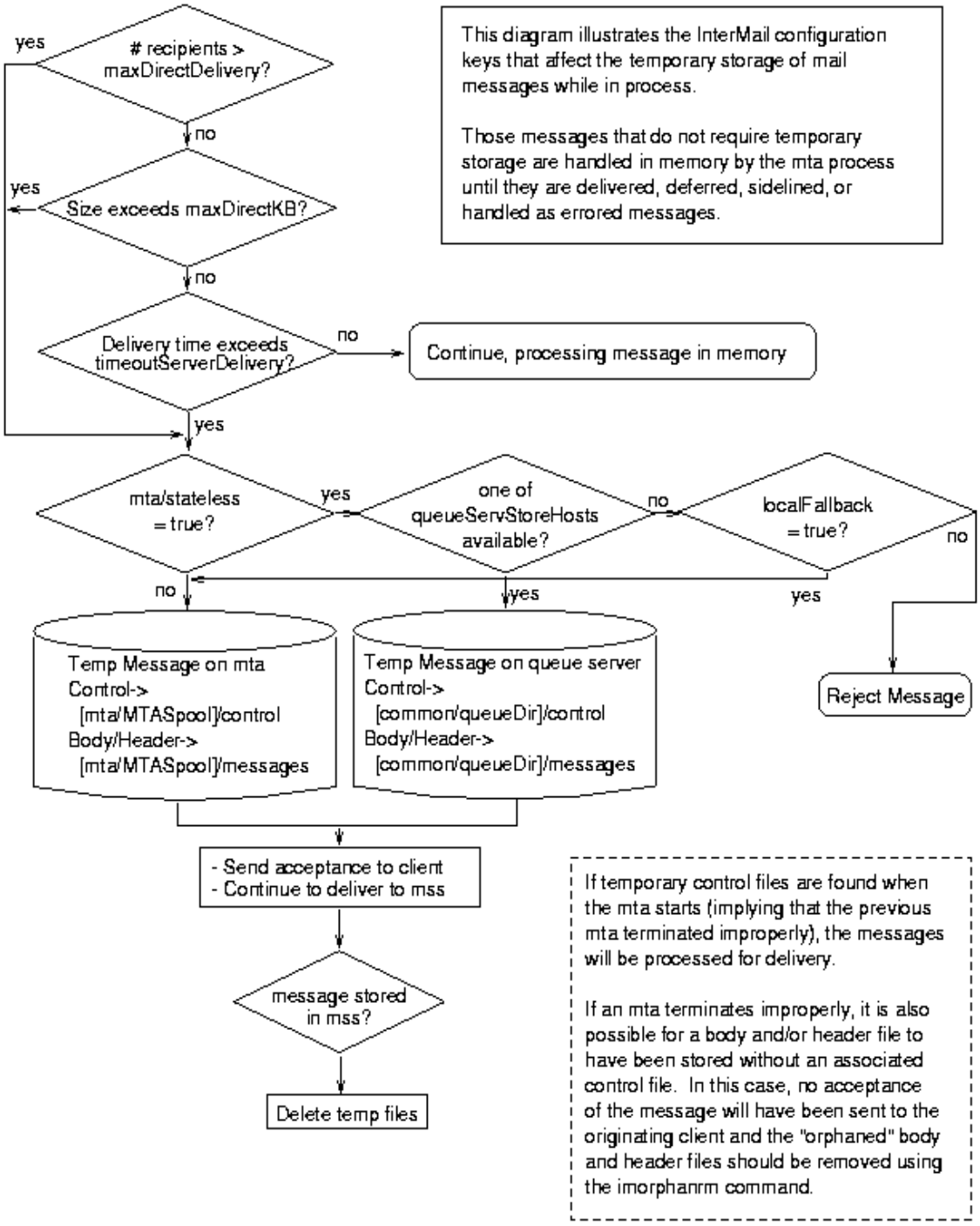

Figure 9 Temporarily stored mail

Local Deferred Mail

Occasionally, there may be a delay in local mail delivery because the MSS or ISD is temporarily unavailable. Temporary message storage is necessary until the connection is available again, at which time the MTA automatically re-attempts delivery.

The `deferProcessInterval` configuration key sets the number of seconds between attempts at redelivery for messages deferred because the MSS or ISD is temporarily unavailable. For example, if the value of the `/*/mta/deferProcessInterval` key is 600, the MTA re-attempts delivery every 10 minutes (600 seconds).

In addition, a message that is received for an account whose mailbox is over quota and the configuration key `/<host>/mta/bounceOnQuotaFull` is set to `false` can cause a local deferral of mail. The delivery will be re-attempted at regular intervals as defined by the `/<host>/mta/deferProcessInterval` configuration key.

When a message is deferred for this reason, a `MsLimitTotalSize` warning log message will be written to the `mta.log` file.

Figure 10 shows how InterMail processes local deferred mail.

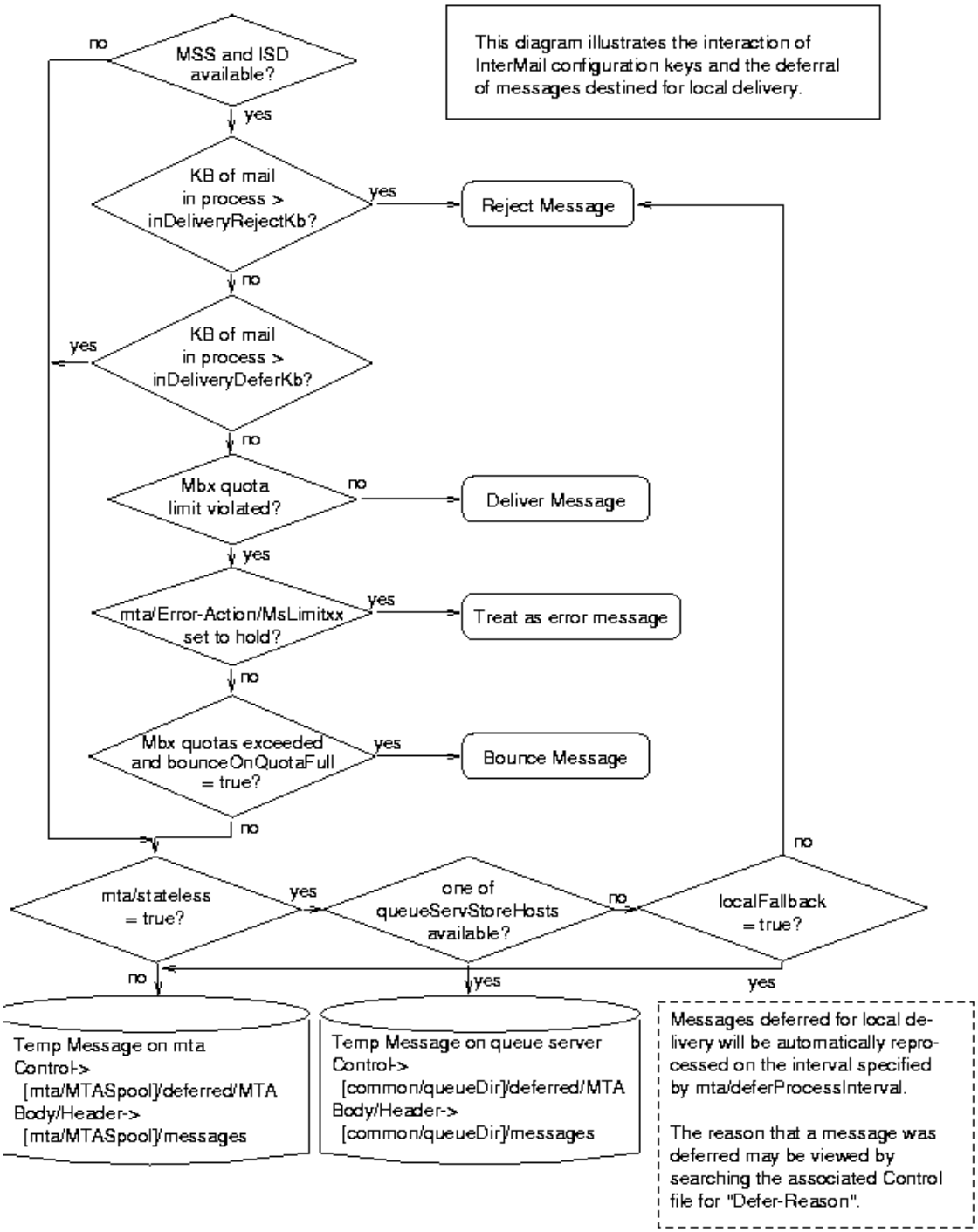

Figure 10 Local deferred mail

Remote Deferred Mail

When the remote mail server is unavailable, InterMail must store mail temporarily for remote mail domains. The system periodically tries delivering mail deferred for this reason. No direct intervention is necessary.

The `outboundDeferProcessInterval` configuration key sets the number of seconds between attempts to deliver mail to remote servers that are temporarily unavailable. The smaller the time interval between reprocessing attempts, the sooner mail delivery is likely to occur, but the greater the demand on the MTA's resources.

For example, if the value of the `outboundDeferProcessInterval` configuration key is 180, the MTA re-attempts delivery every 3 minutes (180 seconds).

Note: A separate configuration key (`/*/mta/outboundDeferProcessInitialWait`) controls the elapsed time between the restarting of the MTA and the initial attempt at reprocessing of deferred mail.

Figure 11 shows how InterMail processes remote deferred mail.

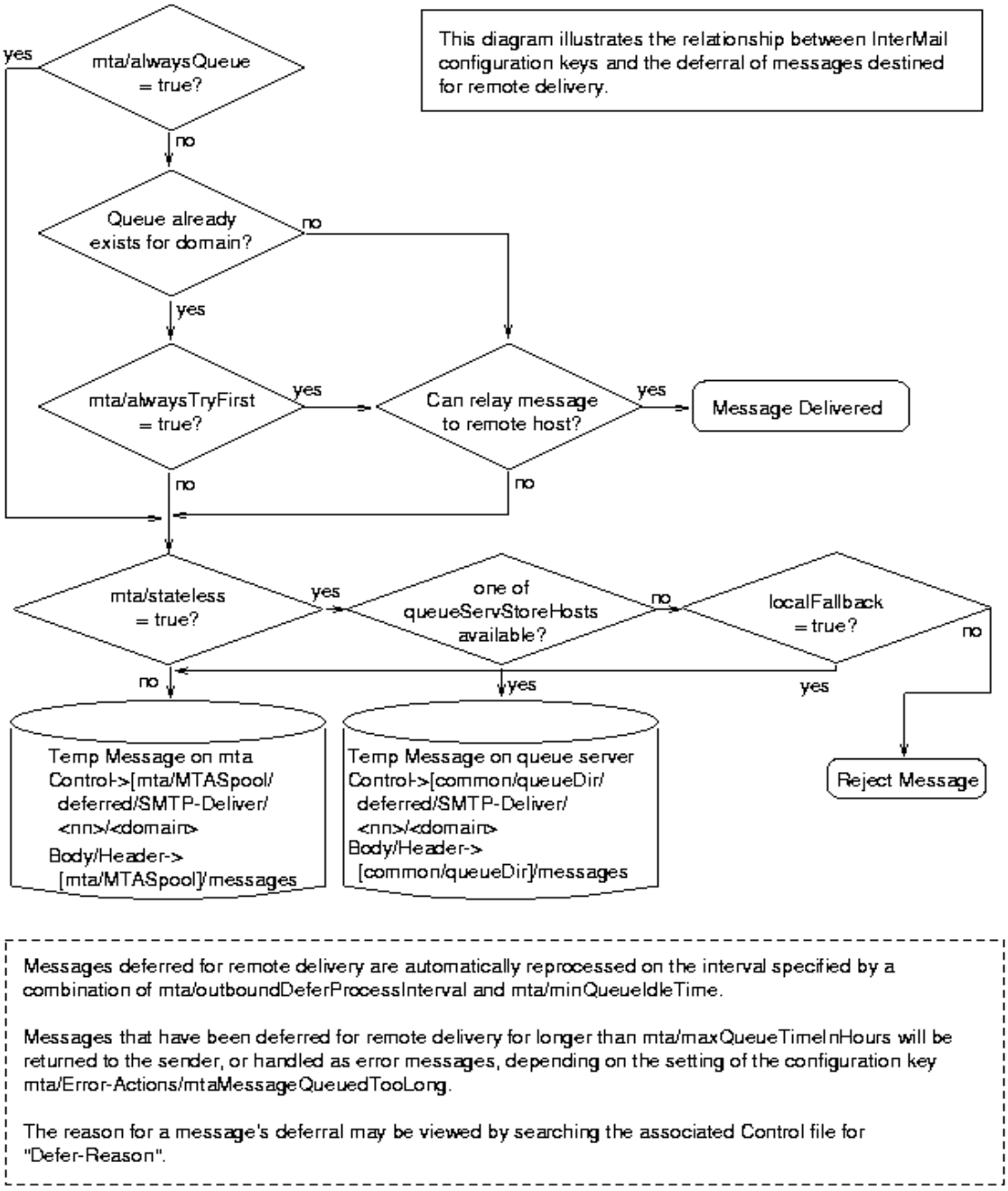

Figure 11 Remote deferred mail

Sidelined Mail

Message sidelining is one method of deferral over which you have total control. Basically, sidelining allows you to “hold” messages that you suspect are unsolicited commercial e-mail (UCE) or otherwise find interesting or suspicious. Sidelining can occur based on system policies, or system-wide or per-user SIEVE rule violations.

With InterMail’s mail sidelining option activated, the system moves messages that meet your criteria to a temporary location, from which you can view and process the messages at your leisure.

For a detailed discussion of sidelining options, see Chapter 5.

Figure 12 shows how InterMail processes sidelined mail.

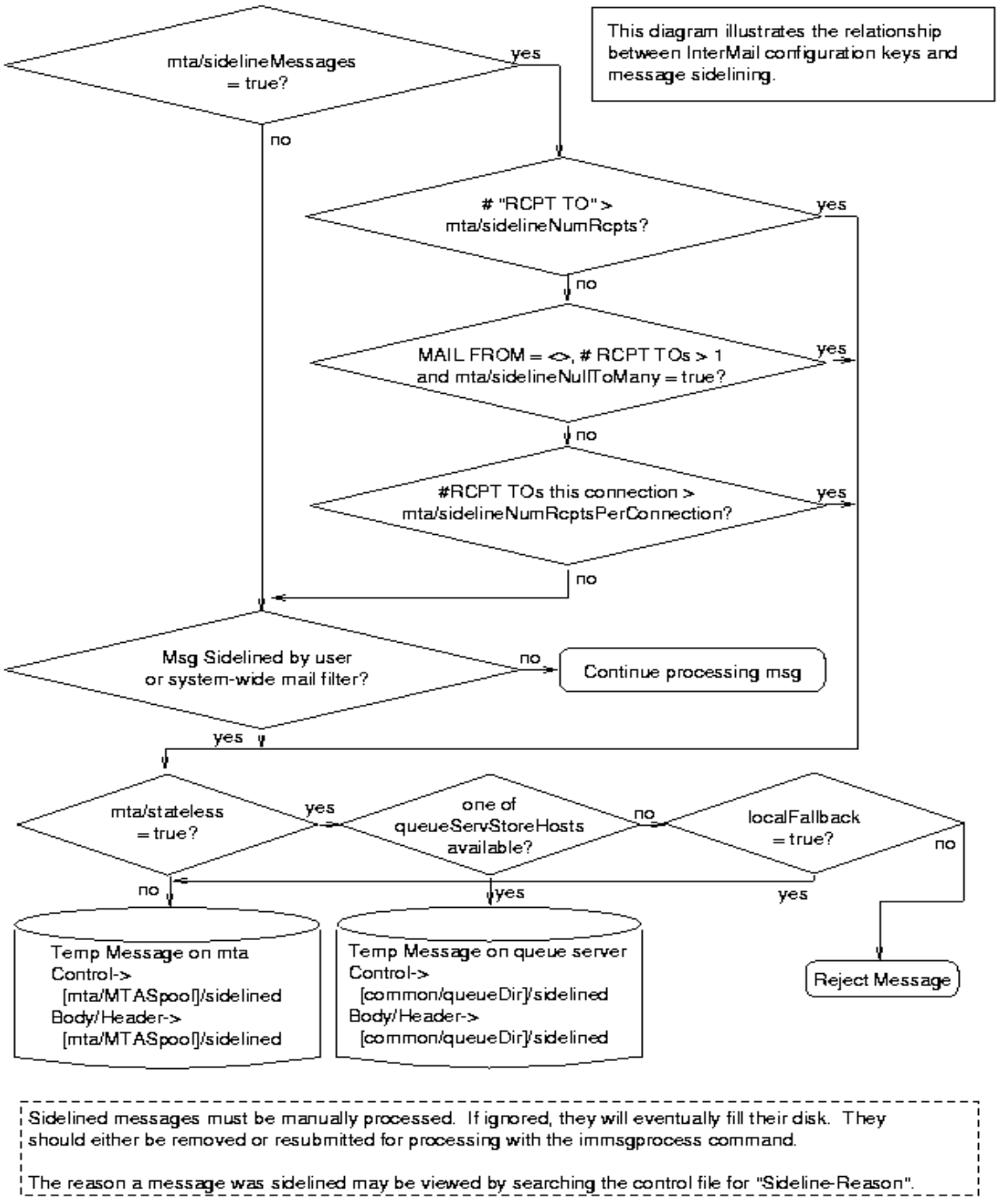

Figure 12 Sidelined mail

Mail Held Due to Errors

Sometimes a problem with the message itself may cause it to be deferred. For example, the system cannot bounce mail for an unknown user if the `MAIL FROM:` address is invalid.

As the system administrator, you must review and dispose of mail held due to errors. If you do not, the disk can fill up, causing delivery problems.

Figure 13 shows how InterMail processes mail held due to errors.

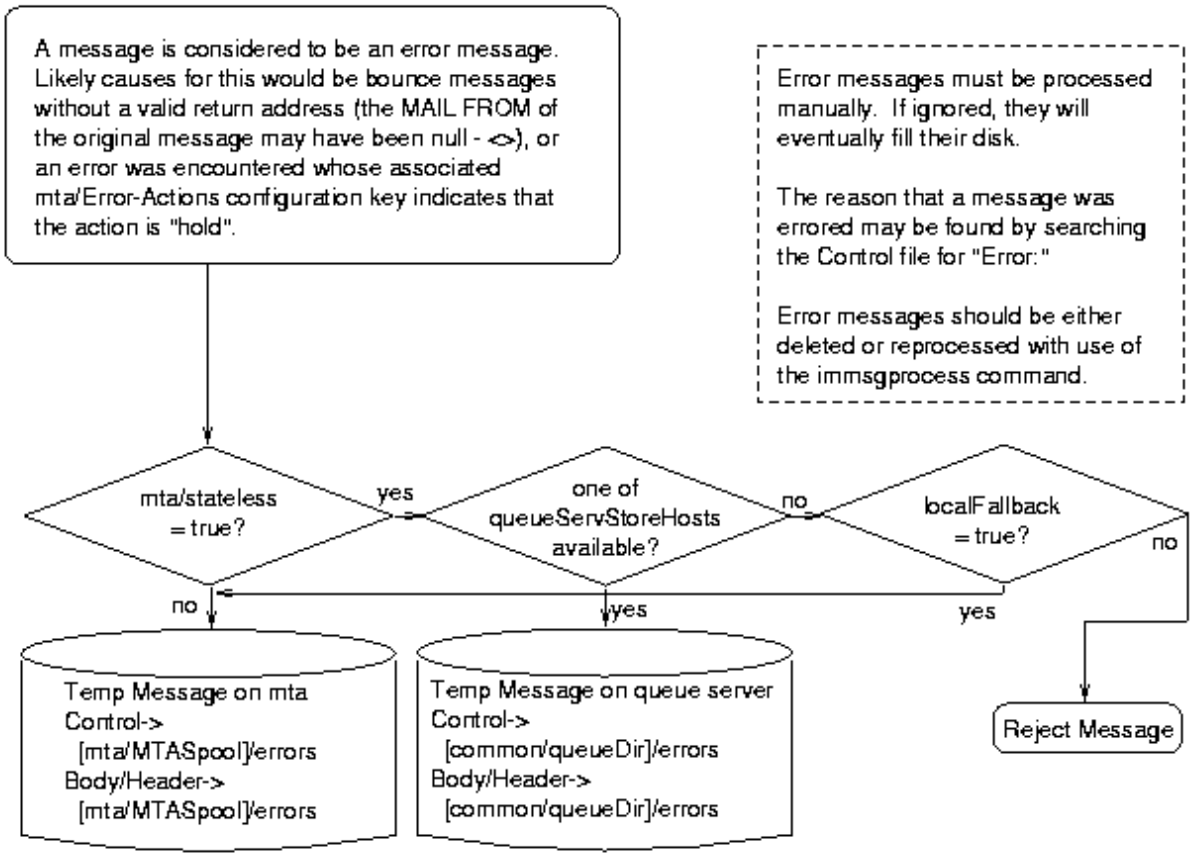

Figure 13 Mail held due to errors

Mail-in-Process Files

This section discusses the storage location, format, and directory structure of mail-in-process files.

Storage Location

Mail in process is typically stored on one or more Queue servers. However, if the servers listed in `mta/queueServStoreHosts` are not available, or if the `mta/stateless` configuration key is set to `false`, mail in process is stored on the MTA server.

It is possible for mail in process to be stored on both servers at once (if, for instance, a Queue server had been unavailable for some time and mail in process had been stored on the MTA during that interval). This is not a problem. An MTA can process stored messages from either or both sources as long as it is available.

Filename Format

The system stores mail in process differently from mail delivered to users. (For a discussion of permanent mail storage, see Chapter 8.)

There are three temporary files associated with each message in process:

- The **Control file** contains information about the message, including its current status in the delivery process and the names of the corresponding Header and Body files. Typically, the Control file will contain the reason why the message has been stored to disk rather than delivered from memory.
- The **Header file** contains the header information for messages, including `To:`, `Cc:`, `Bcc:`, `From:`, `Sender:`, and `Reply-To:` addresses.
- The **Body file** contains the body of the message, including the text and any attachments.

When the system has successfully delivered a message in process, it deletes these temporary files.

The Control, Header, and Body files share a prefix, which is unique for each set of files. The prefix consists of a time stamp, a 3-letter count of the number of messages received (the first is AAA, the next is AAB, and so on), the process ID (pid), the MTA hostname, and the HELO/EHLO or the IP address, assuming that one or the other exists.

Example:

```
19980601171253.AHA6542.mta1@rome.minorcorp.com
```

Each of the three files ends with a `-Control`, `-Header`, or `-Body` label that identifies the particular message component. Viewed together, the files might look like this:

```
19980601171253.AHA6542.mta1@rome.minorcorp.com-Control
19980601171253.AHA6542.mta1@rome.minorcorp.com-Header
19980601171253.AHA6542.mta1@rome.minorcorp.com-Body
```

The MTA generally stores mail in process in the `$INTERMAIL/queue` directory. You can configure this directory using the `queueDir` configuration key. Figure 14 shows the structure of the MTA's `queue` directory.

Directory Structure

Figure 14 shows the file structure of in-process mail. In some cases, mail in process will be stored on the local spool directory of the MTA (as defined by the `spoolDir` configuration key). The file system layout of the spool directory is the same as that of the queue directory. In cases where the MTA's local spool directory is used, "queue" should be replaced with "spool" in the following sections to determine the physical location of mail in process.

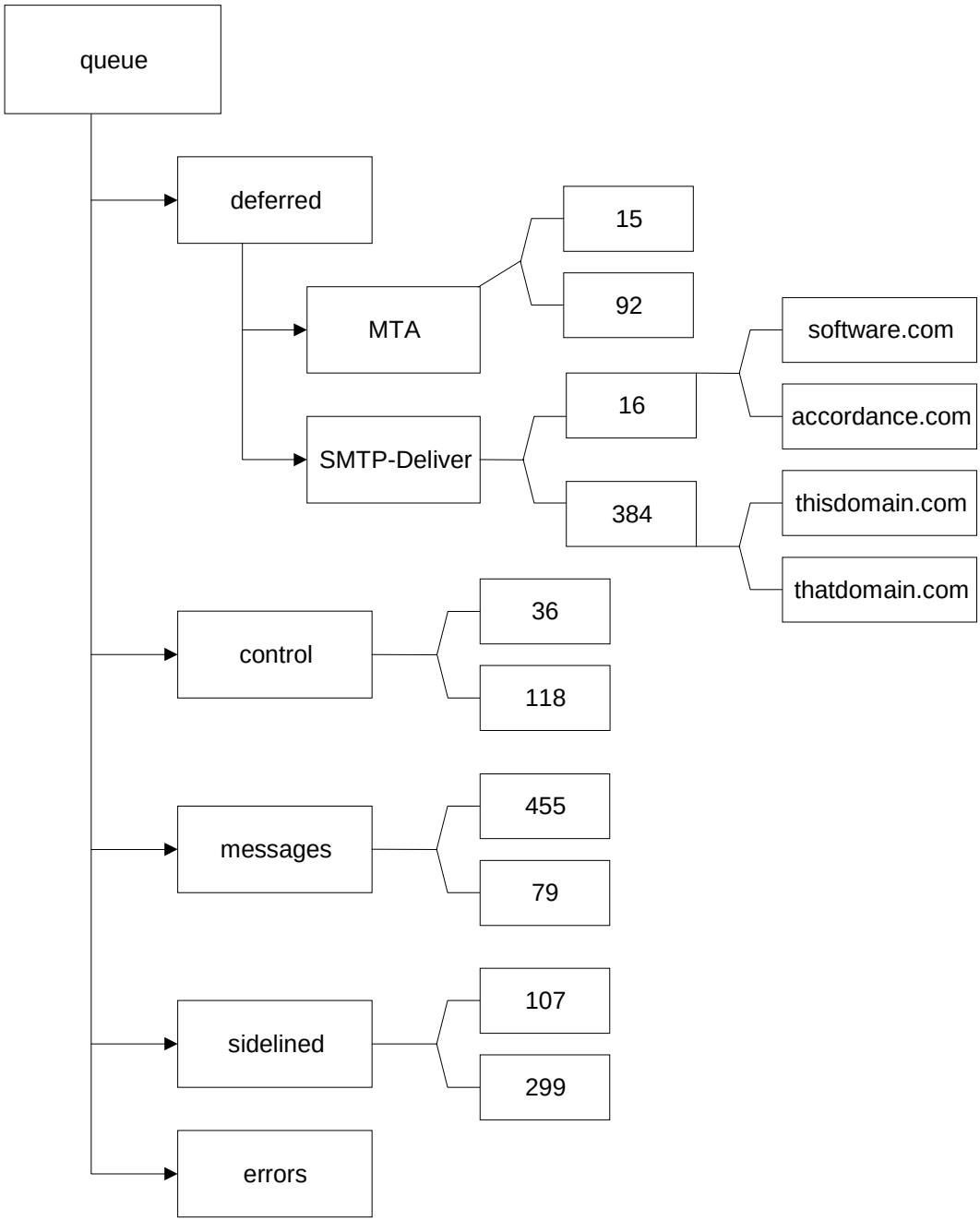

Figure 14 Structure of the MTA's queue (or spool) directory

The numbered boxes in Figure 14 show the structure of the MTA's queue (or spool) directory represent buckets, which are a series of numbered subdirectories. Buckets allow further subdivision of files within the directory structure, thereby maximizing performance by minimizing the possibility of many processes or threads simultaneously trying to access the same directory.

Note: The `bucketCount` configuration key controls the number of bucket directories. You set its value at installation and should never change it once the system is operational.

The location of files within the `queue` directory varies according to the reason for their storage:

Reason for Deferral	Control File Location	Header and Body File Location
Temporary storage	In a bucket in the `queue/control` directory; for example, `queue/control/237`	In a bucket in the `queue/messages` directory; for example, `queue/messages/237`
Local deferred mail	`queue/deferred/mta`	`queue/messages`
Remote deferred mail	In a subdirectory, by domain, within a bucket in the `queue/deferred/SMTP-Deliver` directory; for example, `queue/deferred/SMTP-Deliver/345/megacorp.com`	In a bucket in the `queue/messages` directory
Sidelined mail	In a bucket in the `queue/sidelined` directory	In a bucket in the `queue/sidelined` directory (the same bucket as the Control file)
Mail held due to errors	`queue/errors`	`queue/errors`

Managing Stored Mail

InterMail handles most mail in process automatically. Mail stored because it exceeds message size limits continues to be processed and delivered as soon as possible. Mail deferred because a local or remote host is unavailable is reprocessed by the system at regular, configurable intervals.

There are, however, two types of temporarily stored mail that do require some management. These are sidelined mail, and mail deferred due to system errors.

Reviewing and Reprocessing Sidelined Mail

This section describes methods for reviewing and processing sidelined mail. For a discussion of when and how to sideline mail, see Chapter 5.

When you enable one of InterMail's sidelining options, the system does not deliver or bounce incoming mail that meets the conditions specified. Instead, it moves the mail to the `queue/sidelined` directory, where the mail remains until you move or delete it. The system never deletes sidelined mail automatically, so if you elect to use sidelining, it is important that you check this directory on a regular basis to prevent these messages from piling up.

Suppose a message for more than 100 recipients has been sent to your `sidelined` directory and you must now decide what to do with it. If the message is actually junk mail, you'll probably want to delete it. But if the message turns out to be legitimate, you'll want to deliver it to its intended recipients.

As described in the previous section, the sidelined message has three components: Control file, Header file, and Body file. All files are simple text files and may be viewed with any text editor. To determine why the message was sidelined, you can search the Control file for "`Sideline-Reason`". You can also view the body and header file to determine whether the mail is actually junk mail, or contains a valid message.

If the message is junk mail, you can delete its Control, Header, and Body files with a normal operating system command.

Example:

```
rm 19980114235158898.AAA250@paris.minorcorp.com-*
```

In contrast, if the message is valid for delivery to its intended recipients, you use the `immsgprocess` administrative command to reintroduce the message for normal delivery The `immsgprocess` command has the following format:

```
immsgprocess <Control file>
```

Example:

```
immsgprocess 19980114235158898.AAA250@paris.minorcorp.com-Control
```

The `immsgprocess` utility moves the Control file of the message to the `queue/deferred` directory and the Header and Body files to the appropriate bucket in the `queue/messages` directory. The system then treats the message like ordinary deferred mail, and the MTA automatically attempts to deliver it to all its intended recipients.

Reprocessing Mail Deferred Due to System Errors

Most messages that the MTA receives are either delivered to their intended recipients or bounced when the MTA attempts delivery to a local domain with an unknown recipient. In the latter case, if the MTA is unable to return a bounce message because the original sender's return address is invalid, all three message components (Control, Body, and Header files) go into the `queue/errors` directory. To determine why a

message was shunted to the `errors` directory, you can search the Control file for "`Error:`".

In most cases, you will simply delete these files, since there is generally no way to fix the problem. If the problem can be fixed and it is desirable to have the message processed, the immsgprocess command must be used. It is important to keep an eye on this directory, though, because with large mail installations it can fill up very quickly.

Splitting Queues

At configurable intervals, InterMail automatically reprocesses messages deferred because a remote mail domain is temporarily unavailable. However, even though this process is automatic, manual intervention may be useful when queues become large.

InterMail creates a separate queue directory for each remote domain to which it cannot deliver mail immediately. In very large mail systems, it is not uncommon for some queues to become quite large, especially when a very busy remote site is offline for a long time. The larger a queue becomes, the longer it takes to write additional messages there. Splitting a very large queue into two or more smaller queues can speed the queuing process and enhance overall system performance. Given that InterMail opens just one connection per queue when it re-attempts delivery, splitting a large queue can also help speed delivery once an unavailable mail host comes back online. If there are several queues per domain, InterMail can open several connections to that domain and deliver messages from these multiple queues simultaneously.

The number of queues you should split off depends on a number of factors, including the number of threads your system has available for MTA mail delivery and the number of connections the domain that has been offline is able to accept.

Suppose that `megacorp.com`, a large remote domain, has been unavailable for several hours and then comes back online. You have a couple of megabytes of mail queued for `megacorp.com`, and now you want to split its queue in order to speed delivery. To determine how many separate queues to split off from the original, you need to know how many other network addresses are delivery points for `megacorp.com`. You can then create one new `megacorp.com` queue for each valid address. (Splitting off more queues than the number of valid addresses would be useless, since you can only connect to one address per queue.)

With this in mind, you run the following `nslookup` command:

```
nslookup -q=mx megacorp.com
```

Suppose this returns the following:

```
megacorp.com preference = 10, mail exchanger = a.mx.megacorp.com
megacorp.com preference = 10, mail exchanger = b.mx.megacorp.com
megacorp.com preference = 10, mail exchanger = c.mx.megacorp.com
megacorp.com preference = 10, mail exchanger = d.mx.megacorp.com
```

This tells you that there are four MX records for `megacorp.com`, which means that you could have four queues—and four simultaneous connections—to `megacorp.com`.

But before you decide to split your existing megacorp.com queue into four queues, consider the following:

- How much mail you have to send. If the amount isn't large, four connections may be a waste of resources for both you and megacorp.com.
- Whether megacorp.com will permit you to make four simultaneous connections to their site.
- Whether megacorp.com publishes the other MX records that point to its address. If it does not, your nslookup would not return anything but the address of its main mail exchanger.
- What the preference levels of the MX records are. MX records with values higher than 10 (such as 15, 20, or 25) are likely to indicate backup mail servers. If you tried delivering your queued mail there, these backup servers at megacorp.com would then have to reprocess the mail for delivery, which might create a problem for them.

When you are ready to split the queues, you enter:

```
imqueuesplit <destdomain> <newdest> [<newdest>]
```

Where:

destdomain	Is the name of the original queue directory.
newdest	Are the names of the new queue directories you wish to create.

In this case, for example, to split off three new queues for a total of four, you enter:

```
imqueuesplit megacorp.com a.mx.megacorp.com b.mx.megacorp.com
c.mx.megacorp.com
```

The imqueuesplit utility first creates the specified new queue directories. It then moves approximately 75% of the messages from the original megacorp.com queue into the three new directories, balancing the load as equally as possible among the four directories. The imqueuesplit utility always seeks to balance the message load equally, no matter how many queues you create.

With four queue directories for megacorp.com, the MTA now establishes four connections to this domain when it begins to deliver deferred mail. When it is finished executing, imqueuesplit prints out the number of messages it has processed and the domain to which it is routing messages.

Example:

```
Routing 264 messages to a.mx.minorcorp.com
Successful. 264 of 264 messages were processed.
Routing 264 messages to b.mx.minorcorp.com
Successful. 264 of 264 messages were processed.
```



```
Routing 263 messages to c.mx.minorcorp.com
Successful. 263 of 263 messages were processed.
```

Any failure results in one of the following messages:

```
Couldn’t open <file>! <reason>
Couldn’t modify Host-To in <file>!
Couldn’t create <file>! <reason>
Problem writing <file>! <reason>
New file was written incorrectly: <file>!
```

Setting Queuing Options

InterMail offers a variety of other options that affect mail queueing and reprocessing.

Processing Interval Keys

Configuration keys allow you to set:

- The intervals at which the MTA re-attempts delivery of deferred messages
- The amount of time a queue must remain idle between processing times
- The maximum amount of time a message can remain queued
- Whether, if a domain already has a queue, the MTA tries to deliver messages to that domain before adding them to the queue
- A queue for every domain, with all messages in the queue sent during a single connection

Delivery Intervals

The `deferProcessInterval` configuration key sets the length of time between the MTA’s checks for local queued mail. The `outboundDeferProcessInterval` configuration key sets the length of time between the MTA’s checks for outbound queued mail. After each configured interval has elapsed, the MTA checks its `queue` directory and attempts to deliver messages for all domains that are queuing mail. If delivery is again unsuccessful, it queues the mail again until the next processing interval.

The more frequent the reprocessing attempts, the sooner the mail is likely to be delivered. However, the more frequent the delivery attempts, the greater is the demand on the MTA’s resources.

Minimum Idle Time

The `minQueueIdleTime` configuration key tells the MTA queue scanner how much time must elapse between when it last processed a given queue and when it next processes that same queue. The scanner checks the queues at intervals set by the `outboundDeferProcessInterval` key, but it does not actually process them unless the time exceeds the value of the `minQueueIdleTime` key.

This feature helps distribute queue processing power so that a few large directories don't delay processing of all the other directories in the queue. For example, suppose that, because the MTA processed a large queue for `megacorp.com` just 10 minutes ago as specified by the `outboundDeferProcessInterval` key, you want to devote more resources to processing other queues when the next configured delivery time comes around. You therefore set the value of `minQueueIdleTime` to be higher than the value of `outboundDeferProcessInterval`. Now, at each delivery time (specified by `outboundDeferProcessInterval`), the queue scanner checks to see how long each queue has been idle and processes only those queues that have been idle longer than the time specified by `minQueueIdleTime`.

Note: For `minQueueIdleTime` to have any effect, it must be set to a higher value than `outboundDeferProcessInterval`.

Maximum Queue Time

Occasionally it is impossible to deliver a message, as when a message's addressed domain becomes unavailable and does not come back online within a reasonable amount of time.

The `maxQueueTimeInHours` configuration key allows you to set what you consider a "reasonable amount of time." Internet standards recommend a period of at least four or five days. Once the maximum period that you have defined expires, the MTA generates a bounce notice and returns the message to its sender.

Delivery Before Queuing

The `alwaysTryFirst` configuration key determines how the MTA handles delivery of mail to a domain that already has a queue. By default, the MTA does not attempt to deliver additional messages to this domain. Instead, it sends any new messages for that domain directly to the `queue` directory, which adds them to one of the domain's existing queues. You can change the value of this configuration key so that the MTA always attempt to deliver a message before adding it to an existing queue. You should bear in mind, though, that this setting affects all domains for which mail is queuing at any given time and that, since InterMail opens a connection for each delivery attempt, system performance may decrease if a domain is unavailable for a long time.

Queuing for All Messages

The `alwaysQueue` configuration key limits the number of connections to a domain over time by creating a queue for every domain and storing all messages there until the next configured delivery time specified by `outboundDeferProcessInterval`. At that time, the MTA attempts to send all queued messages for a domain during a single connection.

This is useful for situations in which you never want the MTA to attempt immediate delivery, regardless of whether there is a problem that would ordinarily cause messages to queue. One such situation might be when there is a remote domain to which you send large volumes of mail. If, for example, your MTA sends one message

to `megacorp.com` every 5 or 10 seconds, you may decide that it is a waste of system resources to establish one connection per message.

Bear in mind, however, that setting `alwaysQueue` to `true` queues all outgoing messages and therefore delays all mail delivery, since the system never processes messages immediately.

Note: If `alwaysQueue` and `alwaysTryFirst` are both set to `true`, `alwaysQueue` takes precedence, and mail is always queued.

Configuration Key Summary

The following table summarizes the configuration keys to set options for mail deferred due to unavailable local or remote domains. For a more complete description, see the *InterMail Mx Reference Guide.*

`deferProcessInterval`	Interval, in seconds, between delivery attempts for queued local mail (deferred when a user's mailbox or account information is temporarily unavailable). When this number of seconds have elapsed, the MTA attempts to send the deferred messages. The default value is 600 seconds (10 minutes).
`outboundDeferProcessInterval`	Interval, in seconds, between delivery attempts for mail that is queued because a remote domain is temporarily unavailable. When this number of seconds have elapsed, the MTA attempts to send the deferred messages. The default value is 300 seconds (5 minutes).
`minQueueIdleTime`	Minimum time, in seconds, between attempts to deliver queued mail to a given domain. The default value is 60 seconds (1 minute).
`maxQueueTimeInHours`	Maximum number of hours a message can be kept in the queue. Once a message has been in the queue this long, InterMail returns it to its sender. The default value is 96 hours (4 days).
`alwaysTryFirst`	Whether delivery is to be attempted for each message before queuing. When the value is set to `true`, the MTA always attempts delivery of a message before queuing it in its `queue` directory. The default value is `false`.

alwaysQueue	Whether every message for every domain is to be queued and all messages sent during a single connection. The default value is false.

ETRN (SMTP Queue Processing Requests)

Yet another mail queue processing option is the SMTP ETRN command, which can instruct remote mail hosts to attempt delivery of queued messages. This is useful if you have a PPP, SLIP, or similarly intermittent connection to the Internet.

To request manual queue processing, telnet to port 25 of the SMTP server and enter:

```
ETRN @<domain>
```

where <domain> is the domain of the queue to be processed. Your queued mail is then delivered immediately, regardless of the interval specified by the deferProcessInterval key.

Example:

```
220 london.minorcorp.com ESMTP server (InterMail v4.0 212) ready Tue,
12 May
1998 09:20:30 -0700
HELO
250 london.megacorp.com
ETRN @minorcorp.com
250 Ok
QUIT
```

The 250 Ok response acknowledges that the request has been carried out.

Note: ETRN is an open protocol standard described in RFC 1985.

Throttling Mail in Delivery

Mail in delivery is a subset of mail in process. It consists of those messages that the system is actively processing and that have not been explicitly deferred. To protect against excessive system load, InterMail offers a series of controls for managing mail in delivery. Among these controls is throttling, which allows you to limit the amount of mail throughput in the system.

Three configuration keys are involved in throttling mail:

- inDeliveryDeferKb defines how much mail can be in the process of delivery before the MTA is considered overloaded. After this limit is reached, the MTA accepts new messages but defers their delivery. Once the total volume of messages currently in delivery drops below this threshold, the MTA starts delivering messages again.

- `inDeliveryRejectKb` defines how much mail can be in the process of delivery before the MTA is considered overloaded. After this limit is reached, the MTA rejects any new messages, returning them with a request asking senders to try again later. Once the total volume of messages currently in delivery drops below this threshold, the MTA starts accepting messages again.
- `inDeliveryStopDeferKb` sets a size limit of mail in delivery at which processing of deferred mail ceases in order to allow the MTA time to catch up.

Note: The system treats mail deferred by throttling like any other deferred mail, and reprocesses it automatically at regular intervals.

The default values of these keys (in kilobytes) are:

```
/*/mta/inDeliveryDeferKb: [1000000]
/*/mta/inDeliveryRejectKb: [0]
/*/mta/inDeliveryStopDeferProcessKb: [10000]
```

These keys complement each other, making their relative values significant.

For example, the `inDeliverDeferKb` limit defers mail but does not reject it outright. In contrast, the purpose of `inDeliveryRejectKb` is to stop additional messages from being accepted when the demand for simultaneous message delivery is so high that even temporary deferral cannot prevent a serious decline in performance. Therefore, if the value of the `inDeliveryDeferKb` key is `1000000` (the default), the value of the `inDeliveryRejectKb` key should be set higher than `1000000` in order for this more drastic measure to be reserved for last.

The system checks the `inDeliveryStopDeferProcessKb` at the start of each processing interval. If the current amount of mail in delivery exceeds the limit set by this key, processing of deferred mail does not occur, and an `MtaTooBusyStopDefer` entry is written to the log. The system will again attempt to deliver deferred mail after the next processing interval.

Note: The system writes log entries each time one of the mail-throttling controls is activated. You should review log files regularly for `MtaTooBusyDefer`, `MtaTooBusyReject`, and `MtaTooBusyStopDefer` entries, which are evidence of excessive system load.

8

Managing Mail Storage

Each InterMail system processes a variety of messages—some destined for local recipients, and others generated by local users for delivery to remote destinations. While mail is being processed, InterMail may store any of these messages temporarily. However, once delivery is completed, the system must persistently store all messages addressed to local users until the users retrieve or delete them.

This chapter covers:

- Physical and logical message storage
- Creating, moving, and deleting mailboxes

Note: Temporary storage of mail in process is discussed in Chapter 7.

Message Storage

A typical InterMail system includes thousands of users, each with a large number of messages. In installations of this size, efficient storage of message data and fast access to message information are critical. To accommodate both goals, InterMail distinguishes between physical and logical mail storage, and provides separate storage mechanisms for each.

The Message Store Server (MSS) is the InterMail component responsible for persistent storage of mail. Each MSS host contains:

- a **Message File system**, which accommodates physical message storage
- a **Message Store database**, which manages logical message storage

For each message delivered to a local user, the system must store information in both the Message File system and the Message Store database. The Message File system stores the content of the message, while the Message Store database stores additional information about the message (status, intended recipients, and so on).

The advantage of separating physical and logical mail storage becomes apparent when you consider persistent storage of a single message addressed to multiple local users. Figure 15 illustrates storage of a 5-MB message addressed to local users John Doe and Susie Queue.

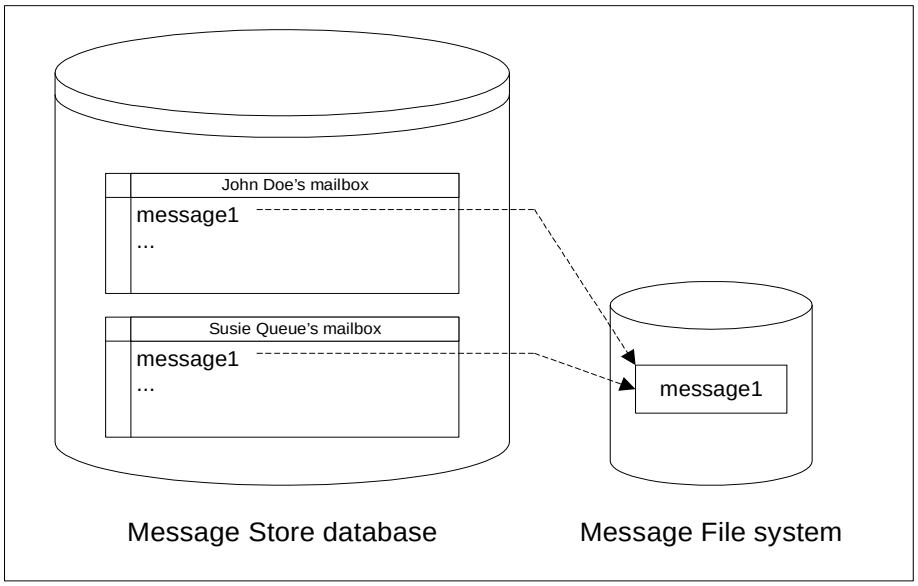

Figure 15 A single message stored for multiple users on the MSS

Multiple entries exist in the Message Store database to record receipt of the message by both John and Susie, to note status information for each, and to reference the location of the associated message file. However, the system stores the 5 MB of message data, which is common to both recipients, only once, as a single message file in the Message File system.

Physical Message Storage

The Message File system stores the body and header of each message as a single binary file. The body of the message includes the text and any attachments. It remains in the Message File system until all of its intended recipients have read and deleted it, or until it has reached its expiration limit. The header includes a summary of the contents of the message, as well as a description of the path the message has taken on its way to the recipient. Message data is static; once InterMail stores it, it never changes.

The Message File system stores one file per message, regardless of the number of intended recipients for that message. Message files reside in a large directory tree created for this purpose. In a very large InterMail system, there may be thousands of directories in this tree, together containing millions of stored messages. The configuration key `messageFilesDir` specifies the top of the Message File system hierarchy (by default, `$INTERMAIL/msgfiles`). This directory contains three items:

- `messages`, a directory that contains the thousands of "bucket" directories that store individual message files.

- `buckets.dir`, a file that lists the name of the messages directory (by default, `messages`).
- `buckets`, a file that contains a partial list of the bucket directories within `messages`. The MSS uses this file to determine the names and locations of the available bucket directories. To balance message volume among bucket directories, this directory list excludes the one-third of all bucket directories that contain the greatest message volume.

This Message File system directory tree is created automatically during installation. However, if you later need to create another directory structure to accommodate greater message activity, you can do so using the `imbucketscreate` administrative command.

Caution! You should never manually modify the files or directories that make up the Message File system. If you need to modify the structure of the Message File system, use `imbucketscreate`. For more information on `imbucketscreate`, see the *InterMail Mx Reference Guide*.

Note: For information about routine tasks needed to maintain the Message File system, including expanding, balancing, and defragmenting it, see Chapter 9.

Logical Message Storage

The Message Store database stores information about a message, such as its status, intended recipients, and so on. Unlike the message itself, which is static, information in the Message Store database is dynamic. For example, the status of a message changes once a recipient has retrieved it.

Within the Message Store database, the following data objects define a logical relationship between end user accounts and individual messages:

- Mailboxes
- Folders
- Messages

Note: These are abstract data objects. Although they define a hierarchical relationship, they do not literally exist in a physical data structure. This is important to keep in mind, especially when you are moving mailboxes from one MSS host to another.

Mailboxes

A mailbox is a storage area for messages sent to an end user’s account. Each account in the Integrated Services Directory (ISD) that uses the local delivery method has a mailbox. Each mailbox “contains” a series of folders, which in turn contain the

messages that the end user has received. Mailboxes are at the top of the MSS database object hierarchy. As such, most administrative functions in the MSS database operate on these objects.

Note: A mailbox is required by an account only if the account uses the local delivery method. Accounts that use only forwarding delivery do not require or make use of an MSS mailbox.

Folders

A folder is a container for messages. Each folder exists within a specific mailbox (and therefore, for a specific end user). Folders can exist within another folder. By default, all InterMail mailboxes contain the following folders:

- INBOX
- SentMail
- Trash

`INBOX` is the folder that receives all new messages from the MSS. Although end users who retrieve mail using a POP server cannot see the folders, they always retrieve their messages from the `INBOX` folder.

IMAP clients allow end users to create new folders in the MSS; copy, delete, and move messages between folders; and delete folders. Like many clients, IMAP clients allow users to save a copy of all outgoing messages to a specific folder (in this case, `SentMail`) while moving "deleted" messages to another folder (in this case, `Trash`).

Note: IMAP terminology sometimes refers to folders as "mailboxes." Be careful not to confuse these terms. In InterMail, mailboxes are the top-level objects in the Message Store database (one per account), whereas folders are message containers in mailboxes (many per account).

When InterMail encounters a message that the POP or IMAP server could not retrieve for some reason, such as because of loss of the corresponding message file or corruption of database information, InterMail moves the message from its folder into the `.ERRORS` folder (creating the folder first, if necessary), thereby enabling normal processing to continue for subsequent messages.

Note: The `immsgprocess` administrative command reprocesses messages that have moved to the `.ERRORS` folder. For information on this command, see the *InterMail Mx Reference Guide*.

Messages

A message object in the Message Store database corresponds to an individual message that is "in" one or more mailboxes. The information stored for each message object in the Message Store database includes:

- The path to the corresponding message file in the Message File system.
- Pointers to the mailboxes and folders in which the message resides. These pointers establish that the message is "in" one or more mailboxes, even though only one message object exists in the Message Store database.
- Message headers, which include all information in the message that precedes the body. This header information is the only data that is in both the Message File system and Message Store database.
- Message status flags, which indicate whether the end user has read or deleted the message. If the message is in more than one folder (because it was received by two different end users, for example), the message object includes a set of status flags for each folder..

Note: Message status flags have special meaning to IMAP clients, which typically make use of this information to report status information to end users.

Creating Mailboxes

Although mailboxes belong to accounts, the InterMail system does not create an account's mailbox when it creates the account itself in the ISD. Mailboxes and their default folders (`INBOX`, `SentMail`, and `Trash`) are created later, either automatically or manually.

Because it requires minimal administrative effort, automatic creation is by far the most commonly used method. However, if circumstances warrant it, you can create mailboxes manually using `imboxcreate` or an API function. For example, you might choose to create mailboxes manually when you have a large number of newly provisioned accounts and you expect a high percentage of these accounts to become active for the first time. In this case, automatic mailbox creation might cause excessive system load.

Automatic Creation

InterMail automatically creates a mailbox for an existing account the first time that it needs that mailbox. Two events can trigger automatic creation:

- The account receives a message and needs to store it in the account's mailbox.
- The end user attempts to access his or her mailbox through the POP or IMAP server.

By waiting to create mailboxes until they are needed for message storage or retrieval, InterMail enables the MSS to avoid unnecessary processing. This method also avoids the unnecessary creation of mailboxes for accounts that use only forwarding delivery, not local delivery.

Note: The local delivery method specifies that messages received for an account must be stored in the account mailbox. If an account does not use local delivery (that is, uses forwarding delivery only), it does not require a mailbox.

The `createsMboxes` configuration key controls the automatic creation of mailboxes. If the value of this key is `true`, the system creates a mailbox for an account the first time that it needs one. If the value of this key is `false`, you must create mailboxes manually (as described in the following section). You can define this key separately for the Message Transport Agent (MTA), POP server, IMAP server, and Web server or—more commonly—you can define it once globally using the configuration database entry `/*/common/createsMboxes`.

Manual Creation

You can create mailboxes manually in the Message Store database in two ways:

- By executing the `imboxcreate` administrative command
- By creating a program that calls the appropriate InterMail API functions

This manual covers only the use of `imboxcreate`; for information on using the InterMail API libraries, see the *InterMail Mx Reference Guide*.

The `imboxcreate` command creates a single mailbox in the Message Store database. The syntax of this command is:

```
imboxcreate [-help] <host> <internalID>
```

Where:

`-help`	Provides a usage statement.
`host`	Is the name of the MSS host on which the mailbox is to be created.
`internalID`	Is the internal ID number of the account to be associated with the mailbox.

Note: To determine the internal ID number of an account, use the `imdbcontrol` or `imaccountquery` administrative command. For descriptions of these two commands, see the *InterMail Mx Reference Guide*.

Example:

To create a mailbox on the host `paris` for the account whose internal ID number is `123456`, enter:

```
imboxcreate paris 123456
```

When it completes its operation, `imboxcreate` displays the results:

```
Message store 123456 created!
```

Moving Mailboxes

InterMail enables you to move mailboxes from one MSS host to another. You may want to do this for several reasons, such as that:

- Your InterMail system is expanding and requires an additional MSS host.
- You are decommissioning one MSS host and replacing it with another.

Executing imboxmove

You must execute the `imboxmove` command on the MSS host from which mailboxes will move. The syntax of this command is:

```
imboxmove [-i|-b][-d <sourceHost>] <destHost> <file> [<file>...]
```

Where:

`-I`	Runs `imboxmove` in interactive mode, which requires you to type `<Enter>` after each operation.
`-b`	Runs `imboxmove` in batch mode, which disables all prompts.
`-d <sourceHost>`	Delete successfully moved mailboxes from source host
`<destHost>`	Is the name of the destination MSS host to which mailboxes will move.
`file`	Is the name of a file that contains the primary e-mail address (`<username@domain>`) of each account whose mailbox is moving to the destination MSS host.

When executed, `imboxmove` does the following:

1. Retrieves the list of addresses from the address file specified.
2. Verifies that the status of each of the specified accounts in the ISD is Active. If an account is not active—for example, if it is locked—its mailbox cannot be moved.
3. Changes the status of each account to Maintenance, which queues mail to these users internally and does not deliver that mail to the MSS. This step occurs for all

accounts simultaneously, so all of the affected accounts remain unavailable until `imboxmove` completes its operations.

4. Copies the database records associated with each account—the mailbox, folders, and messages—to the destination MSS host's Message Store database.
5. Copies the message files associated with the moved mailboxes to the destination MSS host's Message File system.
6. Changes the value of each account's MSS host attribute in the ISD to the new MSS host.
7. Resets the status of each account to Active.

Depending on the number of mailboxes being moved and the volume of messages that they contain, these operations may take a long time to complete. To reduce the time required, it is a good idea to move only a few hundred mailboxes at a time.

Note: The `imboxmove` utility does not delete mailboxes from the original MSS host after moving them. If you want to delete mailboxes after moving them, use the `imboxdelete` command, described in "Deleting Mailboxes" on page 156.

Sample Scenario

In this example, the MSS host `venus` will go offline permanently, and a new MSS host named `mercury` (on which InterMail is already installed) will replace it. To move all mailboxes from `venus` to `mercury`, you:

1. Log in to the MSS host `venus`.
2. Create a file, called `addressfile`, that contains a list of e-mail addresses for the accounts whose mailboxes should be moved.
3. Execute `imboxmove`, specifying the address file created in step 2 as the source for account information:

   ```
   imboxmove -b mercury addressfile
   ```

   Mailboxes are moved, and `imboxmove` reports on mailboxes that it could not move (for example, mailboxes with locked accounts).
4. Repeat the above steps for additional blocks of accounts until all mailboxes have been moved successfully.
5. Review the information on mailboxes that were not moved and resolve the associated issues so you can move them later.

Deleting Mailboxes

Mailboxes cannot be deleted automatically; you must delete mailboxes manually. You do this with the `imboxdelete` administration command, which deletes from the

Message Store database both the specified mailbox and all of the folders and messages contained within it.

Note: Deleting a mailbox is not the same as deleting an account. Accounts are in the ISD, which `imboxdelete` does not affect.

The syntax of this command is:

```
imboxdelete [-v] [-a|-m <file>] <host> <internalID>
```

Where:

`-v`	Verbose execution.
`-a <file>`	An input file containing e-mail addresses.
`-m <file>`	An input file containing mailbox numbers.
`<host>`	Is the name of the MSS host with the mailbox to be deleted.
`<internalID>`	Is the internal ID number of the mailbox to be deleted.

Example:

To delete a mailbox that has the internal ID number `010671` from the MSS host `mercury`, enter:

```
imboxdelete mercury 010671
```

When it completes the mailbox deletion, `imboxdelete` confirms that the operation was successful:

```
imboxdelete:  Message store 010671 deleted!
```

Removing Deleted Messages

After messages have been deleted from the Message Store database, it is important to delete the associated message files from the Message File system. You do this through a process known as “garbage collection.”

To understand how a message becomes “deleted,” recall that, for each message, the Message Store database contains (among other things) a list of pointers to the mailboxes and folders in which the message resides. As recipients delete a particular message from their mailboxes, the list of pointers for that message object is updated to reflect these deletions. As long as at least one folder contains the message, the system keeps that message object and its associated message file. However, once every recipient of the message has deleted it, and the system has removed the final pointer to it in the Message Store database, the system considers the message deleted. At this point, the system must run garbage collection on its associated message file in the Message File system.

The administration command that deletes messages from the Message File system, `immssgc`, runs regularly (by default, once an hour) as a `cron` job. When executed, `immssgc` queries the Message Store database for a list of messages marked for deletion in the database, and it permanently deletes both the database object and the message file associated with each message. Because it may take a long time to complete, `immssgc` deletes only a limited number of messages in one execution.

The `immssgc` command uses the `remoteDatabaseOption` configuration key. For example, if you intend to run `immssgc` locally, you must set the `remoteDatabaseOption` configuration key to `false`.

The syntax of this command is:

```
immssgc [-altrb][-c <number>][-h <hours>][-f][-p][-v]
```

Where:

`-altrb`	Specifies the alternate rollback segment `RB_IMMSSGC`.
`number`	Is the maximum number of messages to be deleted in a single execution. If no value is given, the default value of 1,000 is used.
`hours`	Is the minimum age (in hours) of messages to be deleted. Messages younger than this are retained.
`-f`	Causes a "full" garbage collection, in which every message in the Message File system is checked for mailbox references and, if there are none, deleted. This option is generally not advisable, because it is a very slow process that may use system resources inefficiently.
	Note: You might use the `-f` option in the unlikely event that `im_messagedeleted` is corrupted, and the index is corrupted.
`-p`	If the command includes the `-v` option, prompts you before processing each batch of messages.
`-v`	Requests verbose execution.

Caution! Failing to run `immssgc` on a regular basis may cause the size of the Message File system to grow at an alarming rate. It is essential that you implement garbage collection on a regular basis.

9

System Monitoring and Maintenance

The InterMail system is designed to operate 24 hours a day, seven days a week. Like all mission-critical applications, it warrants constant monitoring and maintenance, and adjustment of individual components in accordance with performance requirements. This chapter describes some of the monitoring and maintenance tasks that are required to ensure smooth running of the InterMail software.

Regular operational procedures on the InterMail system consist of three basic categories of tasks: system monitoring, performance monitoring and tuning, and system maintenance.

- System monitoring is the regular review of service parameters that indicate the system's condition. This involves, among other things, monitoring server availability, checking physical disk usage, and checking log files. System monitoring should be done on regular basis.
- Performance monitoring is the inspection of key performance indicators that reflect overall system performance. The performance indicators help determine how much system capacity is available for future growth.
- System maintenance involves regular tasks necessary to maintain system operations for extended periods. This includes tasks like backing up log files and expanding disk space, if required.

This chapter covers:

- System monitoring
- Performance monitoring
- Monitoring Oracle databases
- InterMail performance tuning
- System maintenance

System Monitoring

System monitoring is an important set of tasks that checks the InterMail system for running state. The purpose of these tasks is to identify operational problems so that operations personnel can be notified if a service-impacting event is about to occur or already has occurred. Immediate response to this notification can help prevent serious problems.

This section describes the system monitoring tasks with respect to:

- Server availability
- Network availability and utilization
- Physical disk usage
- File system usage
- Log files
- `syslogd` output
- `cron` jobs

Server Availability

In a production InterMail environment, it is important to ensure that all servers are available and can respond in a timely manner. One quick method of checking server availability is to `telnet` to the POP, IMAP, SMTP, and HTTP ports on the appropriate InterMail hosts. Another method is to ping these hosts using `imservping`.

Telnetting to Server Ports

In the following example, a telnet session starts on port 110 (the POP port):

```
paris% telnet 0 110
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
+OK InterMail POP3 server ready.
quit
+OK ? InterMail POP3 server signing off.
Connection closed by foreign host.
```

After the `telnet` command is entered, the POP (or SMTP or IMAP) banner (such as "`+OK InterMail POP3 server ready`") should appear immediately (within 3 seconds). If the banner does not appear immediately, there may be excessive stress on the server.

Pinging Servers

Each InterMail server can also be pinged with timeout values to determine if the servers are available and responding. To determine server availability, run `imservping` as in the following example:


```
paris% imservping 1 5 mta

Fri Jan 08 10:03:18 EST 1999. imservping: (Info) mta responded
```

In this example, `imservping` pinged the MTA with a warning time of 1 second (to begin pinging) and maximum time of 5 seconds (within which to report).

Network Availability and Utilization

The network used by the InterMail servers should be monitored at all times for integrity and bandwidth utilization. There are many commercial products available for this kind of monitoring, most of which can also be configured to monitor error rates on server interfaces. Alternatively, the UNIX `netstat` command can be used to monitor various interface and protocol measurements on a server.

Physical Disk Usage

Disk usage refers to the amount of physical memory (hard drive space) available to InterMail at any one time. Certain InterMail components, specifically the Message Store database and the Directory database, need to be checked on a regular basis for their size. Regularly check the devices or mount points where these databases reside for disk space usage.

In addition, the InterMail Message File system (`msgfiles`) and other parts of the InterMail file system that are not server-specific (such as the `logs` directory) tend to grow quickly. Without an effective archiving and backup strategy in place, they can consume large amounts of hard drive space. The directories where growth is likely to create problems are the `logs` and `spool` directories. In some cases (depending on how sidelining has been implemented), the growth of the `/spool/sidelined` directory can also become problematic.

Using `df` or a similar utility, observe the amount of space used in a system, by mount point. After determining the relevant mount point, run `du` on the directories at that mount point, and determine which file sizes are large enough to warrant inspection. Typically, files that cause a drain on disk space, and the strategy for dealing with these files, are as follows:

For these files:	In this directory:	Do this:
Message files	`msgfiles`	If space becomes an issue for message files, allocate more space.
Journal files	`journal`	Determine or revisit the backup strategy for old journal files.
Log files	`log`	Set log rollover to occur more frequently; physically move old log files to a separate file system or media source.

For these files:	In this directory:	Do this:
Sidelined mail	`queue/sidelined` or `spool/sidelined`	Manually inspect messages and reprocess them using the `immsgprocess` command.
Deferred mail	`queue/deferred` or `spool/deferred`	Fix any undeliverable mail and reprocess this mail using the `immsgprocess` command.
Messages held in error	`queue/errors` or `spool/errors`	Manually inspect and clean out periodically for messages to and from bad users. If the problem with a message is corrected, you can resubmit the message for processing using the `immsgprocess` command.
Oracle archived redo logs	Oracle directory	Maintain these files as part of the Oracle backup strategy. For a discussion on backing up Oracle, see Chapter 11.
Orphaned message files	`queue/messages` or `spool/messages`	Remove orphaned message files using the `imorphanrm` command. Do not attempt to delete them by hand. `imorphanrm` checks to ensure that the files truly are orphans before removal.
Temporary files	`$INTERMAIL/tmp`	Remove files in this directory when the InterMail servers are not running. System startup is a good time to clean out the `$INTERMAIL/tmp` directory.

File System Usage

System administrators should be notified immediately if any file system exceeds 90% of its capacity.

It is recommended that the Message File system on the Message Store Server (MSS) not be more than about 70% full under normal circumstances. This provides a buffer for unexpected events, such as a sudden, or seasonal, surge of traffic or a network problem that disables POP access for a time.

The `imsysmon` utility monitors the file systems that are key to the InterMail system.

Log Files

InterMail generates extensive log files that contain a running record of InterMail operations, including information about system usage, message flow, and the number of connections to and from InterMail servers. Most events printed to these logs are just informational, but some events may indicate a serious problem or a recurring problem. Periodic inspection of these logs is the only method of detecting certain types of problems and may provide advance notice so that preventive measures can be taken.

Each log event has a severity level that indicates how serious a particular event is and to what extent InterMail has been affected by the event. The InterMail logs use the same severity levels as the `imsysmon` utility. The severity levels used by the log files are described in the following table.

Severity Level	Description
`Notification (Note)`	Does not indicate a problem; is used for informational purposes. Notification messages are printed only to the log file. **Example:** Current disk usage on the monitored file systems, and the time and date of each system monitor cycle. **Priority number:** 0
`Warning (Warn)`	Indicates a more serious event than an error. Warnings are printed to the console as well as the log file. The cause of warning events should be investigated as soon as possible, since these events represent potentially serious problems. **Example:** Disk usage above 80% (configurable) on any of the monitored file systems, failure to open SMTP connections to indicate Urgent or Fatal events, and unhandled signals. SMTP failure states are Warning events; they cannot be given a higher priority since that would trigger e-mail and pager notification, which in turn event would produce a cascade of SMTP failures. **Priority number:** 1
`Error (Erro)`	Indicates a potential problem if this event occurs frequently. Errors are printed only to the log file. **Example:** Failure to find the process `pid` files in the PIDDir (usually in `~imail/tmp`). This is an error, since the process check step is not possible without these files. **Priority number:** 2
`Urgent (Urgt)`	Indicates a pending failure and triggers e-mail notification, and printing to the console and log file. You can also configure InterMail to have these events written to the operating system log files. The cause of urgent events should be investigated and resolved as soon as possible. **Example:** Disk usage above 90% (configurable) on any monitored file systems, client access (SMTP and POP) port banner failure, and process-not-found errors. **Priority number:** 3

Severity Level	Description
`Fatal (Fatl)`	Indicates that a failure is imminent or has already occurred and triggers pager and e-mail notification and printing to the console and log file. You can also configure InterMail to have these events written to the operating system log files. The cause of fatal events needs to be investigated and resolved immediately to prevent service disruption, or to restore service if the system is already down. **Example:** Disk usage above 98% (configurable) on any monitored file systems or MSS not responding to ping indicating that it is down. **Priority number:** 4 or higher.

For details about InterMail logging operations, see Chapter 10.

syslogd Output

The `syslogd` daemon on a UNIX system is used to manage messages concerning operating system, hardware, and daemon problems. You should periodically monitor messages written to the system log files as part of monitoring the UNIX system.

On Sun systems, the system log file is typically `/var/adm/messages`.

See the `syslogd` man page for information on configuring the logging levels for the `syslogd` daemon.

Logging Information to System Log Files

InterMail also gives you the option of having logging information written to the operating system log files. The benefit of this is that it consolidates all the critical information—log events with the severity level Fatal and Urgent—in a single set of log files.

Select this option by setting the `logToSystem` configuration key to `true`:

```
/*/common/logToSystem: [true]
```

cron Jobs

You should monitor the exit status of all `root` and `imail` `cron` jobs. You can do this by reading mail sent to the `root` and `imail` users on a regular basis. Mail is sent if `cron` jobs fail or produce output (`cron` jobs should be set up to produce output only in a failing condition).

You can also monitor the `crond` daemon's log file for the exit status of `cron` jobs. Any line that contains an exit status that is non-zero indicates a `cron` job that has failed. The line indicating that the process has completed will contain `rc=n` at the end in the case of a failed `cron` job, where `n` is the exit status of the job.

Performance Monitoring

During the course of InterMail operations, you should check certain system resources to make sure that the system is not overloaded. InterMail performance can vary greatly, depending on the system's hardware configuration, system usage, additional third-party software, and network connectivity. However, the areas that typically require tuning are:

- Memory and swap usage
- Disk performance
- Network interface performance
- CPU Performance
- Oracle performance
- InterMail server interaction

This section describes each of these performance areas and suggests tools to be used for monitoring. The actual tool used will depend on the UNIX platform on which the InterMail system is running.

Memory and Swap Usage

Although available RAM is a consideration for any client/server application, it may be difficult to determine what is acceptable for total memory usage. However, you can monitor the amount of RAM and swap space in use with `vmstat` or an equivalent utility:

```
paris% vmstat

 procs     memory           page           disk          faults      cpu

 r b w   swap  free  re  mf pi po fr de sr s0 s1 -- --   in   sy   cs
us sy id
 0 0 0  17504 11760   0 147  6 10 14  0  1  1  1  0  0  118 1597  196
3  3 95
```

The important numbers to track are the swap space used and the available free swap space (swap is 17504 and free is 11760 in this example). In the example, 60 % of all swap space is used (`swap` column / (`swap` + `free` columns)) which is within reasonable parameters. If the total used swap space were 75 %, you would want to watch RAM usage. If this level continued, it would indicate a possible problem. If the total used swap space were at 95 %, this would indicate a definite problem.

It is also important to monitor the memory scan rate. A high scan rate will negatively impact the performance of a system. To bring the scan rate down it may be necessary to add memory to the system, or add swap if the system requires backing store for all stack-type data. See your system tuning guide.

Note: Analyze the swap partition to determine if it is large enough to handle the transaction volume. If not, it may be necessary to add more space to the swap partition, expand available virtual memory, or decrease the number of available connections and processes in InterMail.

Disk Performance

InterMail (particularly the MSS and Queue servers) is a disk-intensive application. It is therefore essential to tune disk performance to obtain the best throughput possible.

You can check disk performance with several tools, such as `iostat`.

```
paris% iostat -x 10
                                 extended disk statistics
disk      r/s  w/s   Kr/s   Kw/s wait actv  svc_t  %w  %b
sd3       0.4  0.1    2.0    1.6  0.0  0.0   17.7   0   0
sd4       0.8  0.1   12.3    0.5  0.0  0.0   48.7   0   0
```

In this output, the number of reads per second (`r/s` column), writes per second (`w/s` column), and average service time appear. These numbers will vary; however, as a rule, if the service time is high, or if the reads or writes per second are low, problems may exist in the system.

Another potential problem area is access times. For instance, if it takes too long for a connection to the MSS to access the hard drive, problems may ensue.

To check access times for a given host, issue a `sar` command:

```
paris% sar -d  10 10

SunOS paris 5.5.1 Generic  sun4m    1/27/99

14:25:03   device        %busy   avque   r+w/s  blks/s  avwait  avserv
14:25:13   sd0              81     0.8     101    1978     0.0     8.1
           sd1               0     0.0       0       6     0.0    10.3
14:25:23   sd0              81     0.8      97    2024     0.0     8.4
           sd1               0     0.0       0       5     0.0    18.9
```

With this output, you can observe problems or bottlenecks on a particular hard drive. You can also determine CPU performance based on the average number of requests that are outstanding (`avque`) and the average time taken to process a request (`avwait`). In general, a high `avque` time indicates problems and possible file contention issues where the system is exhibiting poor disk performance.

Regardless of which tool you use, the goal is to keep the service time, % wait, and number of waiting processes as low as possible.

If a particular device is identified as a performance bottleneck, you must take action to remove the bottleneck. These actions might include:

- Analyze the device to ensure that there are no hardware problems and that the cache, or other options associated with the device, are being used effectively.

- Check if the device has become fragmented. If it has, run a utility to reduce fragmentation.
- Analyze the data that is assigned to the device. Check if some of the data can be moved to a less busy device?
- Add spindles to the device (assuming here a virtual device such as a Veritas filesystem). InterMail disk access tends to be very random, and I/O is in relatively small blocks, so the more spindles with which it has to work, the better).
- Check if the device houses Oracle indexes. If it does, reorganizing the indexes to compress free space might help.

CPU Performance

To analyze the CPU load, issue the following command:

```
paris% sar -u 10 10

SunOS paris 5.5.1 Generic  sun4m    01/27/99

20:58:39    %usr    %sys    %wio   %idle
20:58:49       1       2       0      97
20:58:59       1       2       0      97
20:59:09       1       2       0      98
```

Generally, you should calculate load from user processes and system (kernel) processes. If the percentage of CPU load (`%usr` + `%sys`) is greater than 95 %, this is a concern; however, this may be short-lived behavior. When CPU load is excessive for at least 30 minutes, this indicates a problem. If these CPU problems last 2 weeks or more, it may be appropriate to add additional CPUs or another host of the same type to the InterMail system.

Another indicator of CPU capacity would be the run queue statistics (viewable with the uptime command). A high run queue indicates that the system is experiencing a bottleneck somewhere. CPU and disk statistics should be checked.

Networked Resources

Problems with networked resources are similar to those with disk space, but with more difficult solutions. Most of the time, network input/output capacity is not a problem. If you observe network problems, first make sure that the network is clean (that there is no unexpected traffic from other devices) and that all segments are working correctly.

For example, enter `netstat -i 5` to display network traffic on all network interface cards for the past 5 seconds:

```
paris% netstat -i 5
input   le0       output           input  (Total)    output
packets errs  packets errs  colls  packets errs  packets errs  colls
525829  0     94851   1     25     630894  0     199916  1     25
8       0     1       0     0      8       0     1       0     0
11      0     2       0     0      11      0     2       0     0
```

The `netstat` output can be used as a gross grained indicator of network health. In general:

- The `errs` should be essentially 0. Any substantial errors would point to a hardware problem with the associated interface.
- The collision rate should be under 5% (that would be (`#coll/ #outputPackets`)*100). A high collision rate is an indication of excessive bandwidth utilization. If investigation proves that there is no "rogue" user occupying the bandwidth, the capacity of the network should be expanded.

Monitoring Oracle Databases

InterMail's Message Store database and Integrated Services Directory (ISD) are Oracle databases. These databases require regular monitoring and maintenance in order to function at peak capacity.

This section tells you how to perform some routine monitoring functions, as well as how to check for problems that can adversely affect the performance of InterMail, such as fragmented database indexes or lack of available disk space for tablespaces.

Reorganizing Database Indexes

Index reorganization improves performance by eliminating the fragmented free space that builds up in database indexes over time. Eliminating index fragmentation also increases the performance of the database. You reorganize indexes using `imdbindexreorg` on the host where the Message Store database runs. The `imdbindexreorg` command must run during a maintenance window when services that modify the database are not running.

The syntax for the `imdbindexreorg` command is:

```
imdbindexreorg -tablespaces <filename> -bloat <percentage>
```

Where:

`<filename>`	Specifies the name of the file in which the list of tablespaces that require backup after the `imdbindexreorg` command runs is to be written.
`<percentage>`	Specifies the sensitivity of the index reorganization by setting a high-water mark for the percentage of index space used. When an index has a bloat percentage greater than the specified value, the system reorganizes that index. In this way, the system can reorganize a critically bloated index and leave the other indexes alone.

The `imdbindexreorg` reads the value of `db/ indexReorganizationTimeLimitMinutes`, which specifies the amount of time (in minutes) that the `imdbindexreorg` command is to run. If the `db/ indexReorganizationTimeLimitMinutes` configuration key is too high, index

reorganization may take longer than the maintenance time window. If this key is too low and `imdbindexreorg` runs too infrequently, the command will not have enough time to complete defragmentation on all the indexes that need it. The `-timelimit` argument (in minutes) can override this key.

For more details about `imdbindexreorg`, see the *InterMail Mx Reference Guide*.

Checking Free Space

The challenge in managing space in databases is to determine when a tablespace needs to expand. InterMail provides two tools to check on Oracle database usage and specifically estimate when the available free space will need to expand: `imdbspacecheck` and `imdbspacequickcheck`.

imdbspacecheck

The `imdbspacecheck` utility monitors the remaining free space in an Oracle database, estimates when more space will be necessary, and logs an Urgent message when space needs to be added. It calculates when more space needs to be added by checking all possible permutations in the Message Store database, as well as the historical growth of tables. Historical information about database growth is maintained in the log file `imdbspacecheck.<DB_instance>.hst`. The `imdbspacecheck` utility can run while InterMail is in full operation and it can be set to run regularly.

Two conditions cause `imdbspacecheck` to log an Urgent message:

- An extent is equal to or greater than the percentage set by the `db/extentFullWarningThresholdPercent` configuration key. This indicates the possibility that this object would not be able to add an extent.
- An object's last extent will probably fill within the time specified by `db/extentGrowthAllowanceDays`, as a result of which the object may not be able to add an extent. This option is effective only in the case where `imdbspacecheck` is run regularly, creating a usable history file.

A typical `imdbspacecheck` command looks like the following:

```
imdbspacecheck -thresholdpercent 70 -thresholddays 60 -dir MSS
```

In this example, `imdbspacecheck` will report any combination of database growth that will prohibit an extent that is currently at 70 % (set by `-thresholdpercent`) to add an additional extent within 60 days (set by `-thresholddays`).

Note: The parameters used in `imdbspacecheck` take precedence over the values of `db/extentFullWarningThresholdPercent` and `db/extentGrowthAllowanceDays`.

---

imdbspacequickcheck

Since `imdbspacecheck` is memory-intensive and can take a long time to run, InterMail provides a second command, `imdbspacequickcheck`, for quick checks of free space in a database. In contrast to `imdbspacecheck` which checks every possible combination of table growth, and provides in-depth future projections, `imdbspacequickcheck` performs a cursory check of current conditions.

The `imdbspacequickcheck` utility searches the database to determine what the result would be if any tables in the database needed an additional extent in isolation. If it finds that any tables will have a problem growing, it reports this information. The `imdbspacequickcheck` utility can thereby warn of an impending space crisis.

Like `imdbspacecheck`, `imdbspacequickcheck` logs an Urgent message if it finds an object (table or index) whose last extent is over the specified high-water mark (90% full by default), and the object cannot grow. Tables and indexes that are known not to grow are exempt from this check.

In typical system, you should schedule `imdbspacequickcheck` as a `cron` job, executed approximately once every 30 minutes. Since `imdbspacequickcheck` imposes an insignificant load on the monitored system, it can be run much more frequently than `imdbspacecheck`.

A typical `imdbspacequickcheck` command looks like:

```
imdbspacequickcheck -thresholdpercent 70
```

In this example, `imdbspacequickcheck` will report any database growth that will prohibit an extent that is currently at 70% (set by `–thresholdpercent`).

Expanding Tablespaces

Once you determine that a particular database needs to grow and that there is available hard drive space to accommodate this increase, run the `imdbspacegrow` command to expand the available tablespace.

Example:

```
imdbspacegrow -dryrun
```

When executed with the `-dryrun` argument, `imdbspacegrow` does not actually perform the database expansion but instead logs a description of what would happen if the database were enlarged. It is a good idea to perform a dry run of `imdbspacegrow` and review the results before actually adding the space.

Example:

```
Database name:       IMM1
Database access:     local
ORACLE_HOME:         /disk2/oracle/7.3.3
Ready to proceed?    (Y/N) [yes]
```


Once you confirm the database status information, a list of tablespace names appears, and you can choose the name of the tablespace you want to expand.

```
The database is partitioned into 5 tablespaces.
TABLESPACE LIST
1) SYSTEM
2) TEMP
3) TB1
4) TB2
5) TB3
Please select a tablespace from the list by number: [5]
You selected tablespace TB3.
After you make a selection, InterMail will report the current status of
the tablespace that you chose.
For your information, the tablespace TB3 is currently composed of the
following files:

FILE NAME                                       SIZE IN BYTES
/disk2/oracle/admin/D1ALE733/TB3.dbf            32522240 bytes
Ready to continue? (Y/N) [yes]
```

The utility prints information about the recommended sizing for the additional file and then prompts you to specify the size of the new tablespace file:

```
Enter the size of the file to be added to the tablespace: [0] 32522240

------------------------------------------------------------
```

Next, you are prompted to enter the full path for the new file. Typically, Oracle database files use the `.dbf` extension.

```
Enter full path name of new file:
/disk2/oracle/admin/D1ALE733/TB3_2.dbf
A file named /disk2/oracle/admin/D1ALE733/TB3_2.dbf of size 32522240
bytes will be added to tablespace TB3.

Proceed? (Y/N) [yes]

------------------------------------------------------------
```

At this point, InterMail will add the new file to the tablespace. Since this example specified `-dryrun`, the results of the operation appear, but the file is not written to disk.

```
Oracle added a new file named "/disk2/oracle/admin/D1ALE733/TB_3.dbf"
to tablespace "TB3". The actual size of the file is 0 bytes.

NOTE: this differs from the size you requested (32522240).

The tablespace TB3 should be hot backed up now, since its structure has
been altered in a significant way.
```

It is extremely prudent to do a hot backup of InterMail immediately after creating more tablespace. Therefore, it is advisable to run `imdbspacegrow` during a maintenance window or other nonpeak time.

Viewing Database Information

When `imdbspacecheck` or `imdbspacequickcheck` warns of low tablespace, you can run `imdbspacereport` to obtain a snapshot of space allocation in the database, and thereby determine the nature of the possible space crisis. The `imdbspacereport` utility prints out information about the database, including the exact Oracle version used, the value of all database parameters, and information about space usage in the database.

To run `imdbspacereport`, enter:

```
imdbspacereport -dir MSS
```

The argument `-dir MSS` indicates the name of the Message Store database instance for which the report is to be produced.

Performance Tuning

In InterMail, performance tuning is typically an iterative process in which you observe response times, available memory, and other system activity, modify certain parameters, and then observe again. You may also do tuning in response to bottlenecks, problems, or events that have been logged by InterMail.

Tuning Based on Observed System Performance

If the average access time to the MSS is unacceptable, try decreasing the value of `mta/timeoutServerDelivery`. This causes mail to spool on the Message Transport Agent (MTA) more, thus reducing the burden on the MSS.

When the MTA appears to be running slowly, try the following:

- Decrease the value of the `mta/cacheLimitInKB` configuration key.
- Increase the value of the `mta/maxDirectDelivery` configuration key
- Increase the value of the `mta/maxDirectKB` configuration key.
- Check to see if the volume has peaked recently, or if any other processes on the machine are starving the disk/memory or CPU.

Note: In order to improve access times, it may be necessary to decrease the number of available connections or add new hardware.

Tuning Based on Common Log Events

If connections are timing out on the POP server (which you can determine by looking for `NioMaxConnection` events in the `popserv.log` file), try the following:

- Increase the value of the `popserv/maxSessions` configuration key.
- Increase the number of Files Descriptors in a 3:1 ratio to the value specified in the `popserv/maxSessions` configuration key.

If connections are timing out on the MTA (which you can determine by looking for `NioMaxConnection` events in the `mta.log` file), try the following:

- Increase the value of the `mta/smtpDeliverNumThreads` configuration key.
- Increase the value of the `mta/smtpAcceptNumThreads` configuration key.
- Increase the value of the `mta/fileDescriptors` configuration key.

System Maintenance

System maintenance consists of regular tasks necessary to maintain system operations for extended periods. The tasks described in this section should be carried out periodically to ensure a healthy InterMail system.

Backing up Message and Directory Information

For reliable mail service, it is important to back up critical components of the InterMail system regularly. Backup efforts should focus primarily on message information stored in the Message File system and Message Store database, and account information stored in the Directory database. A sound backup process should be implemented and tested at all sites.

For details about backup and recovery, see Chapter 11.

Expanding the Message File System

When the Message File system is running low on space, you can it by adding additional filesystems.

To expand the Message File system:

1. Either mount an additional external storage device or symbolically link an existing partition under the directory defined in the `mss/messageFilesDir` configuration key.

   Although it is possible to add an external mounted device on an additional host, for performance reasons it may be desired to add the additional device on the host that currently contains the Message File system.

2. Run `imbucketscreate` once on the new subdirectory, designating the hierarchy and number of leaf directories. (For more information on `imbucketscreate`, see the *InterMail Mx Reference Guide*.)

3. Run `imbucketscreate` again, this time with no parameters, to balance the load among all directories in the Message File system, including the newly created subdirectory.

Example:

```
paris% cd msgfiles
paris% ls
messages        buckets      buckets.dir
paris% ln -s new_message_fs messages_2
paris% pwd
/vol1/imail/msgfiles
paris% ../lib/imbucketscreate messages_2 10 10
imbucketscreate: creating 100 directories under msgfiles_2.
imbucketscreate: finished creating 100 directories under msgfiles_2.
paris% ls
messages        messages_2      buckets      buckets.dir
paris% more buckets.dir
messages
messages_2
paris% ../lib/imbucketscreate
```

This example adds a new filesystem called `messages_2` to the InterMail `msgfiles` directory.

The initial invocation of `imbucketscreate` creates the Message File system directory structure under `messages_2` with ten leaf directories and ten leaf subdirectories. It also writes an entry to the `buckets.dir` file that contains entries for all directories in which InterMail can store messages.

Finally, `imbucketscreate` runs again, without arguments, in order to identify those directories with the most free space (this includes the newly created directories). The list of the most empty directories is stored in the file buckets, where it is accessed by the MSS.

Balancing the Message File System

The `imbucketscreate` utility distributes message files evenly in the directories of the Message File system, thereby improving performance when the system reads message files from, or writes message files to, individual directories.

The `imbucketscreate` command should be run as a periodic `cron` job. For more information on the `imbucketscreate` utility, see the *InterMail Mx Reference Guide*.

Defragmenting the Message File System

Because of the transient nature of the message files stored in the Message File system on the MSS, the file system tends to become fragmented. This can seriously reduce the amount of usable file system space. Use the utilities of by your file system management software to check the amount of fragmentation and recapture the disk fragments.

For example, if you are using the Veritas file system manager, use the `fstyp -v` command to display the amount of fragmentation for a file system. Then, if required, run the Veritas file system manager command `fsadm` regularly to recapture small disk fragments.

Archiving Log Files

Log files must be pruned during system maintenance, but it is strongly recommended that you back these files up before they are deleted in order to create an archive of operational data.

Rollover is a means of controlling the size of log files, and of archiving old log files to secondary storage without disrupting running applications. When a log file rolls over, the system closes and archives it, creates a new log file, and continues logging to the new file.

To specify how many times per day log files are to roll over, use the configuration key `rolloversPerDay`.

Example:

```
/paris/common/rolloversPerDay: [2]
```

In this example, log files roll over twice a day. A value of `0` disables rollover altogether.

In calculating rollover times, the system always uses Greenwich Mean Time (GMT). By default, the first rollover period of a day starts at midnight (00:00:00 GMT). However, you can change this by setting the configuration key `rolloverTimeZero` (expressed in seconds) to establish a time offset from GMT.

Example:

```
/paris/common/rolloverTimeZero: [18000]
```

In this example, the rollover period will begin at 05:00:00 GMT (18000 seconds or 5 hours after 00:00:00 GMT).

Tips to Manage Log Files

To maintain and organize log files efficiently, follow these guidelines:

- Set the rollover time for log files to a reasonably quiet period.
- Compress and/or archive log files regularly. For this, you can write a script that runs as a `cron` job.
- Leave the event log files for the last few days uncompressed since you will probably be accessing them frequently. Archive log files older than a week.

Handling Sidelined Mail

When you enable InterMail's sidelining feature, the system does not deliver or bounce incoming mail that meets the conditions specified. Instead, it moves the mail to the `spool/sidelined` directory. InterMail does not delete sidelined mail automatically; if you use sidelining, it is important that you check this directory on a regular basis to prevent sidelined messages from piling up.

To see how many mail messages have been saved to the `sidelined` directory, enter:

```
find spooldir/sidelined -type f -name "*Control" | wc -l
```

In this command `spooldir` is the directory defined by the configuration variable `/*/common/spoolDir` (typically `$INTERMAIL/spool`).

The InterMail system moves the Control, Header, and Body files for a sidelined message into a numbered bucket (subdirectory) in the `sideline` directory.

To determine why a Control file has been placed in the `sideline` directory, search the file for `Sideline-Reason:`. If the message looks like a legitimate message, you can resubmit it using the `immsgprocess` command. Otherwise, you can delete the Control, Header, and Body files.

For more information on sidelining mail options and the review and reprocessing of sidelined mail, see Chapter 5 and Chapter 7.

Handling Mail in the Errors Directory

Most messages that the MTA receives are either delivered to their intended recipients, or bounced when the MTA attempts delivery to a local domain with an unknown recipient. If the MTA is unable to return a message because the original sender's return address is invalid, all three message components (Control, Body, and Header files) go into the `spool/errors` directory. It is important that you check this directory regularly because it can fill up very quickly.

To see how many mail messages have been saved to the `errors` directory, enter:

```
find spooldir/errors -type f -name "*Control" | wc -l
```

In this command, `spooldir` is the directory defined by the configuration key `/*/common/spoolDir` (typically `$INTERMAIL/spool`).

To determine why a Control file has been placed in the `errors` directory, search the Control file for `Error:`. If the message can be corrected, you can fix and resubmit the message using the `immsgprocess` command. If it cannot be fixed, you can delete the Control, Header, and Body files.

If for some reason the `errors` directory grows very large, it is best to re-create it altogether. In that case, review the Control files and enter:

```
mkdir newerr
# Ensure that ownership/permissions of newerr are the same as errors
mv errors olderr; mv newerr errors
rm -rf olderr
```

Processing Bad Messages

If for any reason messages cannot be retrieved by the POP server, InterMail puts them in a `.ERROR` folder in the Message Store database. These messages should be reviewed and cleaned up regularly.

To generate a list of “bad” messages, run `imbadmsglist`. If a message can be fixed, you can fix it and run `imbadmsgfix`. If it cannot be fixed, you can delete it with `immsgdelete`.

10

Monitoring Tools

InterMail provides several monitoring tools to help you track system conditions and performance. This chapter discusses those tools. System monitoring and maintenance tasks are covered in Chapter 9 in this manual.

This chapter covers the following topics:

- Simple Network Management Protocol (SNMP)
- `imsysmon` utility
- Event logging

SNMP

Simple Network Management Protocol (SNMP) can monitor network and system traffic statistics and reportable parameters. In the InterMail system, SNMP provides information such as the number of connections to the POP server since the server started, the total number of messages in the Message Transport Agent (MTA), and the total number of messages in the Message Store Server (MSS). The SNMP system "samples" this information and sends it to output on the user-defined SNMP monitoring station.

This information is particularly useful in InterMail because you can view it in real-time, without running scripts or parsing logs. Standard SNMP monitoring stations (for example, HP OpenView) can display information about the present state of a server, as well as archived information.

For information on the InterMail SNMP server, and for a complete list of reportable parameters available for SNMP monitoring, see the *InterMail Mx Reference Guide*.

Configuring an SNMP Server and Monitoring Station

In InterMail, any remote host can view SNMP information. For example, although InterMail runs on the UNIX platform, an SNMP monitoring station can run on a Windows NT machine that receives InterMail information from SNMP.

The process of configuring SNMP to run on a monitoring station involves several steps both on the InterMail host running the SNMP server and on the machine running the SNMP monitoring station.

SNMP Server Configuration

When configuring the InterMail SNMP server for operation, adjust the following keys as necessary:

- The `/*/common/trapMask` configuration key, which lists the severity level of events that SNMP reports. Valid values for `trapMask` are "notification," "warning," "error," "urgent," and "fatal."

  Note: As with all configuration keys, you can set the `common/trapMask` configuration key on a server-by-server basis, by adding a configuration key for each server (such as `mta/trapMask`). By specifying severity levels for all servers, you can reduce the amount of unwanted information displayed on an SNMP monitoring station.

- The `/*/snmp/masterAgentHost` configuration key, which specifies the name of the host that is running the SNMP Master agent.

  The SNMP thread on each server maintains a queue of events (traps) that it dispatches periodically. This activity takes place when the thread is not busy processing requests. The following configuration keys relate to the queuing and dispatching activities of the SNMP thread:

- The `/*/snmp/trapQueueSize` key, which specifies the number of events the trap queue can track and store. Valid values range from 512 to 4096. After reaching the value specified by this key, the system adds new events to the "bottom" of the queue and no longer tracks events that are at the "top" of the queue. The default size is 1024 (only 1 KB of memory for the queue itself). If the queue were to fill up, some traps might not actually make it to the monitoring station, but the system would still report the events to the appropriate server logs.
- The `snmp/eventMaxWait` configuration key, which specifies the maximum time (in milliseconds) that the SNMP thread will wait for a request from a management station. Valid values range from 3000 and 10000 (3 to 10 seconds) with a default of 3000 (3 seconds).

Here is an example of the configuration keys that configure the SNMP server:

```
.....
/*/common/trapMask:
    [fatal]
    [urgent]
/*/common/trapQueueSize: [1024]
/*/snmpdm/eventMaxWait: [5000]
......
/venus/snmpdm/masterAgentHost: [venus]
......
```


SNMP Monitoring Station Configuration

The process of configuring an SNMP monitoring station varies depending on the particular monitoring software. However, the following general steps always apply:

1. Find the `Ovtb` directory for the SNMP monitoring station, and create a subdirectory under it (for instance, `swcom`).
2. Establish an FTP connection to the InterMail host running the SNMP server, find the directory that contains the MIB (Management Information Base) files (with the extension `.my`), and copy these files to the new subdirectory.
3. Compile these files into `*.mib` files as instructed in the documentation provided with the SNMP monitoring station software.
4. Identify the host that is running the InterMail SNMP server to the SNMP monitoring station.

The imsysmon Utility

You can monitor InterMail servers by telnetting to ports or pinging. However, the `imsysmon` command can provide time-sensitive monitoring of servers. The `imsysmon` command checks the overall health of the InterMail system and reports status to a log file. When `imsysmon` discovers events of a user-definable severity level (see Chapter 11 for more information on severity levels), it will optionally send an e-mail and/or page to a list of specified addresses. In this way, real-time monitoring can continue 24 hours a day 7 days a week.

The imsysmon.ini file

Before running the `imsysmon` command, you should configure `imsysmon` to specify the location of system files, which servers to monitor, where to write log files, and who will receive e-mail and/or pager alerts. Modifying the `imsysmon.ini` file (located in `$INTERMAIL/config`) configures the `imsysmon` command.

The `imsysmon.ini` file contains four sections:

- IMSYSMON—variables controlling the running environment of `imsysmon`
- INTERMAIL—variables setting the location of InterMail
- ORACLE—variables setting the location for Oracle instances
- DEFAULT—variables setting the threshold limits for parameters that affect the operation of `imsysmon`

The following four subsections describe the variables and meaning of values found in the `imsysmon.ini` file.

IMSYSMON: Setting the Running Environment for imsysmon

The IMSYSMON section of the `imsysmon.ini` file controls how `imsysmon` should behave—where it should expect certain information to exist, to whom it should send

e-mail and pager notifications, and threshold levels for certain routine operations. The variables in the IMSYSMON section and their meaning are as follows:

`BannerTime`	Number of seconds to wait for a banner to appear.
`MailHosts`	List of mail hosts that can deliver e-mail and pager notifications. It is important to specify at least one mail host that is not in your InterMail system, in the event that InterMail MTAs are not available.
`MailRcptTo`	Email address(es) that will receive a message when a Fatal or Urgent event occurs.
`MaxConnAttemptNote`	Maximum number of attempts in connecting to a port.
`MinErrDiskUsage`	Minimum amount of disk usage before reporting an “Error” event.
`MinFatlDiskUsage`	Minimum amount of disk usage before reporting a Fatal event.
`MinUrgtDiskUsage`	Minimum amount of disk usage before reporting an Urgent event.
`MaxErroFreeExtents`	Minimum percentage of free extents available before reporting an Error event.
`MaxFatlFreeExtents`	Minimum percentage of free extents available before reporting a Fatal event.
`MaxUrgtFreeExtents`	Minimum percentage of free extents available before reporting an Urgent event.
`PageRcptTo`	E-mail address(es) that will receive a message when an Urgent event occurs.
`TimeOut`	Time (in minutes) that `imsysmon` will wait for a task or a probe to complete, before timing out and reporting an error.

INTERMAIL: Setting the Running Environment for InterMail

The INTERMAIL section of the `imsysmon.ini` file controls the interaction of `imsysmon` with InterMail. The variables in the INTERMAIL section and their meaning are as follows:

`UserName`	Name of the InterMail user. This will typically be the same value as the `common/commonUser` configuration key.

ORACLE: Setting the Running Environment for Oracle

The ORACLE section of the `imsysmon.ini` file consists of one variable listing the Oracle instances that the `imsysmon` command will monitor.

`OracleSID`	List of Oracle instances to monitor. Typically, there will be at least two instances: one for the Message Store database and one for the Integrated Services Directory.

DEFAULT: Setting the Threshold Limits for imsysmon

The DEFAULT section of the `imsysmon.ini` file determines which events to monitor. This section also determines what severity level `imsysmon` should consider a particular error when reporting it to a log file, displaying it in standard error, and/or sending a mail or page (if the event is Urgent or Fatal). Whereas the IMSYSMON section sets thresholds in percentages or numbers, the DEFAULT section sets the severity level that `imsysmon` will report when the item reaches the threshold percentage or number.

The following values represent severity levels in the DEFAULT section:

`0`	Notification
`1`	Warning
`2`	Error
`3`	Urgent
`4`	Fatal

For each of the following variables in the DEFAULT section, you can set a value for which severity level to report. If the user does not wish `imsysmon` to monitor and report a particular event, they can “comment out” the event.

`FatlDiskSpace`	The percentage of disk usage is at least `MinFatlDiskUsage`.
`FatlFreeExtents`	The number of free extents is less than `MaxFatlFreeExtents`.
`OraInstPingFail`	An Oracle instance (that the `OracleSID` variable defines) is not responding.
`ProcNotFound`	An InterMail server is not responding to a ping.
`FatalMsgsFoundInLog`	Fatal messages found in server logs.

`AlarmTimeout`	A timeout has occurred while waiting for a task.
`ArchivePathNotFound`	Could not find the Oracle archive path for a specific instance.
`DiskSpaceCheckFail`	An error has occurred while trying to perform a disk space check.
`LogFileOpenFail`	Could not open the `imsysmon` log file.
`UrgtDiskSpace`	The percentage of disk usage is equal to or greater than `MinUrgtDiskUsage`.
`OraTabOpenFail`	Cannot open or read the `oratab` file.
`PortBannerFail`	A failure to connect to server port expecting a banner has occurred.
`UrgtFreeExtents`	The number of free Oracle extents is less than or equal to `MaxUrgtFreeExtents`.
`UrgtMsgsFoundInLog`	Urgent messages found in server logs.
`CheckFreeExtentsFail`	There has been a failure to check for free Oracle extents.
`ErroFreeExtents`	The segment for an Oracle instance is low.
`IMAPQuitError`	The IMAP server rejected a “QUIT” command.
`IMDbSpaceChecker`	There has been a failure to run `imdbspacechecker`.
`IMDbLogParsingFailure`	There has been a failure to parse an InterMail log file.
`IMServPingFail`	There has been a failure to run `imservping`.
`LogSegmentCreationFailure`	There has been a failure to create a log segment.
`LogSegmentReadFailure`	There has been a failure to read a previously saved log segment.
`POPQuitError`	The POP server rejected a “QUIT” command.
`SendMailAttemptFail`	There has been a failure to notify an event via Ssendmail.
`SMTPDataError`	SMTP server rejected our “DATA” address.
`SMTPFromError`	SMTP server rejected our “MAIL FROM” address.
`SMTPRcptToError`	SMTP server rejected our “RCPT TO” address.

`SMTPMessageBodyError`	SMTP server rejected the body of our message.
`SMTPNotifyFail`	Failure to notify of event via e-mail.
`SMTPQuitError`	SMTP server rejected our “QUIT” command.
`SocketSMTPMailFail`	Failure to notify of event via e-mail.
`SockOpenFail`	Unable to create a socket to communicate with one of the servers.
`UnhdlSignal`	An unaccounted-for signal has occurred.
`WarnDiskSpace`	The percentage of disk usage for a file system is at least `MinErrorDiskUsage`.
`BadPortBanner`	Received a bad port banner
`NoteDiskSpace`	A notification message on disk space.
`NoteFreeExtents`	A notification message on free extents.
`NoteOraInstResponded`	A notification message that an Oracle instance has started.
`OraSidNotFound`	Oracle instance (`Oracle_SID`) not found.
`OraHomeNotFound`	Oracle home directory for specific instance not found.
`ProcFound`	Found a process.
`PortBanner`	Banner found for a server port.

Modifying the imsysmon.ini file

The `imsysmon.ini` file is large and offers many options for configuration. The following example shows how to set some sample variables in this file.

```
[IMSYSMON]

# Maximum amount of secs. to wait for a banner.
BannerTime = 20

# MailHosts - The list of hosts that will be used by imsysmon to send
#             email messages. Add one host per line between the line
#             containing <<EOT and the line containing EOT.
MailHosts = <<EOT
EOT

# MailRcptTo - The email adress(es) that will receive a message when
#              an urgent event happens.
MailRcptTo = <<EOT
EOT

# LogEventFilters - List error mnemonics and associated args to describe
#                   the log event(s) that you do *not* want to be notified
#                   about.
LogEventFilters = <<EOT
MtaControlFileWriteFailed
EOT

# EmailFromAddr - The address imsysmon uses to send its email messages
#                 from. This account is NOT created automatically. It
#                 must exist and be active for imsysmon email notification
#                 to work.
EmailFromAddr = imail

# The maximum percentage the directory cache can be out of sync with the
# directory database before raising an "urgent" message.
OutOfSyncThreshhold = 90

# The maximum number of attempts in connecting to a port.
MaxConnAttemptNote = 5

# MinErrDiskUsage - The minimum amount of disk usage before raising
#                   an "error" message.
MinErrDiskUsage = 80

# MinFatlDiskUsage - The minimum amount of disk usage before raising
#                    an "fatal" message.
MinFatlDiskUsage = 98

......
......

[INTERMAIL]

# UserName - The name the the InterMail user.
UserName = imail

[ORACLE]

# OracleSID - The list of Oracle instances that will be monitored.
OracleSID = <<EOT
IMM1
IMD1
EOT

[DEFAULT]

# Percent of disk usage is at least MinFatlDiskUsage.
FatlDiskSpace = 4
......
```


Figure 16 Sample imsysmon.ini file

Running imsysmon

After configuring the `imsysmon.ini` file, invoke the `imsysmon` command on a command-line by typing `imsysmon`. However, in order to run `imsysmon`, you must be `root`.

Note: To run the `imsysmon` utility, you need to set the `ORACLE_USERINFO` environment variable.

For a complete description of available syntax for `imsysmon`, see the *InterMail Mx Reference Manual*.

The following example shows the execution of `imsysmon` and some possible problems that it may report.

```
paris% ./imsysmon
imsysmon:  Monitor procedure started for host paris on Fri Jan 08
14:46:15 PDT 1999
           Initializing......

imsysmon:  Initialization complete. Running with these values

        HostName               paris
        OpSys                  SunOS
        ImailUName             imail
        OracleUName
oracle
        InterMailDir           /vol1/imail
        VerboseMode            0
        LogMode                1
        LogFile                /vol1/imail/log/
imsysmon.paris.19990108144615.log
        MailDeliveryHosts      earth:7131
        WaitTime               2
        Environment: imail
        Environment: oracle
imsysmon:  monitoring InterMail modules: mss mta popserv
--
Checking module mss...
--
imsysmon:  FATAL!  19980508144624:  mss does NOT respond to connection
requests.
--
Checking module mta...
--
…
```

Figure 17 Running Imsysmon

Based on the output shown in Figure 17, `imsysmon` creates a log file (`imsysmon.venus.19980508144615.log`) and sends an e-mail to the addresses that `MailRcptTo` specified.

InterMail Event Logging

This section describes the various types of log files and log file formats, and explains how to read log data.

InterMail generates extensive log files that contain a running record of InterMail operations, as well as information about system usage, message flow, and the number of connections to and from InterMail servers. You can configure how much information is to be logged.

In addition to being an important tool for system monitoring, log files are very useful for debugging since they allow you to retrace the steps that led to an event or problem.

Types of Log Files

InterMail generates three kinds of log files:

Log File Type	Suffix	Contents
Event log	`.log`	Error, warning, and informational messages recorded as the events occur.
Statistics	`.stat`	Statistical information for a period of time.
Trace	`.trace`	Diagnostic information recorded as events occur. Trace files are typically turned on at the direction of the InterMail technical support staff.

You should regularly review the `.log` files as part of routine system monitoring.

InterMail writes all log files to the same log directory. To specify the log directory, use the `common/logDir` configuration key. The value must be a valid path name, relative to the InterMail home directory. For example:

```
/*/common/logDir: [log]
```

For more information about the configuration keys mentioned in this chapter, see the *InterMail Mx Reference Guide*.

Event Log Files

All InterMail server processes (such as `mss`, `mta`, and `popserv`) are capable of writing events to `.log` files. Each server process maintains a current log file that can be rolled over after a configurable period of time. The server then starts a new log file.

References to current log files appear in files whose names include a server name and file type (for example, `popserv.log` contains a reference to the current log file for popserver). Similarly references to archived log files appear in files whose names include a server name, a host name, a date/time stamp, and the log type (for example, `popserv.lyons.199805050000-0700.log`).

The time stamp in log files can be in local time or Greenwich Mean Time (GMT). If you want time stamps in local time, set the configuration key `gmtLogTimes` to `false` (the default value). If you want time stamps in GMT, set the value of the `gmtLogTimes` key to `true`.

If the time stamp is in local time, the log filename will include a time zone offset; for example, `popserv.lyons.199805050000-0700.log` where `–0700` is the time zone offset.

By default, the system does not print event logs to `stderr`. To turn this capability on, and have all event logs of severity level Warning and higher to `stderr`, set the value of the configuration key `servWarnToStderr` to `true`:

```
/*/common/servWarnToStderr: [true]
```

Statistics Files

Statistics files contain statistical information for a configurable period of time. You can use these statistics to measure system performance.

Each server generates a statistics file. Initially, the statistics file reports all the statistics that the server generated. You can use the `reportParamsInterval` configuration key to specify how frequently you want the `.stat` file generated. The default value for this key is 180 seconds. After the specified interval, the server checks whether any reportable parameters have changed and reports any changes.

Statistics files follow the same naming convention as event log files, but with a `.stat` extension. Therefore, for example, the filename for a statistics file might be `popserv.paris.199805050000-0700.stat`.

Trace Files

Trace files contain diagnostic information recorded as events occur. This is an important option for debugging and measuring system performance, since it tracks the flow of messages through the InterMail system. You generally turn trace files on only at the direction of the InterMail technical support staff.

If the message tracing capability is turned on for a server, it adds an entry to its log file whenever it processes a message. These message tracing entries allow you to trace messages through the system, and help to track down problems with messages.

To set message tracing, use the configuration key `messageTracing`. You can set this key independently for each server.

You can set the level of detail (verbosity) of the `.trace` file output using the configuration key `traceOutputLevel`. For example:

```
/paris/mta/traceOutputLevel: [rme=1,sock=2]
```

By default, output to the `.trace` file does not print to `stderr`. To turn this capability on for any InterMail application, you can set the value of the configuration key `traceToStderr` to `true`, or you can launch the program with a `-traceToConsole` option.

Log File Formats

This section explains the formats of the event log file, statistics file and trace file.

Event Log File Format

Inside each event log file, numerous log “events” appear. The format for entries in an event log file is:

```
<logEntry>::= <date> <time> <host> <program> <pid> <lwp> <thread>
<severity> ";" <event>
```

Figure 18 shows a typical event log in an event log file.

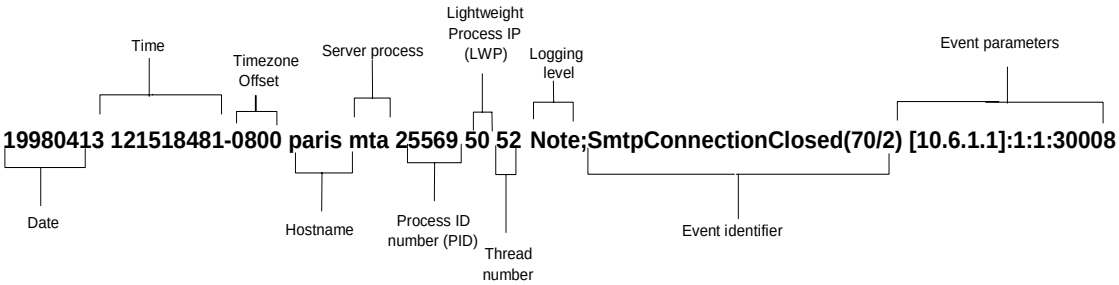

Figure 18 Event log format

The event log shown in Figure 22 provides a notification that, on 13 April 1998, at 12:15 PM local time, the MTA had an SMTP connection close on the host named `paris` from the IP address 10.6.1.1. one second after connection time. The client sent 1 message of length 30008 bytes.

Event logs consist of the following fields:

Date — The date stamp in YYYYMMDD format.

Time — The time stamp in HHMMSSmmm format, where mmm is in milliseconds, in local time or GMT. If the time stamp is in local time, the timezone offset also appears.

Host name — The name of the machine where the server resides.

Server Process — The name of the process that logged the event (such as mta or popserv).

Process ID number (PID) — The standard PID of the operating system.

Light Weight Process ID (LWP) — The lightweight PID; not supported by all UNIX platforms.

Thread Number — The thread number for a multi-threaded process.

Logging Level — The logging level for the log event. This level indicates the level of effect that the event will have on a server or process. Logging levels are Fatal (Fatl),

Urgent (Urgt), Error (Erro), Warning (Warn), and Notification (Note). See Chapter 9 for additional information.

Event Identifier — The event's specific name, containing a prefix denoting the general event category, an abbreviated description of the event, and a message set ID and message ID (a code that refers to a message type that other commands use to produce a more human-readable format).

Event Parameters — Details about the logged event, including information to help trace data through a system, diagnose operating system problems, and so forth.

There are a few guidelines for the format of this field:

- Event parameters are colon delimited.
- Events associated with a network connection will contain the network address of the connection.
- Message events will include a message ID as their first parameter.
- Events associated with a specific client identity will provide the user name when available.
- Process IDs should appear, to establish parent-child relationships when created/reaped.

An event log may also include, at the end, any of the following colon-separated additional information:

`user=%s`	Mail user
`mss=%s`	Mail host
`port=%d`	Port
`mbox=%s`	Mailbox
`from=%s`	From string
`fldr=%s`	Folder
`msgid=%s`	Message ID
`origMsgid=%s`	Original Message ID
`size=%d`	Size
`cmd=%s`	Command
`rme=%s`	RME operation
`fromhost=%s`	From host

`desthost=%s`	Destination host

For example:

```
19980507 133423832-0700 paris popserv 2089 9 16
Note;PopConnTimedOut(66/15) 240:user=jane:mss=lyons:port=5010:mbox=30:
cmd=retr 1:fromhost=127.0.0.1
```

Statistics File Format

The format for entries in a statistics log file is:

```
<rplLogEntry>::= <date> <time> <host> <program> <pid> ";" <event>
```

Figure 19 shows a typical statistics file entry.

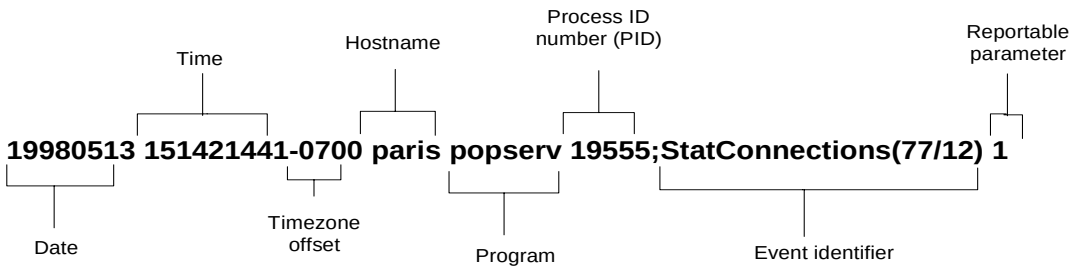

Figure 19 Statistics log file

A statistics file entry consists of the following fields:

Date — The date stamp in YYYYMMDD format.

Time — The time stamp in HHMMSSmmm format, where mmm is in milliseconds, in local time or GMT. If the time stamp is in local time, the timezone offset also appears.

Hostname — The name of the machine where the process resides.

Program — The name of the process that logged the statistic (such as mta or popserv).

Process ID number (PID) — The standard PID of the operating system.

Event Identifier — The reportable parameter's specific name which is an abbreviated description of the reported statistic. This field also contains a general event category number, and a message ID (a code that refers to a message type that other commands use to produce a more readable format).

Reportable Parameters — Reported statistics.

Trace File Format

The format for entries in a trace file is:

```
<traceEntry>::= <date> <time> <host> <program> <pid> <lwp> <thread>
";" <traceFeature> <traceLevel> <traceData>
```

For example:

```
19980511 122236824-0700 paris popserv 2089 5 8;wait=1 entering
waitAll(timeout={5,0}), manager is ef16bb78([-1])
```

Reading Log Files

Log files can contain hundreds or thousands of lines. Although you can view log files in their original format, InterMail also allows you to parse them to produce a more readable format. You can do this using the following two commands:

- `imlogsum` (only for `.log` files)
- `imlogprint(`for `.log`, `.stat`, and `.trace` files)

Using imlogsum

You can use the `imlogsum` administrative command only for reading `.log` files.

The `imlogsum` command produces a summary of output that is easy to read and allows you to see useful system information while ignoring less important information. The output summary includes:

- A list of message types that have occurred in this log
- The total number of each type of message produced
- The number of messages that the MTA handled while the file was logging activity in the system

For example:

```
paris% /vol1/imail/lib/imlogsum -v mta.venus.199708250000.log
File ‘mta.paris.1997199708250000.log’ {
    796: AcctUnknownUser -> The user is not known at this site
1717835: MsgTrace -> Message

1: ProcLaunchReport -> Launched as user
            80472: SmtpConnectionReceived -> An SMTP client  connected
     56: SmtpDNSLookupTimedOut -> A lookup timed out
      6: SmtpQueueProcess -> The deferred queue is being processed
}

************* Totals of messages in ‘.log’ files *************

     796: AcctUnknownUser -> The user is not known at this site
 1717835: MsgTrace -> Message
       1: ProcLaunchReport -> Launched as user
   80353: SmtpConnectionClosed -> An SMTP client closed its connection
to the server after seconds of connect time
   80472: SmtpConnectionReceived -> An SMTP client  connected
      56: SmtpDNSLookupTimedOut -> A lookup timed out
       6: SmtpQueueProcess -> The deferred queue is being processed

************************************************************
               Message Traffic Histogram
************************************************************

Time delivered-paris-int
08/25/97 19:00 4677
08/25/97 20:00 2753
08/25/97 21:00 15513
08/25/97 22:00 25362
08/25/97 23:00 35513
____________________________________________________________
Total: 83818
************************************************************
```

You can also use the `-l <severity>` argument to limit the output to only those messages of a specified severity level.

For more information about the `imlogsum` command, see the *InterMail Mx Reference Guide*.

Using imlogprint

You can use the `imlogprint` command to present `.log`, `.stat`, and `.trace` file information in a readable format. You usually use this after prefiltering the logs to remove unwanted entries. You can include or exclude log entry fields, have output printed on multiple lines, and include field tags to make the meaning of each field clear.

For a log file with time stamps in GMT or a timezone other than the local one, you can use `-l[ocaltime]` to have the time stamps converted to local time.

While `imlogprint` typically accepts a log entry from standard input, you may use also direct an entire log file into `imlogprint` in the following manner:

```
imlogprint -t -m < mta.log
```

The `–t` option places descriptive tags in the beginning of each output field and the `–m` option prints output on multiple lines. If you choose to direct a log file, you will probably want to redirect this output to a second file (such as `mta.log.printout`) in the following manner:

```
imlogprint -t -m < mta.log > mta.log.printout
```

Example

The first of the two examples below shows a standard `imlogprint` command and resulting output. In the second example, the command also includes `–x 1-2` to exclude the first two fields of the log entry (the date and time).

```
mercury% imlogprint -t -m
19971021 021512922 mercury popserv 14812 7 14 Note;PopProtocolErr(66/1) AUTH
twinkie:cmd=AUTH twinkie
date=10/21/1997 time=02:15:12.922 GMT host=mercury prog=popserv pid=14812
lwp=7 thread=14 sev=Notification;[errorCode=PopProtocolErr]
        A POP protocol error occurred during session: POP cmd: "AUTH
        twinkie".cmd=AUTH twinkie
```

```
mercury% imlogprint -t -m -x 1-2
19971021 021512922 mercury popserv 14812 7 14 Note;PopProtocolErr(66/1) AUTH
twinkie:cmd=AUTH twinkie
host=mercury prog=popserv pid=14812 lwp=7 thread=14
sev=Notification;[errorCode=PopProtocolErr]
        A POP protocol error occurred during session: POP cmd: "AUTH
        twinkie". cmd=AUTH twinkie
```

For more information about the `imlogprint` command, see the *InterMail Mx Reference Guide*.

Accessing Log Data Through Named Pipes

Logs are archived records of past InterMail transactions. For circumstances in which you want to see log entries in real-time, you can use a named pipe to access any events reported in log files, and you can view the output in a console window.

You use the following configuration keys to control named pipes:

- `logNamedPipeMode`
- `statNamedPipeMode`
- `traceNamedPipeMode`

If you want to run a console window in the background, set the `logNamedPipeMode` configuration key to `1`. This causes the log output to be written to the pipe (for example, `popserv.pipe.log`) as well as to the file (for example, `popserv.log`).

Named pipes exist in the same directory as log files (the directory specified with the `logDir` configuration key) and have the extension `.pipe.<filetype>` (file type can be `log`, `stat`, or `trace`).

Web Server Logging

The Web server has a separate logging system than the other InterMail servers. The log files generated by the Web server are located in `WEBEDGE_DIR/config_mdn/logs` where `WEBEDGE_DIR` is the top directory in which the Web server is installed.

The Web server generates three log files:

- `access`
- `errors`
- `IMError`

access Log File

This file stores requests made by the connecting clients/browsers. An entry in this log contains the following information:

- The IP address of the client
- The date and time of the request
- The request made by the client on the server

The following is an example log entry in this file:

```
127.0.0.1 - - [08/Jun/1999:11:16:27 -0400] "POST /cgi-bin/gx.cgi/
AppLogic+MobPref? HTTP/1.0" 200 8651
```

errors Log File

This file contains errors, events, and actions taken by the Web server. It also lists errors and exceptions detected during program execution. Informational messages (preceded by `Info:`) generally occupy a single line, whereas error messages (preceded by `Error:`) include the program stack trace, which generally takes up several lines.

For example:

```
Info: Tue Jun 08 09:46:33 EDT 1999 Parsing template:
M:\mobility\vcafe\config_mdn\Templates\ad.htm
```

You can specify the Web server logging level from the Web server administration screen. To select the logging level:

1. From the Administrator menu, select Web Server.
2. In the Server Configuration section, go to the Logging level drop-down list and select the logging level you want:
   - Full displays server operation details that are useful when you are trying to isolate problems.

- Normal logs errors and significant events.
- None disables logging.

3. Click Submit.

   The logging level is selected.

You can archive `errors` log files by specifying the time of day at which you want the log files rotated, as well as the number of log files you want archived.

To set logging time and number information:

1. From the Administrator menu, select Web Server.
2. In the Advanced Configuration section, fill in the following fields:
   - In Rotate Log Hour, enter the hour of the day at which you want the log files rotated, from 0 (midnight) to 23 (11:00 p.m.). The server automatically switches to new log files at the time of day specified (archiving happens, in general, once every 24 hours, unless the Log Rotate Hour gets changed in the middle).
   - In Max Archived Logs, enter the number of archived log files you want kept. The default is 30. The archived logs are present in the same directory as the current log files. The archived files get a number appended to them - the higher the number the older the archive. So, while the current `errors` file is simply called `errors`, the one just archived is `errors.0`, the one before that `errors.1`, and so on.
3. Click Submit.

   The logging time and number information is set.

IMerror Log File

The `IMerror` log file replicates the information contained in the `errors` log file. However, it presents the log information in the InterMail log file format. The entries follow the same logging levels used in the InterMail log files. This helps users who may have developed tools to analyze InterMail logs. An example entry in the `IMerror` log file looks like the following:

```
19990608 162641395-0400 gromit MDN 00000 00 0 Note;Server running at:
10.64.5.42 on port: 82 document root:
M:\mobility\vcafe\config_mdn\docs(0/0)
```


11

Backup and Recovery

Using a comprehensive backup strategy combined with properly tested recovery procedures, you can recover data in the event of catastrophic failure. This chapter covers the basics of backup and recovery for the InterMail system. The specific procedures used at your site will depend on your own provisioning system, the size of the database, available hardware, and available person-hours.

This chapter covers:

- Planning your backup and recovery strategy
- Recommended backup hardware
- Backing up the Configuration database
- Backing up the InterMail Oracle databases
- Restoring the InterMail Oracle databases
- Backing up and restoring the Message File system
- Backing up and restoring mail in process
- Testing backup and recovery procedures

This chapter assumes you have thorough knowledge of your operating system, system configuration, and backup software, as well as a good working knowledge of InterMail and its operations and a general understanding of backup strategy options. You should have considered the requirements for recovery and the expected level of service that your chosen backup strategy will protect. Knowledge of Oracle is helpful but not necessary.

Planning Your Backup and Recovery Strategy

Hardware failure, data corruption, or operator error can severely damage your InterMail system. A comprehensive backup strategy combined with an understanding of recovery techniques allows you to design your system to maintain the required level of service even if an unforeseen disaster occurs.

InterMail backup and recovery procedures address three types of disaster scenarios:

- **Hardware failures**—CPU failure, memory errors, disk errors, disk controller failure, device driver failure, or total system failure
- **Software failures**—Errors that occur in the operating system or related third-party software applications
- **Operator errors**—Errors that result from unexpected user behavior, such as the erroneous deletion of files or logs or accidental shutdown of the system

Backup efforts in InterMail should focus primarily on:

- Account information contained in the Directory database
- Configuration information contained in the Configuration database
- Persistent mail storage in the Message File system and the Message Store database
- Temporary mail storage in the Queue directory and the local `spool` directories of the Message Transport Agent (MTA)

You should perform regular system or environment backups in addition to the procedures discussed here. Environment backups should include regular incremental backups of non-InterMail files and occasional backups of InterMail binaries and configuration files.

Note: You cannot consider any backup and recovery strategy effective until you fully test it under field conditions. You should conduct regular tests in a lab environment to ensure that your backup and recovery strategies work.

Recommended Backup Hardware

The first step in developing a backup and recovery plan is to decide upon the hardware to be used. This section provides general recommendations with respect to disk arrays, tape devices, and high-availability solutions.

Note: For MTA hosts, regular online backups are not possible due to the transient nature of the data. The most reliable strategy for backing up MTAs is full hardware mirroring including three-way mirrors with redundant controllers.

Disk Arrays

Disk arrays are fast and capture images of data quickly, without much overhead on the production system. However, disk arrays are expensive and have limited storage capacity. If cost is an issue, a spare disk array may be less desirable than tape. If performance is the prevailing issue, the disk array option is clearly the best choice. It

is not necessary that spare disk arrays be redundant arrays of inexpensive disks (RAID) or mirrored.

Tape Devices

Tape devices take much longer than disk arrays to capture images, and they tend to have higher error rates than disks. However, tape devices are inexpensive and have unlimited storage potential. Multiple backups of any given file reduce the risk of file-write errors.

For long-term storage, saving images to tape is the only practical solution. Digital linear tape (DLT) is much faster and has higher capacity than digital audio tape (DAT), although it is also more expensive, especially compared with DLT jukeboxes.

Ideally, the first step would be to copy backed up files to a separate array of fast disks in order to minimize impact on production systems. The next step would be to copy the files to tape or another backup medium.

High-Availability Solutions

Another strategy that may increase system reliability is the use of a high-availability solution. Systems usually implement high availability as a software failover mechanism that runs as a daemon process. High availability requires at least one redundant machine with a configuration and shared disks, identical to those of the production machine it will recover. During failover, the redundant machine essentially assumes the identity of the failed production machine. Software.com does not provide a high-availability solution, but has certified several third-party tools for use with InterMail systems.

Backing Up the Configuration Database

The Configuration server is on the master configuration host. This server maintains the Configuration database and is responsible for distributing the contents of that database to all other InterMail hosts. It is extremely important that you back up the information in the Configuration database.

You should back up the Configuration database:

- After a major installation
- Each time the configuration information changes

You can set a path with the `extraCopyConfigDBPath` configuration key in order to have the Configuration server write an extra copy of the Configuration database file.

Should it ever be necessary to revert to an earlier version of the Configuration database, use the `versionConfigDB` configuration key.

You can also use the configuration key `versionConfigDB` to make a backup copy of the current Configuration database. If `versionConfigDB` is set to `true`, each time the Configuration server writes a new Configuration database, it renames the old one

to `config.db.<YYYYMMDDHHMMSS>`. This makes it easy to revert to an earlier Configuration database should that be necessary. You should periodically examine the size of the directory containing the backed up Configuration database files and delete files that are no longer needed.

Backing Up the InterMail Oracle Databases

InterMail contains two Oracle databases: the Directory database and the Message Store database. The backup and recovery procedures are the same for both databases.

The Directory database is part of the (Integrated Services Directory (ISD), which also contains several local Directory Cache (non-Oracle) databases and their associated servers. Given that the ISD is the sole source of directory information for the InterMail system, it is essential that the Directory database be backed up regularly. The local caches can be reproduced from the main Directory database and, therefore, do not have to be backed up separately.

The frequency of backups depends on how quickly the data in the Directory database changes. If your site creates or modifies a large number of accounts, you should back up this data often. We recommend that you make a hot backup of the Directory database nightly. For complete inforamtion on hot backups, see “Making a Hot Backup” on page 205.

The Message Store database maintains tables of pointers to message files and their statuses (such as Read or Deleted) in the Message File system. The actual message files containing the message content are stored in the Message File system.

For every Message Store server in your InterMail system, there is a Message File system and an associated Message Store database. If the system loses one, the contents of the other are meaningless. You should back them both up.

Figure 20 illustrates the structure and content of the Message File system and Message Store database.

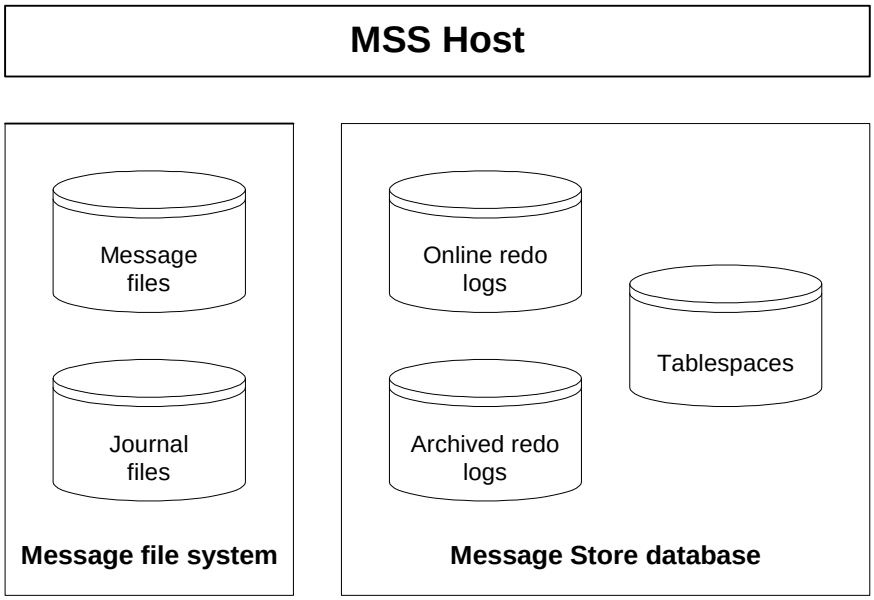

Figure 20 Message File system and Message Store database

Oracle Database Files to Be Backed Up

The following files must be included in MSS backups because they are stored both in an Oracle database and in the Message File system:

- **Message files**—InterMail message files are stored in the `Msgfiles` directory.
- **Journal files** —InterMail creates a transaction record for every message file created and deleted and stores these records in journal files.

In addition, for successful recovery of an Oracle directory database or Message Store database, it is critical that the following files be backed up:

- **Online redo logs**—These files record the changes made by the most recent transactions in the Oracle database. They are similar to InterMail journal files. One online redo log is the current log in which Oracle records all new changes. If the system loses the online redo logs, in particular the current log, it also loses the most recent set of changes to the database.

  When the current log becomes full, Oracle switches to a different redo log and begins recording changes there, and it copies the previous log to the `archive_dest` directory as an archived redo log.

  Oracle's software mirroring feature protects the online redo logs. The software mirroring of Oracle redo logs is set up automatically by the Oracle installation utility.

Note: It is impossible to perform a hot backup on online redo logs. Online redo logs must be mirrored in a robust fashion so that you can be confident that at least one mirror will survive. The redo logs can be backed up "cold," meaning that the database is down when the files are copied. This is, however, of limited value, in that the data in the redo logs typically changes reapidly. The only copy of the most recent changes is in the redo logs only. Therefore, if the redo logs are lost, some of the most recent changes to the database will also be lost.

- **Archived redo logs**—These are copies of online redo logs that have "filled up" with transactions and been archived by the system. Make backup copies of these files.
- **Tablespace data files**—These include the system tablespace, temporary tablespace, data table tablespace, rollback segment tablespace, and, optionally, index tablespace. Make backup copies of all of these files.

Note: Tables in the Message Store database and the Message File system can be in one of two unsynchronized states. In the "widows" state, the database tables contain references to missing message files; in the "orphans" state, there are message files which have no corresponding pointers in the database tables. If one of these unsynchronized states exists when file recovery is needed, InterMail uses the information contained in the Message Store database as the authoritative information.

Types of Backup Used with Oracle Databases

There are three types of backup used for Oracle databases:

- Static file backups
- Complete image ("cold") backups
- Hot backups

Static File Backups

In the Directory database, you should back up the entire `$ORACLE_HOME` directory. This directory includes Oracle files needed to support the operation of the Directory database (e.g., Oracle configuration files an administrator would make minor changes to from time to time). These files are only rarely written.

Note: Dynamic database files are not usually kept in `$ORACLE_HOME`, where they would complicate Oracle upgrades.

Static files, such as the host system software, Oracle software, and configuration files require occasional backups since they change rarely. For example, the Oracle binary files change only when the software is upgraded, and the `.cshrc` and `.profile` files change only when manually edited.

Caution! This type of occasional backup should include only static files described above, not the highly dynamic files that contain the database data. Do not include any of the database data, online redo logs, or control files in this type of backup. These files typically have a `.dbf` or `.ctl` extension. Inadvertently restoring one of these files from this type of occasional backup could be disastrous, since the backup data would be obsolete.

Complete Image Backups

A complete image backup of both InterMail-specific files and operating system files should be made immediately following a new or upgraded installation. Although this type of backup is done only infrequently, it is important for recovery since it provides a "snapshot" of the entire system at one point in time. A complete image backup should take place whenever the system is in a quiescent state; the more recent a complete backup is, the more useful it will be during recovery.

Note: Whenever installing any new software, you should back it up immediately as part of your general backup strategy. If you have a customized system, remember to back up your customized directories as well.

Immediately after an InterMail installation, no services have started. If you have started them manually or if this is an existing InterMail system, you must stop them on all the MTA, MSS, and ISD machines before making the backup.

You should copy all directories and files created as a part of InterMail and any other files that have changed as a result of the installation, as well as the system configuration files and user files.

Following this process, you should have a complete image of the working system and the operating environment. At this point, the InterMail system can begin running. If this is a new installation, the system should also be ready for provisioning new accounts.

Hot Backups

Typically, a `cron` job performs a hot backup of the Message Store database and the Oracle directory database nightly, while the service is up and running. Another `cron` job periodically wakes up to make copies of archived redo log files.

Making a Hot Backup

Two InterMail utilities, `imdbhotbackup` and `imoracopyarchredo`, run as `cron` jobs to make hot backups of:

- Critical data files
- Archived redo logs

These files are crucial to recovery of the Message Store or Directory database. If any one of them is missing, the system can lose the results of committed transactions, resulting in lost mail or vanishing user accounts.

Critical Data Files

The files that contain the database tables, the rollback segments, and the system tablespace are considered critical data files. The database tables specify which messages are in which mailboxes; the rollback segments are vital to complete recovery; and the system tablespace contains a "map" of the database that describes the schema and organization of tables, blocks, and files. All these are critical pieces of information.

To make backup copies of critical data file, you should run the administration command `imdbhotbackup` as a `cron` job. The complete syntax for this script is shown in the *InterMail Mx Reference Guide*.

By default, the `imdbhotbackup` utility handles backups only to disk. To back up a file to tape, you must write a customized command and pass it to `imdbhotbackup` with the `–backupcommand` parameter.

Backing Up to Disk

In the following example, the system dedicates a disk array, `/volB`, to storing backups. The array is instructed to keep two backup copies: the latest backup and the previous one. In the example, the `crontab` entry belongs to the `oracle` user, so that will be no permission problems with access to data files.

```
30 3 * * * ORACLE_HOME=/volX/oracle/7.3.2 ORACLE_SID=IMM1 LD_LIBRARY_PATH=/
volX/imail/lib:/volX/oracle/7.3.2/lib /volX/imail/bin/imdbhotbackup
–cronjob –sleeptime 120 –othertemporarytablespaces TEMP –backupdestdir /volB
–totalbackupskept 2 –logdir /volB/BACKUP_LOGS –dbname IMM1 –id imail/imail
```

Backing Up to Tape

Following is a sample `crontab` entry for the `oracle` UNIX user for backup to tape. The `crontab` entry belongs to Oracle so there will be no permission problems with access to data files. The `oracle` user is not a member of the InterMail group and will not be able to access the Configuration database. Therefore, the `–logdir` parameter prevents the command from trying to look up `logDir` in the Configuration database and failing.

```
30 3 * * * ORACLE_HOME=/volX/oracle/7.3.2 ORACLE_SID=IMMX LD_LIBRARY_PATH=/
volX/imail/lib:/volX/oracle/7.3.2/lib /volX/oracle/imdbhotbackup
–cronjob –sleeptime 120 –othertemporarytablespaces TEMP –backupdestdir /
volY/backup –totalbackupskept 1 –logdir /volY/backup/LOGS –backupcommand
/usr/local/bin/backuptotape –dbname IMMX –id imail/imail
```

When you back up to tape, you must specify the `–backupdestdir` argument (as is shown in the example above). The destination directory stores the following:

- Scripts for re-creating indexes and temporary tablespace
- A backup history file maintained by `imdbhotbackup`
- A backup copy of the database control file

The following is an example of a customized `backuptotape` command:

```
#! /bin/sh
if echo "$1" | cpio –ocBv > /dev/tapexas
   then
     exit 0
   else
     exit 1
fi
```

Handling a Crash During a Hot Backup

If a crash occurs while `imdbhotbackup` is running, it is likely that, when you reboot the machine, the Oracle background processes will launch successfully but that the database will not come back up cleanly. Issuing a `ps` command will reveal that the database processes are running, but the database will not accept any commands.

The problem is that a crash during a hot backup of a data file leaves the file header in an inconsistent state. The sytem detects this at startup, issues an error saying "ORA-01113: file X needs media recovery", and refuses to open the database fully.

To get the database fully operational:

1. Set the `ORACLE_HOME` and `ORACLE_SID` environment variables properly.
2. Log in as the `oracle` user.
3. Run the Oracle `svrmgrl` utility as follows:

```
% echo $ORACLE_HOME $ORACLE_SID
% $ORACLE_HOME/bin/svrmgrl
>connect internal
>startup
ORA-01113: file N needs media recovery
ORA-01110: datafile N: '/foo/bar/snafu.dbf'
>recover datafile '/foo/bar/snafu.dbf'
>alter database open;
```

4. If the system issues another "ORA-01113" error, repeat the `recover datafile` and `alter database` commands.
5. Issue an `exit` command to terminate `svrmgrl`.

Archived Redo Logs

Archived redo logs are critical to recovery if the system loses a critical data file. If one of the archived redo logs is missing, the backup copy can be re-created only to the time at which the missing log started recording changes.

You should back up archived redo logs shortly after the system creates them. This ensures that two copies always exist: one in Oracle's `archive_dest` directory and

one in the backup directory, or one in Oracle's `archive_dest` and one among the online redo logs.

Losing one archived redo log makes full recovery impossible. The `imdbcopyarchredo` utility addresses this problem by copying archived redo logs soon after creation so that two copies exist on independent devices.

The `imdbcopyarchredo` utility also regularly prunes old archived redo logs from the `archive_dest` directory, preventing them from filling up the `archive_dest` directory.

You should run the `imdbcopyarchredo` utility as a `cron` job at intervals that are a small multiple of the average time between log switches. This ensures that the utility copies an archived redo log before the system begins to overwrite its online redo log original.

Restoring the InterMail Oracle Databases

This section explains how to recover components of the InterMail Oracle databases, including:

- Oracle home directory
- System tablespace
- Temporary tablespace
- Data table tablespace
- Rollback segment tablespace
- Index tablespace
- Online and archived redo logs
- `archive_dest` directory
- Files from a combination of categories

Your backup strategy should ensure that you can fully recover the database if all the files on any one device are lost. Whatever the specifics of your strategy are, you should do hot backups nightly and have a day's worth of archived redo logs available in `archive_dest`.

Oracle Home Directory

If the system loses the `oracle` home directory, including all the Oracle binaries, libraries, and database parameter files, try copying Oracle binaries from the `$ORACLE_HOME` directory on another machine.

You should re-create the `oracle` home directory, including `.cshrc` and `.profile` files. `TNS_ADMIN` is typically set to `$ORACLE_HOME/network/admin`; in general, this should be set to the path of the directory that contains the SQL*NET configuration files `listener.ora`, `tbsnames.ora`, and `sqlnet.ora`.

ORACLE_SID must be set to the name of the database. ORACLE_HOME must be set to the path of the root of the Oracle system file hierarchy. LD_LIBRARY_PATH must include $ORACLE_HOME/lib. In the following instructions, assume that HOME is the oracle user's home directory path.

To recover the Oracle home directory:

1. Copy $ORACLE_HOME from another machine
2. Make the directories $HOME/admin/$ORACLE_SID/dump_dest/bdump, …/cdump, and …/udump.
3. Make the file $HOME/admin/$ORACLE_SID/pfile.
4. Copy the hot backup file BAK.init$ORACLE_SID.ora to $HOME/admin/$ORACLE_SID/pfile/init$ORACLE_SID.ora.
5. Create a symbolic link:

   ```
   ln –s $HOME/admin/$ORACLE_SID/pfile/init$ORACLE_SID.ora
   $ORACLE_HOME/dbs/init$ORACLE_SID.ora
   ```

6. Copy the hot backup file BAK.configIMDB.ora to $HOME/admin/$ORACLE_SID/pfile/configIMDB.ora.
7. Copy the hot backup files BAK.tnsnames.ora and BAK.listener.ora into $ORACLE_HOME/network/admin/tnsnames.ora and …/listener.ora.

System Tablespace

The system tablespace file, SystemTablespace.dbf, records the schema and structure of the database, and Oracle cannot start the database without it.

You can use the following procedure to recover the Oracle system tablespace file only if all of the following conditions are met:

- You are certain that you are restoring only the data files affected.
- You are certain that you are operating on the correct tablespace.
- All of the online redo log files are intact. If this is not the case, a full database recovery must be done, and the assistance of an experienced database administrator is recommended.
- All of the archives redo log files from the time of just before the last hot backup of SystemTablespace.dbf up to the present, without any gaps, are in the directory $ORACLE_HOME/dbs/init$SID.ora file parameter log_archive_dest. If your archived redo log files are not all in this directory, you must use the set logsource <path> command to specify the path of the directory where they can be found.

To recover the system tablespace file:

1. Copy BAK.SystemTablespace.dbf from your last hot backup to SystemTablespace.dbf in the proper directory.

2. Run Oracle Server Manager and start the Oracle instance in Restricted Mount mode:

```
SVRMGR>  connect internal
SVRMGR>  startup restrict mount
```

Note: If you are restoring `SystemTablespace.dbf` to a different path or file system, issue the following command at this point:

```
alter database rename file '/<oldpath>/
SystemTablespace.dbf' to '/<newpath>/
SystemTablespace.dbf'
```

---

3. Roll the restored file forward using archived redo log files:

```
SVRMGR>  recover database
```

Oracle examines the `SystemTablespace.dbf` file and automatically determines that it needs to be rolled forward. It automatically determines the name of the first archived redo log file it needs. Oracle then "guesses" at the complete path of this file, and asks for your approval. Press the Return key to approve Oracle's choice. Oracle then rolls the `SystemTablespace.dbf` file forward using the archived redo log. It then determines the name of the second archived redo log it needs, "guesses" at a path, and prompts you to approve. This process continues until the `SystemTablespace.dbf` file is brought completely up to date. Do not enter `Cancel` in response to Oracle's prompt. Telling Oracle to cancel recovery at this point could cause serious consequences.

```
SVRMGR>  alter database open
```

4. Start the affected MSS Oracle instance on the recovered MSS machine, and start InterMail on the MSS.

Temporary Tablespace

By default, `imdbhotbackup` does not back up temporary tablespaces. However, it does include scripts with the nightly hot backup that can re-create each one quickly and easily. For every temporary tablespace that is not backed up, the scripts `FilesOfflineTEMPTBS.sql` and `RecreateTempTbsTEMPTBS.sql` are included in the hot backup.

To recover temporary tablespace:

1. Create all necessary directories, including the directories in the temporary tablespace file paths that appear in the scripts, if they do not exist.
2. Run Oracle Server Manager and start the Oracle instance:


```
Svrmgrl
>connect internal
>startup mount
>@/archive/store1/HOT_BACKUP_COPIES_IMD1/CURRENT/FilesOfflineTEMPTBS.sql
>recover automatic database
>alter database open
>@/archive/store1/HOT_BACKUP_COPIES_IMD1/CURRENT/
RecreateTempTbsTEMPTBS.sql
```

Data Table Tablespace

To recover a data table tablespace file, make a new online copy from a hot backup copy of the corrupted file.

Note: The procedure for restoring a lost data table tablespace file is much the same as for restoring a lost system tablespace file, as shown above in "System Tablespace" on page 209.

To recover data table tablespace:

1. Copy `BAK.SystemTablespace.dbf` from your last hot backup to `SystemTablespace.dbf` in the proper directory.
2. Run Oracle Server Manager and start the Oracle instance:

   ```
   SVRMGR> connect internal
   SVRMGR> startup mount
   ```

3. Issue the following commands:

   ```
   SVRMGR> recover database
   SVRMGR> alter database open
   ```

Rollback Segment Tablespace

Rollback segments are vital to Oracle database recovery. If the rollback segment tablespace file, `RollbackTablespace.dbf`, is corrupted, copy a hot backup copy to where the online copy should be.

Note: The procedure for restoring a lost rollback segment tablespace file is much the same as for restoring a lost system tablespace file, as shown above in "System Tablespace" on page 209.

To recover rollback segment tablespace:

1. Copy `BAK.SystemTablespace.dbf` from your last hot backup to `SystemTablespace.dbf` in the proper directory.
2. Run Oracle Server Manager and start the Oracle instance:

   ```
   SVRMGR> connect internal
   SVRMGR> startup mount
   ```

3. Issue the following commands:

```
SVRMGR> recover database
SVRMGR> alter database open
```

Index Tablespace

By default, the hot backup does not include the index tablespace. However, it does include scripts for re-creating all indexes from scratch; these scripts are named `IndexFilesOffline.sql` and `RecreateIndexTablespaces.sql`.

To recover index tablespace:

1. Make certain that `immssgc`, the garbage collector, is not running and cannot run until you restore all the indexes.
2. Comment out the `imail crontab` entry for `immssgc`.
3. Enter `/bin/ps –ef | grep immssgc` to make sure it is not running.
4. Run Oracle Server Manager:

```
Svrmgrl
> connect internal
> startup mount
> @IndexFilesOffline.sql
> alter database open
> @RecreateIndexTablespaces.sql
```

Online and Archived Redo Logs

You should have hardware or software to mirror the online redo logs in order to reduce the likelihood of losing both copies. You cannot recover the database fully if these logs are lost.

If the system loses one mirror, you can restore it by copying the other mirror. By default, redo log files have names such as `RedoGrp<N>Mem<X>.dbf`. Files that have equal values of the variable `<N>` in their redo log names are mirrors of one another. The variable `<X>` is the mirror number, ranging from 1 to the number of mirrors.

If the system loses all of the archived redo logs, perform a hot backup as soon as possible.

Combination of Files

If the system loses files belonging to more than one category, combine the instructions from each section as follows:

1. Do all the steps from each section up to the point of running `svrmgrl`.
2. Run the Server Manager.
3. Do all the subsequent steps from each section up to the point of issuing the `recover automatic database` statement.

4. Issue the `recover automatic database` command, if it applies.
5. Do all the subsequent remaining steps from each section.

Backing Up and Restoring the Message File System

The Message File system contains the content of all messages for all InterMail users. Since the Message File system is very large, making a full system backup nightly is inconvenient. Instead, you should make incremental backups, which add the new messages in the Message File system to the backup copy of the Message File system. The incremental backup then continually updates the full backup.

To keep the backup copy from becoming too large, you must remove those messages from the backup copy that the system has deleted from the actual Message File system. To do this:

1. Compare the garbage collector journal files against the backup. Garbage collector journal files contain the paths of deleted message files.
2. Delete these message files from the backup copy.

Backing Up the Message File System

To back up the Message File system, you should establish two utilities as nightly `cron` jobs:

- Run the `imdbmsgbackup` utility to make incremental backups of the Message File system.
- Run the `imdbplaygcjrn` utility to play old garbage collector journal files against the backup and remove unnecessary messages.

Making Incremental Backups of the Message File System

The administration command `imdbmsgbackup` queries the Message Store database to determine the list of message files to back up. This is more efficient than entering a `stat` command on a million message files to determine which ones are new.

The complete syntax for this script is shown in the *InterMail Mx Reference Guide*.

The `imdbmsgbackup` utility does not back up message files for messages not in a message store. Therefore, it does not matter whether you run this utility before or after the garbage collector runs. However, you should avoid running `imdbmsgbackup` concurrently with the garbage collector. If you did, there could be contention for the Message File system, and the garbage collector could delete a message that `imdbmsgbackup` is about to back up, resulting in a warning message that a message file is missing.

You should run the `imdbmsgbackup` command nightly.

Removing Unnecessary Messages from the Backup

The administration command `imdbplaygcjrn` plays garbage collector journals that it has not played before, but that are at least 24 hours old, against a backup copy of a Message File system.

The `imdbplaygcjrn` command should run nightly as a `cron` job. It should play garbage collector journals from the InterMail journal directory, run `tar` on the journals it has played, and store the resulting compressed `tar` file.

The complete syntax for this script is shown in the *InterMail Mx Reference Guide*.

Restoring the Message File System

To recover the Message File system, you need:

- Journals that MSS processes maintained
- Hot backups of the Message File system

The journals record all transactions (writes and deletes) to the Message File system. Journaling provides protection for messages received between regularly scheduled backups. Together, journals and backups allow you to recover your system to the exact point it was at before disaster struck. It is therefore important to keep journal files on a disk system that is separate from the Message File system and to make backups of these files at least hourly to protect against file loss

Between hot backups, the system writes journal entries and, depending on the number of messages moving through the system, creates one or more new journal files. Until the next hot backup, journal files are the last record of transactions to the MSS.

Activating the Stand-In

Since restoring the Message File system will take some time, you may want to bring up an initially empty stand-in Message File system so that customers can read new mail while the recovery is in progress. If the `createsMboxes` configuration key is set to `true`, the system will automatically create mailboxes in the stand-in as messages arrive or users attempt to retrieve mail.

Deactivating the Stand-In

After completing the recovery, you can choose either to have the restored system take over and deactivate the stand-in or to leave the stand-in active and keep the restored system offline. In either case, however, you must move all the mail from the inactive system to the active one.

The Message Store database on the inactive system knows which messages are in which user’s mailbox. This data, together with the message body in the Message File system, can help redeliver each message to the proper mailboxes on the active MSS.

Recovery Procedure

The basic strategy for recovering the Message File system is simple: copy the backup MSS and play it against the jounal files. Before you start the recovery, determine which journal files to play against the backup by opening the file `message_file_system_backup/message_backup_state.txt` and looking at the line `LAST_BACKUP_SNAPSHOT_TIME=<YYYYMMDD.HHMMSS>` which tells you the time at which the last incremental backup began. You will need to play all journal files starting with those created or modified just before this time.

Note: Keep in mind the incremental backup time, and make sure you play enough journal files.

To recover the Message File system:

1. Copy the archived Message File system to `$MessageFileDir`.

   You should copy each subdirectory separately by entering `cp -r`. That way, if you encounter a problem while making a copy, you will not have to start over.

2. After copying the Message File system completely, use the `imjrnrecover` utility to bring it up to date.

There are three types of journal files:

- MSS journals
- Queue journals (discussed in “Backing Up and Restoring the Queue Directory” on page 217)
- Garbage collector (GC) journals

After the system is backed up, you can play GC journals at your convenience. The only drawback to delaying the replay of GC journals is that the recovered Message File system may initially contain garbage messages that waste space.

Backing Up and Restoring Mail in Process

The Queue server manages mail in transit and stores it temporarily in the Queue directory on the host on which the Queue server resides. If the system must store mail temporarily and no Queue server is available, the MTA stores the mail in its local `spool` directory.

Figure 21 shows the directory structure used to organize mail in process. The structure of the directory is identical for the Queue server and the MTA; however, the root for the entire structure is called `queue` on the Queue server and `spool` on the MTA.

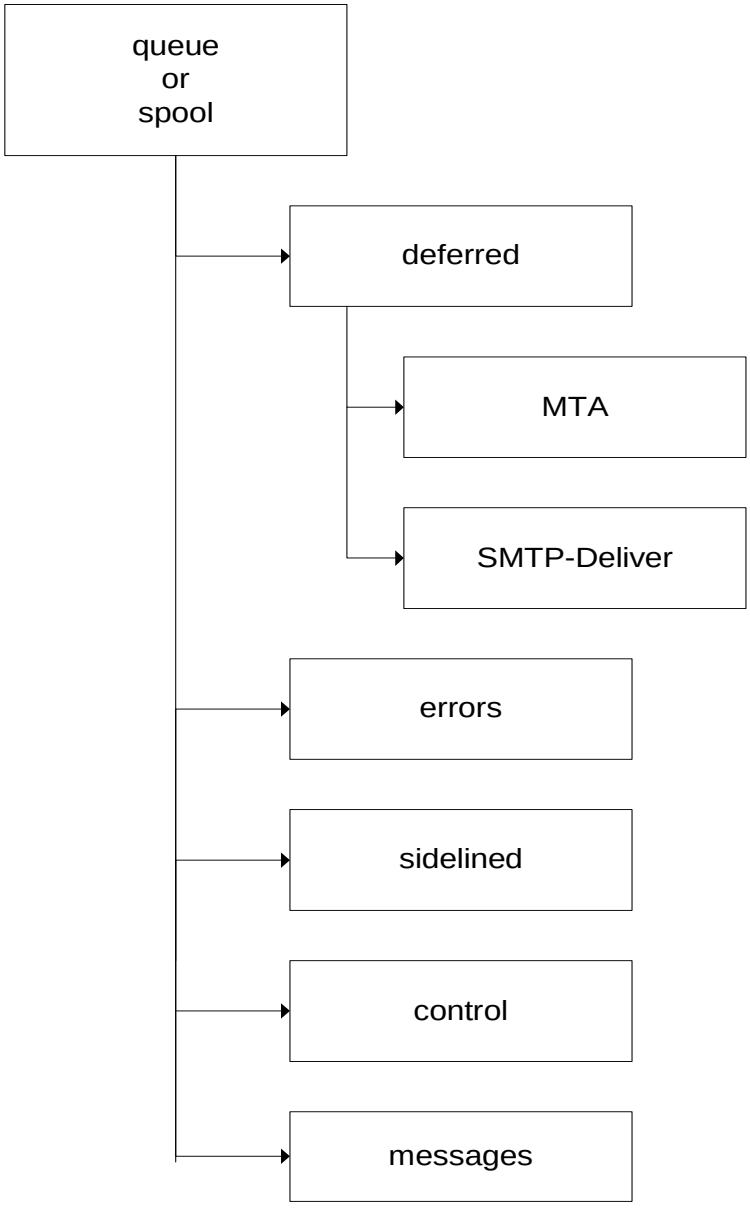

Figure 21 Structure of the queue or spool directory

Backing Up and Restoring the Queue Directory

The Queue directory contains all mail that is deferred, in error, or sidelined. Therefore, it is critical that you back up this file system.

Because the system journals all queuing activity, it is possible to recover all messages in the Queue directory up to the moment at which the directory was lost.

Backup of the Queue directory is a two-step operation:

1. Backing up the contents of the Queue directory.
2. Backing up the queue journal files and garbage collector files.

To recover the Queue directory:

1. Restore the last backup of the Queue directory.

   Note: If you have lost both the Message File system and the Queue directory, you should also restore the last backup of the Message File system.

2. Use the `imjrnrecover` utility to replay the journal files from the time of the last backup. To recover queue data only, limit the replay to the journal files with the `.queue` extension.

Backing Up and Restoring the Spool Directory

The one important difference between the `spool` directory and the Queue directory is that the system does not journal spooling activity on the MTA, making data loss of data possible. To reduce the chances of such loss, the MTA server should have disk or file system redundancy, and you should back up the `spool` directory periodically.

To recover the `spool` directory, restore the last backup.

Testing Backup and Recovery Procedures

In order to ensure that backup and recovery procedures will be effective for your InterMail system, it is important to test all procedures and hardware before using them in a production environment. It is also important to establish a regular schedule of backups and system monitoring to ensure that the InterMail system can run smoothly.

The results of the test procedures described in this section should help you tune your backup and recovery strategy to obtain optimum system performance and reliability.

The test procedures described in this section model the following potential failures and subsequent recovery strategies:

- Total system loss due to total hardware failure, disk array loss, operator error, or environmental conditions, not including database control files and redo logs
- Total system loss, including loss of Oracle data files, control files, and online redo log files, with recovery from nightly backup files only

- Loss of Oracle tablespace and data files
- Loss of Oracle control files
- Loss of individual messages

Preparing to Run Test Procedures

In preparation for testing the procedures, it is recommended that you do the following:

1. **Establish the test platform**—Before bringing your system online, you should construct a trial run and perform a backup and mock recovery using the production configuration.
2. **Document the production environment**—This includes a detailed description of installation options and backup strategy.
3. **Install InterMail**—Create a fresh installation for testing.
4. **Perform an initial backup**—Create a backup of the configuration of your test environment.
5. **Build simulated mailboxes and sample messages**—Add users, mailboxes, sample messages, and other data to simulate a working system.
6. **Back up the simulated work environment**—Take an incremental backup of a working test system.

Total System Loss and Recovery

This procedure simulates a catastrophic disaster by shutting down InterMail and removing the entire system. It assumes that you still have the database control files and redo logs. If this is not the case, see “Total System Loss and Recovery” on page 218.

Test Procedure

1. Create test accounts and send test messages.
2. Shut down InterMail and Oracle:

```
% lib/imservctrl stop
% lsnrctl stop
% shutdownDB IMM1 // or MSS Oracle SID
% shutdownDB IMD1  // or DIR Oracle SID
```

3. Delete the file systems containing the following:
   - InterMail
   - InterMail journals
   - Message text
   - Oracle
   - Oracle tablespace

- Oracle redo archive logs
- Oracle control files

Optionally, you can delete the operating system, machine configuration, and tuning files, and then restore the operating system to the required patch level and reconfigure the system files and tuning parameters required for InterMail.

4. Restore the complete backup and the last available hot backup.
5. Copy the backup control file to all Oracle control file paths. You can find the control file paths in the `$ORACLE_SID_home_directory/pfile/configIMDB.ora` file under the key `CONTROL_FILES`. The backup control file must be copied to the same path/file name as each item in the list before the Oracle instance will start.
6. Start up the database and recover using the redo log files.
7. Replay the MSS journals to restore the latest state of the Message File system:

   ```
   imjrnrecover –a
   ```

8. Restart the InterMail and Oracle services:

   ```
   % startupDB IMM1 // or MSS Oracle SID
   % startupDB IMD1  // or ISD Oracle SID
   % lsnrctl start
   % lib/imservctrl start
   ```

9. From the POP server, attempt to retrieve the test messages created in Step 1.
10. Create new test mail accounts and messages to test the restored services.

Total System Loss and Recovery from Nightly Backup Files Only

If all of the online database data files, control files, and online redo log files have been lost, and only copies of the nightly hot backup files are available, there is no way to fully recover the database. This is because the latest changes to the database were recorded only in the lost online redo logs.

Note: To prevent the loss of online redo log files, make sure your redo log mirroring is very robust.

Test Procedure

1. Log in as the `oracle` user so that files are created with the proper ownership. Make sure the `ORACLE_HOME`, `ORACLE_SID`, and `TNS_ADMIN` environment variables are set properly. `ORACLE_HOME` must be set to the path of the root of the Oracle system file hierarchy. `ORACLE_SID` must be set to the name of the database. `TNS_ADMIN` is typically set to `$ORACLE_HOME/network/admin`. In general, it should be set to the path of the directory that contains the SQL*NET configuration files `listener.ora` and `sqlnet.ora`.

> ***Note:*** If any directories are missing in the paths shown in the subsequent steps, create them.

2. If the files under `$ORACLE_HOME` were lost, restore `$ORACLE_HOME` from a backup, or copy `$ORACLE_HOME` from another machine.
3. Copy the `init.ora` database parameter file `BAK.init$ORACLE_SID.ora`, made by `imdbhotbackup` under the backup destination directory, to `$ORACLE_HOME/dbs/init$ORACLE_SID.ora`.
4. Examine the `init$ORACLE_SID.ora` file for any `ifile` parameters and restore to the proper path a backup copy of each file listed in the `ifile` parameter. The `imdbhotbackup` utility creates backup copies, in the backup destination directory, of all files with an `ifile` parameter.
5. Look for the parameters `background_dump_dest`, `core_dump_dest`, and `user_dump_dest` in the `init$ORACLE_SID.ora file`, the values of which will be directory names. If these directories were lost, restore them from the most recent backup available. If no backup is available, create them. Their being empty is not a problem; simply make sure they are readable and writeable by `oracle`.
6. Look for the `log_archive_dest` parameter in the `init$ORACLE_SID.ora` file. If the directory listed here does not exist, create it, making sure it is readable and writeable by `oracle`.
7. Copy the system tablespace hot backup file `BAK.SystemTablespace.dbf` to `SystemTablespace.dbf` in the proper directory. If you don't know the proper directory path, you can extract that information from the `CREATE_CONTROLFILE.$ORACLE_SID.sql` file created under the backup destination directory. Make sure that the restored file is readable and writeable by `oracle`.

```
cd /vol4a/oracle/admin/IMM1
cp /archive/store1/HOT_BACKUP_COPIES_IMM1/
COPY.BAK.SystemTablespace.dbf SystemTablespace.dbf
chmod u+rw SystemTablespace.dbf
```

8. Delete any remaining `RedoGrp*Mem*.dbf` files. Oracle re-creates the online redo log files from scratch.
9. Examine the `init$ORACLE_SID.ora` file again, this time for the `control_files` parameter, the value of which will list the full paths of mirrors of the control file. Choose one of the backup control files which will have the `.ctl` extension, and copy it to each of the file names. Backup control files have the extension `.ctl`. Make sure that the restored control file mirrors are readable and writeable by `oracle`.
10. If SQL*NET or Net8 is installed, copy the `tnsnames.ora` parameter file, `BAK.tnsnames.ora`, to `$TNS_ADMIN/tnsnames.ora.`

11. If SQL*NET or Net8 is installed, copy the Listener parameter file, `BAK.listener.ora`, to `$TNS_ADMIN/listener.ora`. Examine the restored `listener.ora` file for a `log_directory_listener` parameter, and check that its value is an existing directory readable and writeable by `oracle`. If the directory does not exist, create it.

12. Copy a hot backup copy of each of the `POM*.dbf` files to the proper path. Your database will have a number of these files. To get a list of them, look in the file `CREATE_CONTROLFILE.$ORACLE_SID.sql` created in the backup destination directory. Make sure the restored files are readable and writeable by `oracle`.

```
cd /vol4b/pom
cp /archive/store1/HOT_BACKUP_COPIES_IMM1/COPY.BAK.POM01.dbf POM01.dbf
cp /archive/store1/HOT_BACKUP_COPIES_IMM1/COPY.BAK.POM02_1.dbf POM02_1.dbf
cp /archive/store1/HOT_BACKUP_COPIES_IMM1/COPY.BAK.POM03_1.dbf POM03_1.dbf
cp /archive/store1/HOT_BACKUP_COPIES_IMM1/COPY.BAK.POM04_1.dbf POM04_1.dbf
cp /archive/store1/HOT_BACKUP_COPIES_IMM1/COPY.BAK.POM05_1.dbf POM05_1.dbf
chmod u+rw POM*.dbf
```

13. Restore the rollback segment tablespace datafile, typically named `RollbackTablespace.dbf`, from a backup copy:

```
cd /vol4b/rollback
cp /archive/store1/HOT_BACKUP_COPIES_IMM1/CURRENT/COPY.BAK.rback01.dbf
rback01.dbf
chmod u+rw rback01.dbf
```

14. Remove all `POX*.dbf` files,which provide storage for indexes. Oracle will re-create all indexes from scratch.

15. Make certain that `immssgc`, the garbage collector, is not running and cannot run until all the indexes are restored. Comment out the `imail crontab` entry for `immssgc` by putting a pound character (#) in front of the line that contains `immssgc.` Then enter `/bin/ps –ef | grep immssgc` to make sure the garbage collector is not running. If you see a line containing `immssgc`, remove it.

Caution! If `immssgc` runs during the ensuing steps, tens of thousands of messages could be lost.

```
su - imail
setenv EDITOR vi
crontab –e
/bin/ps –ef | grep gc
exit
```

16. Run the Oracle server manager utility `svrmgrl` by entering:

```
>connect internal
>startup mount
>set echo on
>@/<backup-area>/FilesOfflineTEMP.sql
>@/<backup-area>/IndexFilesOffline.sql
```

Oracle should print "`Statement processed`" in response to each SQL command in the script. If it prints a message beginning with "`ORA-<nnnnnn>`", where `<nnnnnn>` is a number, you must address the problem before continuing the recovery.

17. If there are Oracle errors, look for the following line in the `.sql` file: "`Connect imail/imail@IMM1`" (or the like). In another window and without terminating `svrmgrl`, change it to read "Connect imail/imail." Then try this step again. The script `FilesOfflineTEMP.sql` is created by `imdbhotbackup`.
18. Look for other files named `FilesOfflineFOO.sql` in the backup destination directory. Unless your DBA has created extra temporary tablespace, you won't find any. If you do, run these scripts in the same fashion.
19. Check whether the archived redo log file generated by the database, beginning just before the oldest restored hot backup file up until as recently as possible, are in the default archived redo log destination for the database, as specified in the `log_archive_dest.init.ora` parameter. The log files should be here, unless you are using a hot backup more than a day old or the `log_archive_dest` was lost. However, if they are not, enter one of the following commands:
    - For a Solaris platform:

      ```
      >set logsource /<directory-where-archived-redo-logs-are>
      ```

    - For other platforms:

      ```
      >set logsource /<directory-where-archived-redo-logs-are/a>
      ```

20. Make sure all the archive redo log files are both readable and writeable by `oracle`. To do this, enter:

    ```
    find /directory-where-archived-redo-logs-are –name \*.log –
    exec chmod u+rw \{\} \;
    ```

21. Recover the database by entering:

```
>recover  automatic database [until time 'YYYY-MM-DD:HH:MM:SS'] [until
cancel] using backup controlfile parallel 6;
>alter database open resetlogs;
>@/<backup-area>/RecreateTempTbsTEMP.sql
>@/<backup-area>/RecreateIndexTablespaces.sql
```

Use the the `until time` clause if you want to recover the database up to a specific moment. Otherwise, use the `until cancel` clause. In this case, when the database runs out of archived redo logs to play and prompts you for what to do, enter `CANCEL`.

Note: The best setting of the value of the parallel clause depends on the power of your hardware. You might want to experiment with different values on a test box.

Oracle should print "`Statement processed`" in response to each command in each SQL script. If it prints a message beginning with "`ORA-<nnnnnn>`", where

<nnnnnn> is an Oracle error number, you must address the problem before continuing the recovery. One common reason for this step to fail is that the `TemporaryTablespace.dbf` file already exists. If it does, delete it and rerun the script:

```
>@/<backup-area>/RecreateTempTbsTEMP.sql
>@/<backup-area>/RecreateIndexTablespaces.sql
```

22. If there are Oracle errors, look for the following line in the `.sql` file: "`Connect imail/imail@IMM1`" (or the like). In another window and without terminating svrmgrl, change it to read "`Connect imail/imail`." Then try this step again. The script `FilesOfflineTEMP.sql` is created by `imdbhotbackup`.
23. If there are any other `RecreateTempTbsFOO.sql` scripts in the backup destination directory, run them now.
24. Look for a file named `RecreateIndexTablespaces.*.log` in the current working directory. This file will contain the output from running the script file. To verify that no errors occurred, do a `grep ORA-RecreateIndexTablespaces*.log`.
25. Remove the comment symbol (#) from the `crontab` entry for `immssgc`.
26. If necessary, restart the Listener:

    ```
    lsnrctl status
    lsnrctl start
    ```

    The database is now as fully recovered as it can be. Still, because the only redo logs were lost, so were the effects of the most recent transactions. The most recently accepted messages have been lost and the most recently deleted messages still exist. If the garbage collector `immssgc` was running when changes were lost, many mailboxes can contain messages (widows) that the garbage collector deleted. When a user attempts to retrieve a bad message, the user will get an error, and InterMail will move the bad message from the user's inbox to the user's error folder.

Oracle Tablespace/Data File Failure and Recovery

This section describes how to simulate an Oracle tablespace or data file failure by shutting InterMail down and removing a specific tablespace or data file such as `POM01`, and then restoring the data files from the latest backup and applying Oracle redo logs to restore the previous system.

Test Procedure

1. Create test accounts and send test messages.
2. Shut down InterMail and Oracle:

    ```
    % lib/imservctrl stop
    % lsnrctl stop
    % shutdownDB IMM1 | IMD1 // or MSS Oracle SID
    ```

3. Delete the Oracle `POM01` tablespace datafile.

4. Restore the data file set from the last available hot backup.
5. Shut down the Oracle services.
6. Start up the database and recover using the redo log files, as described in “Oracle Home Directory” on page 208.
7. Restart the InterMail and Oracle services:

```
% startupDB IMM1 // or MSS Oracle SID
% startupDB IMD1  // or ISD Oracle SID
% lsnrctl start
% lib/imservctrl start
```

8. From the POP server, attempt to retrieve the test messages created in step 1.
9. Create new test mail accounts and messages to test the restored services.

Oracle Control File Loss and Recovery

This section describes how to simulate an Oracle control file failure by shutting InterMail down and removing a specific Oracle control file such as `ctrl$ORACLE_SID01.ctl`, then restoring the control file from the latest backup and applying Oracle redo logs to restore the previous system.

Test Procedure

1. Create test accounts and send test messages.
2. Shut down InterMail and Oracle:

```
% lib/imservctrl stop
% lsnrctl stop
% shutdownDB IMM1 | IMD1 // or MSS Oracle SID
```

3. Delete the Oracle control file `ctrl$ORACLE_SID01.ctl`.
4. Restore the backup control file from last available hot backup.
5. Start up the database and restore using the redo log files, as described in “Oracle Home Directory” on page 208.
6. Shut down the Oracle services.
7. Restart the InterMail and Oracle services:

```
% startupDB IMM1 // or MSS Oracle SID
% startupDB IMD1  // or DIR Oracle SID
% lsnrctl start
% lib/imservctrl start
```

8. From the POP server, attempt to retrieve the test messages created in step 1.
9. Create new test mail accounts and messages to test the restored services.

Message Deletion and Recovery

This section describes how to simulate and recover from accidental deletion of messages by shutting down InterMail, moving specific message files from the Message File system to a temporary area, attempting to retrieve these messages from the POP server, and restoring the message files.

Test Procedure

1. Create test accounts and send test messages.
2. Move the test message files you want to test from the Message File system to a temporary location.
3. Attempt to run these messages and receive the `-ERR` message; this places the message entry in the database to a `.ERROR` folder for later repair.
4. Move the message files back to their original locations.
5. Run `imbadmsglist` to produce a message list for `imbadmsgfix` to repair.
6. Attempt to run POP again on these messages.
7. Create new test mail for the affected accounts to test the restored services.

12

Troubleshooting

InterMail is a robust messaging system designed for continuous use seven days a week, 24 hours a day. To best prepare yourself for managing a mail server in this environment, it is important to understand the relationships between system components and the overall impact on mail service should one component become unavailable.

To assist you in obtaining the necessary understanding, this chapter describes a model mail system to use in discussing a variety of troubleshooting scenarios. The intent is not to provide a definitive list of issues and solutions, but rather to equip you with an understanding of the entire system in relation to its individual parts.

For each scenario, the following information appears:

- Immediate effect on the system
- Impact on service
- Possible actions to take to remedy the situation

In addition, several miscellaneous scenarios are presented at the end of this chapter, addressing various problems that can occur throughout the lifespan of a post-installation InterMail system.

Sample System Configuration

The troubleshooting scenarios that follow refer to the sample InterMail system shown in Figure 22. The intent of this diagram is not to represent the typical InterMail installation. In fact, given the flexibility of InterMail, no such standard exists. This sample is one of many viable InterMail configurations. Your specific installation will undoubtedly vary, and you should modify your associated troubleshooting procedures accordingly.

The sample system includes two hosts dedicated to message delivery, four hosts dedicated to message storage, and two hosts dedicated to servicing client requests for mail retrieval. In addition, the sample system includes a host dedicated to system management, and another host dedicated to the storage of account and domain information.

Each message delivery host runs a Message Transport Agent (MTA) and a Directory Cache Server (Hosts A and B in the diagram). Each host dedicated to message storage runs a Message Store Server (MSS) and a Queue server (Hosts C, D, E, and F in the diagram). The hosts responsible for message retrieval run a POP server and an IMAP server (Hosts G and H in the diagram). The host responsible for system management (Host G) has a Configuration server and an SNMP server installed. The Integrated Services Directory (ISD) installed on Host H maintains account and domain information.

Note: A Manager server also runs on each host.

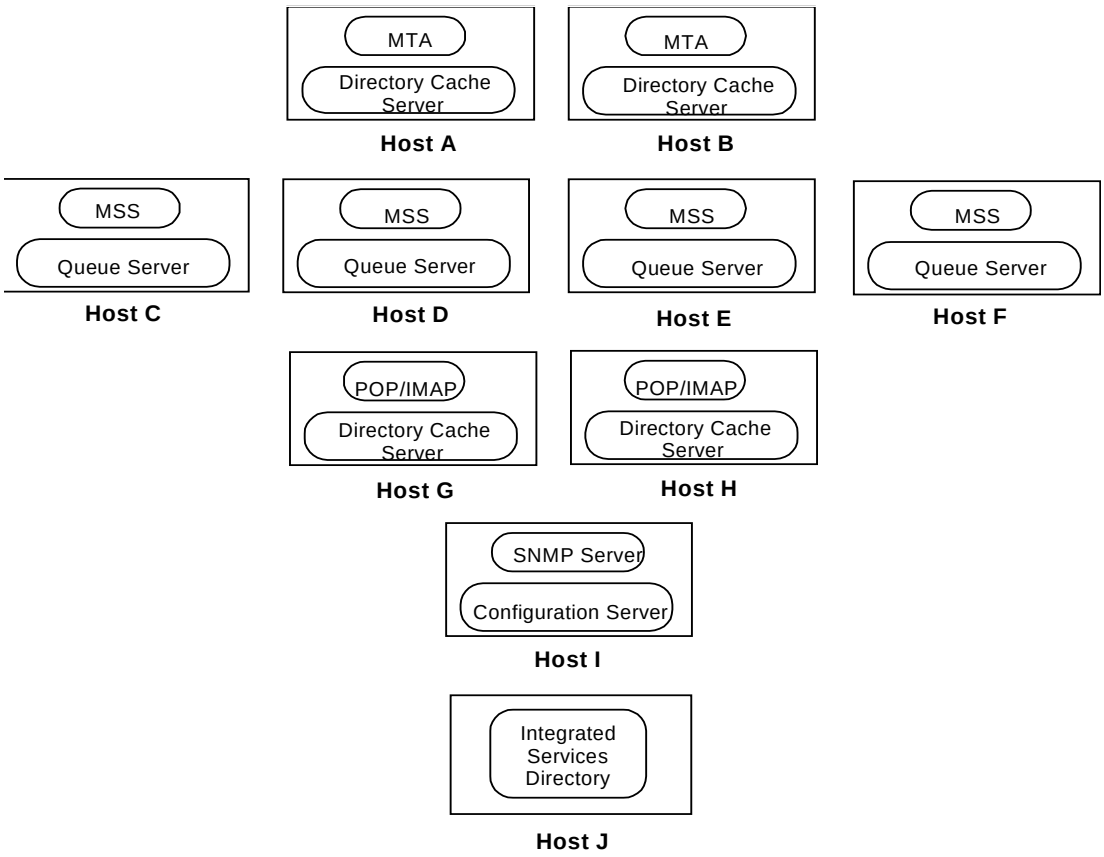

Figure 22 Sample system configuration

To simplify the troubleshooting discussions and concentrate on the most important issues, this scenario makes the following assumptions:

- For this site map, the domain name system (DNS) records the domain name associated with this mail server to multiple IP addresses. When the DNS server performs name resolution, it “round robins” through the records, distributing the load for the domain among several machines.
- The `queueServerHosts` configuration key allows each MTA to locate multiple Queue servers. The primary Queue server (the one the MTA typically contacts) is same host as the MSS with which the MTA is requesting contact.
- The `dirCacheHosts` configuration key allows each MTA, each POP server, and each IMAP server to locate multiple Directory Cache servers. The primary Directory Cache server (the one each server typically contacts) is the one on the same host as the server requesting contact.
- Although the configuration of Host F is fully complete, the servers on that machine are not in production. The sample system reserves this host as a backup that is not actively linked to the production environment and that can substitute for another MSS host if necessary.

Sample Scenarios

This section describes scenarios with respect to various InterMail servers. They refer to particular incidents but can be generalized to other, similar circumstances.

Directory Cache Server Is Unavailable

The Directory Cache server responds to requests for account information. It interacts with the MTA, the POP server, and the IMAP server, supplying and verifying account information as necessary.

The design of the InterMail messaging system allows one Directory Cache server to take over for another automatically, if required. The configuration of the sample system supports this feature.

Scenario

The Directory Cache server on Host A is unavailable.

Effects

When the MTA on Host A is unable to contact the Directory Cache server on Host A, it sends the request for account information to the Directory Cache server on Host B. The MTA continues to send the next 99 requests for account information to the Directory Cache server on Host B. For the 100th request, it re-attempts contact with the Directory Cache server on Host A. If the Directory Cache server on Host A responds, standard operation resumes. If the MTA still cannot reach the Directory Cache server on Host A, it continues sending requests to the Directory Cache server on Host B.

Impact on Service

There is no noticeable effect on mail delivery or retrieval.

Actions

1. Check the Directory Cache server log files for the cause of failure. Bring the Directory Cache server on Host A up again, if possible.
2. If the Directory Cache server on Host A has been down for less time than the value in the `logAgeHours` configuration key, no further action is necessary. It will automatically synchronize the content of its local cache with the content of the ISD on the next update cycle (every 60 seconds, by default).
3. If the Directory Cache server on Host A has been down longer than the value in the `logAgeHours` configuration key, it will be unable to synchronize its cache automatically. Instead, you must either copy a cache file from one of the other Directory Cache servers (on Host B or C) or run the `imdirsync` command to create a new cache file directly from the ISD.

MTA Is Unavailable

The MTA handles receiving all incoming messages. For mail addressed to local users, it obtains account information from the Directory Cache server, then hands the message to the appropriate MSS. For recipients in other domains, it sends the message to the remote location over the Internet.

Scenario

The MTA on Host B is unavailable.

Effects

Round-robin DNS redirects mail to the MTA on Host A. Ordinarily, there are no issues involving spooled mail. However, if no Queue servers are running, the MTA host may have spooled mail locally. Any deferred mail stored temporarily on Host B continues deferred until the MTA on Host B returns to service.

Impact on Service

Mail delivery may be slow because the MTA on Host A is carrying the entire burden. There is no effect on mail retrieval.

Actions

1. Check the MTA log files for cause of failure. Bring the MTA server on Host B up again, if possible.
2. If it appears that the server will be down for more than a few minutes, you should start up an MTA on an alternate InterMail machine and roll the IP address for Host B to that machine.

MSS Is Unavailable

The MSS is responsible for hosting mailboxes, storing incoming messages, and providing access to clients' mailboxes. Large InterMail systems may have several MSS machines.

Each MSS has its own database, which contains a unique set of mailboxes. The ISD maintains information about these mailboxes, including their associated accounts and their exact physical location.

Message storage is transaction-based, and journals record transactions for full data integrity and recoverability.

Scenario

The MSS on Host C is unavailable.

Effects

The system defers internally all mail destined for users with mailboxes managed by this MSS. The MTAs pass the mail to the Queue servers for temporary deferral and re-attempt delivery on a regular basis. See Chapter 7 for more information about delivery of deferred mail.

Impact on Service

Users with mailboxes managed by this MSS are unable to retrieve their mail. There is no effect on mail delivery and retrieval for users whose mailboxes reside on Hosts D and E.

Actions

1. Restart the MSS and resume service; check the MSS log files for the cause of the failure.
2. If it is not possible to restart the server, *and* the MSS database and the Message File system are fine, *and* Host C is dual ported to Host F (the hot spare), then start up an MSS on Host F, roll the IP address from Host C to Host F, and allow Host F to take over.
3. If Host C and Host D are dual-ported, you could fail the MSS on Host C over to Host D. However, be aware that the combined burden cannot exceed 100% of the capacity of Host D. You must also watch for conflicts in MSS ports.

Note: If the MSS and Queue servers are unavailable, local spooling needs to be enabled for spooling to occur. To set up local spooling, set the following keys:

```
/*/mta/localFallback: [true]
/*/mta/stateless:     [true]
```

Message Store Database Is Unavailable

The Message Store database contains data about message store content (folders and message file pointers) along with indexing information for message headers and MSS statistics.

Scenario

The Message Store database on Host C is unavailable.

Effects

The system defers internally all mail destined for users whose mailboxes reside on Host C. The MTAs pass the mail to the Queue servers for temporary deferral and re-attempt delivery on a regular basis.

Impact on Service

Users whose mailboxes reside on Host C are unable to retrieve their mail. There is no effect on mail delivery and retrieval for users whose mailboxes reside on Hosts D and E.

Actions

1. Shut down the MSS pointing to this database.
2. Restore the last good backup of the Message Store database.
3. Restore the archived redo logs.
4. Restart the MSS. (This restores service to all users.)

Message File System Is Unavailable

Message files contain all message data: header, body, and attachments. InterMail stores one message file per message on each MSS host, regardless of the number of recipients to whom the message is addressed.

The effect and recommended response to message file loss varies based on the number of files in question. The scenarios described in this section illustrate the differences in impact and approach.

Scenario 1

A small portion of the Message File system, one leaf directory on Host C, is lost.

Impact on Service

Mail delivery continues uninterrupted. Some users with mailboxes on Host C are unable to retrieve a portion of their mail. There is no effect on mail retrieval for users whose mailboxes reside on Hosts D and E.

Actions

1. Leave the MSS up and running.
2. Copy the directory from your latest file system backup.

3. Run `imjrnrecover` to bring the content of the directory up-to-date.
4. Run `imbadmsglist`. It produces a list of messages that the system could not retrieve using the POP server and therefore moved from a user's `INBOX` folder to the user's `.ERROR` folder.
5. Run `imbadmsgfix` to move the restored messages back to the user's `INBOX` folder.

Scenario 2

The entire Message File system on Host C is lost.

Effect

The system defers internally all incoming mail destined for users with mailboxes on Host C.

Impact on Service

Users with mailboxes on Host C are unable to retrieve their mail. There is no effect on mail retrieval for users whose mailboxes reside on Hosts D and E.

Actions

1. Stop the MSS on Host C.
2. Copy the backup file system.
3. Play the journal files to bring it up to date.
4. Run the `imbadmsglist` and `imbadmsgfix` utilities.
5. Restart the MSS.

For more information on restoring the Message File system, see Chapter 11.

Queue Server Is Unavailable

If the MTA cannot deliver a message immediately (as when a remote host is temporarily offline), it passes the message to a Queue server, which stores it in the `queue` directory. At regular intervals, the MTA requests stored messages from the Queue server and attempts to deliver them again. The Queue server also provides temporary storage of messages that exceed the MTA's limits for message size, number of recipients, or time required for delivery.

The design of InterMail allows one Queue server to take over for another automatically, if required. The configuration of the sample system supports this feature.

Scenario 1

The Queue server on Host C is unavailable.

Effects

When the MTA on Host A is unable to contact the Queue server on Host C, it sends the request for temporary mail storage to the Queue server on Host D. The Queue server on Host D provides temporary storage in its `queue` directory. The MTA periodically re-attempts contact with the Queue server on Host C. If the Queue server on Host C responds, standard operation resumes. If the MTA still cannot reach the Queue server on Host C, it continues sending requests to the Queue server on Host D.

Impact on Service

There is no noticeable effect on mail delivery or retrieval.

Actions

Bring the Queue server on Host C up again, if possible. Check the Queue server log files for the cause of failure.

Scenario 2

The Queue directory on Host C is lost.

Impact on Service

There is no effect on mail delivery or retrieval. Previously deferred mail remains undelivered until the system restores the contents of the Queue directory.

Action

1. Shut down the affected Queue server.
2. Restore latest backup of the `queue` directory.
3. Run `imjrnrecover` to restore the lost files.
4. Restart the server.

POP Server Is Unavailable

The POP server handles requests for message retrieval from any client that supports the POP3 protocol. It communicates with the Directory Cache server to validate the user's login name and password, and to obtain the name of the MSS host on which the user's mailbox resides. The POP server also communicates with the MSS to service requests for message retrieval on behalf of the client.

Scenario

The POP server on Host G is unavailable.

Effects

Round-robin DNS will send client requests to the next POP server (on Host H).

Impact on Service

There is no ffect on mail delivery. Mail retrieval may suffer slightly because there is an extra load on the POP server on Host H. (If the server on Host H is too busy, it may refuse some client connections.)

Actions

1. Restart the POP server on Host G. Check the log files for the cause of failure.
2. If it appears the server will be down for more than a few minutes, you should start up a POP server on an alternate InterMail machine and roll the IP address for Host G to that machine.

IMAP Server Is Unavailable

The IMAP server handles requests for message retrieval from mail clients that support the IMAP protocol. It communicates with the Directory Cache server to validate the user's login name and password, and to obtain the name of the MSS host on which the user's mailbox resides. The IMAP server also communicates with the MSS to service requests for message retrieval on behalf of the client.

Scenario

The IMAP server on Host G is unavailable.

Effects

Host G will drop clients connected at the time of the failure. Round-robin DNS will send new requests for IMAP access to the next IMAP server (on Host H).

Impact on Service

There is no effect on mail delivery. Mail retrieval will suffer because there is extra load on the IMAP server on Host H. (If the server on Host H is too busy, it may refuse some client connections.)

Actions

1. Restart the IMAP server on Host G. Check the log files for the cause of failure.
2. If it appears that the server will be down for more than a few minutes, you should start up an IMAP server on an alternate InterMail machine and roll the IP address for Host G to that machine.

Configuration Server Is Unavailable

The InterMail system includes a single master configuration host for the Configuration server. The Configuration server maintains the master Configuration database and is responsible for distributing the contents of that database to all the other InterMail hosts.

Scenario

The Configuration server is unavailable.

Effects

Servers regularly try to reconnect to the Configuration server (every 15 seconds). Because the Configuration server is unavailable, these attempts fail with `ECONNREFUSED`. The servers attempting connection record this event with an entry in their log files.

The configuration editing utility, `imconfedit`, runs in local mode, meaning that it cannot synchronize with the master Configuration database. It cannot assess the impacts of changes, and it cannot propagate these changes to running servers.

Any attempted installations fail because they need the Configuration server to begin running.

Impact on Service

There is no effect on mail delivery or retrieval. However, it is more difficult to propagate configuration changes. If you could not get the Configuration server started again and you had to make a configuration change, you would have to manually copy the master Configuration database by hand to each InterMail host and then restart the servers on those hosts to force them to read the new configuration settings.

Actions

Go to the Configuration server host, and as the InterMail user (`imail`, by default) start the Configuration server (`lib/imservctrl start imconfserv`). Check the log file to verify that whatever brought the server down is not preventing it from starting up again.

ISD Is Unavailable

The ISD stores domain and account information. The Directory Cache server responds to requests for account information. It has access to the ISD but services most requests directly from its cache, which contains a local copy of the required account information.

Scenario

The ISD is unavailable.

Effects

The Directory Cache servers respond to all requests with information from their local cache files.

Impact on Service

There is no noticeable effect on mail delivery or retrieval. However, you cannot add, delete, or change account information until the database is restored to service.

Action

Try to bring the ISD up again.

Miscellaneous Troubleshooting Scenarios

This section addresses miscellaneous troubleshooting scenarios that can occur in an InterMail system.

Maintaining Login Names for Multiple Domains

In the course of operations, a service provider that is currently handling mail for one domain may become responsible for managing mail from additional domains. When this happens, multiple existing users may have login names that conflict.

For example, whereas originally there may have been one user with a login name of `jdoe`, there may now be two users with the login `jdoe`, one at `software.com` and one at `example.com`. Although these users are in domains with different names, if the mail system uses only login names when authenticating through a directory service, the two `jdoe` users will now be in contention.

In order to allow existing users to maintain their login names and to guarantee uniqueness for these login names, the InterMail system provides domain-based login completion through the `loginDefaultDomainTable` configuration key. This key causes InterMail to fully qualify a user's login name by adding a domain name suffix to the login name when the user logs in from a known IP address. For example:

```
loginDefaultDomainTable: [10.2.62.22: software.com]
                         [13.1.43.23: example.com]
```

In this example, the originating IP addresses are specified for both domains in the system. This causes both the domain name to be appended to a user's login name when the user attempts to make a client connection and this fully qualified login name to then be used in the authentication process.

Note: By default, the `loginDefaultDomainTable` configuration key is empty. In order to enable domain-based login completion, you must specify the IP addresses for all domains in your system.

Disabling the Welcome Message

If the `MsMsgIdNotEvent` log message appears in `mss.log`, it may be due to a user's having disabled the welcome message improperly. If you do not a want a welcome message sent to new users, you must remove the `/*/mss/welcomeMsgID` configuration key from `config.db`. If this key is set to null (`/*/mss/welcomeMsgID: []`), the MSS will attempt to add a welcome message to new mailboxes with a message ID of null. In this situation, there will be no message with this message ID, and `Warn;MsMsgIdNotEvent` will be recorded in `mss.log`.

Note: If `welcomeMsgId` is set to `" "` and there is a message that corresponds to `" "`, this message will inadvertently be placed in all new mailboxes.

Changing a Domain to a Rewrite Domain

If you attempt to change a local domain to a rewrite domain, it is possible to receive the following error:

```
Domain Type Change Not Allowed, Delete Domain First
```

This problem occurs when the domain contains accounts; a domain should not be changed to a rewrite domain when users belong to that domain.

If you wish to change an existing domain to a rewrite domain, you must:

1. Move and/or delete the accounts from that domain.
2. Delete the domain.
3. Re-create the domain as a rewrite domain.
4. If you so choose, add accounts to that domain.

Processing Messages in the Queue

During certain conditions, InterMail's message queue may contain an excessive number of messages waiting to be processed.

Possible symptoms to indicate this condition include:

- Manual or scripted observation of a large number of files in the `$INTERMAIL/queue/control` directory that have not been processed over a period of time.

  Note: The determination of what constitutes a "large number" is site-specific and involves length of average processing time as well as number of files.

- Repeated invocations of `immtacheck` reporting an increasing number of messages waiting to be processed.
- `NioServerGoneDown` events recorded in `mta.log`.

If you want to process these messages, it is recommended that you lower the value set in the `deferProcessInterval` configuration key. This will cause the Control files to be processed more quickly, although the amount of time it takes for the queue to empty will vary.

A more immediate solution is the following:

1. Shut down the Queue server.
2. Move any Control files in the `$INTERMAIL/queue/control` to a temporary location, such as `queue/deferred/MTA-save`.
3. Find and remove widowed Control files.
4. Re-introduce messages from the `deferred/MTA-save` directory to `deferred/MTA` at the interval specified by `deferProcessInterval` by moving them in batches.

5. Restart the Queue server.
6. Shut down the MTA.
7. Restart the MTA.

Creating Accounts

This section describes problems involving account creation.

Nonexistent Default Domain

During a pre-production testing period in operations, the Directory and LDAP databases may be reset . If during this period you do not re-create the default domain or set it as the default domain, a subsequent attempt to add accounts will produce an error such as:

```
imdbcontrol: Oracle Error executing command ca
Domain not known or not active
```

If you experience this condition, you should re-create the default domain and set it as the default domain. For example:

```
imdbcontrol cd software.com
imdbcontrol sdd software.com
```

After performing these operations, you will be able to add accounts as before.

Unsynchronized Accounts in the Directory and LDAP

If during `imbatchload` operations you have uncommented out the following lines in the `sitestartup.batch` file (or other user-defined input file that you are using in conjunction with `imbatchload`), `imbatchload` will create accounts in the LDAP database, but not in the Directory database:

```
Option:
errorOnNonExist: account
```

Attempts to manipulate these accounts through `imdbcontrol` will fail, and these accounts will not be available for mail.

In this case, you should comment out the lines in the input file and rerun `imbatchload`. The accounts will then be visible to command-line utilities and be active.

A

License Information

This appendix contains license information related to InterMail Mx.

InterMail Licensing Agreement

THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL SOFTWARE.COM BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

EMANATE

InterMail incorporates the EMANATE server as part of its monitoring functionality. Software.com licenses EMANATE pursuant to a license agreement with SNMP Research International, Incorporated. The copying and distribution of EMANATE is with the permission of SNMP Research International, Incorporated.

GNU General Public License

Version 2, June 1991

Copyright (C) 1991 Free Software Foundation, Inc.

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the library GPL. It is numbered 2 because it goes with version 2 of the ordinary GPL.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

GNU LIBRARY GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) The modified work must itself be a software library.

   b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

   c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

   d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

   (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

   Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

   This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

   If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

   However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

   When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

   If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

   Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

    You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARIS-

ING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Oracle

Oracle Programs are the proprietary products of Oracle and are protected by copyright and other intellectual property laws. Customer acquires only the right to use Oracle Programs and does not acquire any rights, express or implied, in Oracle Programs or media containing Oracle Programs other than those specified by License. Oracle, or its licensor, shall at all times retain all rights, title, interest, including intellectual property rights, in Oracle Programs and media.

The Regents of the University of California Copyright

InterMail includes software that is copyright © 1990, 1993, 1994, The Regents of the University of California. All rights reserved.

This code is derived from software contributed to Berkeley by Mike Olson.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Re-distributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Re-distributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The Regular Expression Routines

1. Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it, subject to the following restrictions:
2. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
3. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.

4. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.
5. This notice may not be removed or altered.

RSA Data Security, Inc. MD5 Message-Digest Algorithm

License to copy and use this software is granted provided that it is identified as the “RSA Data Security, Inc. MD5 Message-Digest Algorithm” in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as “derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm” in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided “as is” without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

Glossary

absolute pathname

The path to a file beginning at the root directory. See also *relative pathname*.

account

A directory entry containing attributes for a user. An account is used by InterMail and other applications, and includes one or more e-mail addresses, instructions for mail delivery, and auto-reply information. An account typically corresponds to an individual computer user, and is typically associated with a mailbox that stores the messages that the account receives. An account is synonymous with an LDAP Person entry in the ISD.

account class-of-service attribute value

A value for a class-of-service attribute that is relevant only to a specific account. The Directory cache uses it when returning a resolved class-of-service attribute value to a client in the evaluation of a class-of-service rule. In some situations, the account class-of-service attribute acts as a "user preference." When present, an account class-of-service attribute value takes precedence over, or overrides, the corresponding class-of-service attribute value.

See also *class-of-service attribute value.*

account status

An account attribute that defines the current state of the account. The status of an account determines whether normal delivery of mail should occur for the account. The available account status types are Active (normal delivery), Maintenance (message delivery is temporarily delayed), Suspended (message delivery is denied), Deleted (message delivery is permanently disabled), and Proxy (delivery occurs to another mail system).

address completion

The automatic correction of an e-mail address that is not SMTP-compliant. As required by the SMTP protocol, recipient addresses must include both a username and a domain. If a username is present, but not a domain, InterMail appends a default domain name to the username to form a complete address. For example, if the address completion domain is `software.com`, and mail arrives for `john.doe`, the InterMail system completes the address by appending the default domain name and creates the recipient address `john.doe@software.com`.

Address completion is carried out only when a message would be otherwise undeliverable because of an address error.

administration utility

A command-line tool used to search for and modify data and to observe and change behavior in an InterMail system. There are commands for directory, account, mailbox, server, and log management, as well as for monitoring system diagnostics.

administrative mailbox

A mailbox that stores default account information for a new user's mailbox, including new account information, default mailbox data, and customer support. An administrative mailbox differs from other mailboxes in that it is strictly for administering account information, not for message retrieval.

alias

1. An alternative name that is usually short and easy to remember.
2. On Web servers, a way to map an incoming request for a Web page. When an alias appears in a URL, the original address replaces the alias.

API (application programming interface)

A set of routines that defines how programmers may access a particular InterMail service, such as the ISD, a mailbox, or a log file. APIs are typically used to enable utilities to access InterMail services, and to enable provisioning and other applications to be integrated with InterMail.

attachment

Portions of a message that are separate from the main body of the mail message and are not automatically displayed in a mail client, instead requiring some further action from the user.

attribute

Information describing one trait of a directory object. Each attribute has a name followed by one or more values. For example, the attribute `o`, belonging to the object class `organization`, could have the value `Software.com`. The attribute `ou`, belonging to the object class `organizationalUnit`, could have the value `engineering`.

attribute constraint

A parameter assigned to an attribute that determines which administrators and users (if any) may create and modify it, and which legal values they can assign to it.

attribute value

The data associated with an attribute. For example, `555-1212` is an attribute value for the attribute `telephoneNumber`.

authentication

The ability of one entity to determine the identity of another entity. Typically, this involves the use of a username and password pair or of a proprietary key.

auto-reply feature

A feature associated with InterMail accounts. When mail arrives for an account that uses the auto-reply feature, InterMail automatically sends the account's auto-reply message to the sender of the original message.

availability

See *high availability.*

blocking

See *mail blocking*.

Body file

A temporary file that contains the text of a message and any attachments. It is created, along with the associated Control and Header files, when mail is secured on disk.

bounce message

A message that the MSS transmits to the sender of a message that has bounced. Also called *bounce notice.*

bouncing

The process of returning an undeliverable message to its sender.

bucket

A subdirectory for storage of message files or message components (such as a message body, or message Control files). On the MSS and Queue servers, the use of buckets for load balancing enhances system performance.

canonicalization

A method of completing `Mail-From` addresses on incoming messages. When the InterMail canonicalize feature is active, the default domain name (defined in the `defaultDomain` configuration key) is used to complete `Mail-From` addresses that lack a domain. Addresses completed in this way are said to have been *canonicalized*.

certificate

A key used as part of an authentication strategy for users. InterMail uses certificates during MTA and all transactions involving SSL.

child process

A process created by another process. The creating process is the parent process.

class of service

A set of permissions, resource limits, and default user preferences shared by a set of accounts that determines, among other things, the services that the users of each associated account may access. Each permission, resource limit, and preference is represented by a specific class-of-service attribute.

class-of-service attribute

An attribute of a class-of-service entry in the directory. Each class-of-service attribute defines a permission, resource limit, or preference that affects the set of accounts associated with the class of service.

class-of-service attribute rule

A value that defines the relationship between a class-of-service attribute and the account class-of-service attribute values (for example, whether the account class-of-service attribute value takes precedence over the class-of-service attribute value).

class-of-service attribute value

The value assigned to a class-of-service attribute that applies to all accounts associated with the class of service. Sometimes this is referred to as a *global* COS attribute value to differentiate it from the *account* COS attribute value.

See also *account COS attribute value.*

Configuration database

The text file that contains all configuration keys and their associated values.

configuration key

A parameter defined in the master Configuration database and replicated on each host's local Configuration database (`config.db` file) that represents the local and global configuration settings for all InterMail servers. Configuration keys are represented in the form "path/name:value", where "path" defines the scope of the key, "name" identifies the key, and "value" specifies the value of the key.

Configuration server

The InterMail server that holds the master Configuration database and distributes the latest version of it to all InterMail hosts. The Configuration server runs on a single host.

consumer server

The destination server for replicated data. In InterMail, the consumer server is the Directory Cache server.

Control file

A temporary file that contains information about a message, including its current status in the delivery process and the names of the corresponding Header and Body files.

cookie

A piece of information that a Web server provides to a Web client for identification. This information contains data that the server can retrieve later. In addition to tracking details as you browse the Internet, cookies also help the server remember when you previously visited a site.

deferred mail

Mail that the MTA cannot deliver immediately, either because a remote mail host is down or because the MSS or Directory Cache server is temporarily unavailable. Deferred mail goes to the Queue server for later delivery.

Directory Cache server

A component of the ISD that contains a copy of all or part of the master Directory database in its local cache database. The Directory Cache server is capable of servicing the same read and write requests as the Directory server and is regularly updated from the Directory server, preventing bottlenecks to the master Directory database and ensuring quick response time to queries by other servers.

Directory database

A single Oracle relational database that is the authoritative master version of InterMail account information, including an end user's domain, username, password, class of service, e-mail address, and delivery information. Directory cache servers communicate with the Directory server, which in turn accesses the Directory database, to get the updates to their local cache databases. The Directory database is a component of the ISD, together with the Directory server, Directory Cache servers, and Directory Cache databases.

Directory Information Tree (DIT)

An LDAP term for the hierarchical structure that contains all directory entries. Each directory entry is uniquely identified by a distinguished name (DN), which is a path in the DIT used to name and locate the entry, similar to a file's path in a file system.

Directory server

The component of the ISD that maintains an authoritative master copy of the Directory database. It contains Oracle client libraries to communicate with the Directory database and is responsible for storing replication information that is read by the Directory Cache servers.

disk array

A group of multiple physical drives tied together into logical units (LUNs) to support disk striping and/or mirroring. Also called *RAID (Redundant Array of Inexpensive Disks)*.

disk striping

The writing of data blocks across multiple disks in order to enhance throughput.

distinguished name (DN)

An LDAP term for the name of a directory entry that uniquely identifies the entry in the directory. The distinguished name consists of a sequence of attribute name / attribute value pairs that shows the location of the entry in the Directory Information Tree (DIT). For example:

```
dn: uid=johndoe, ou=engineering, dc=software, dc=com
```

distributed architecture

A system design that allows a single software application to span several independent pieces of hardware. Some benefits to a distributed architecture system are incremental/modular backup, possible failure only of single points (as opposed to failure of the entire system), security, and scalability.

domain

One or more IP addresses corresponding to a particular organization. For example, `software.com` is a domain that contains addresses in the 10.2.6.x Class C IP network.

domain name system (DNS)

A system in which names are assigned to IP (Internet protocol) addresses. This structured system contains a hierarchy based on a proper name and type of organization, such as `.org`, `.com`, `.edu`, or `.mil`. DNS has two primary benefits. First, it provides users with a convenient and easy path to find an organization (with human-readable names instead of nonintuitive numbers). Second, it supports name lookup with a system of name servers that are dedicated to maintaining DNS records, polling for DNS records, and recording new entries.

draining (of a server)

The process of shutting down a server without interrupting any current client connections. When a server is drained, it does not accept any new connections and waits for all of its current connections to close before shutting down.

See also *stopping*, *killing*.

DSN (delivery status notification)

Return notification sent to e-mail senders on other systems that also support DSN informing them of e-mail delivery success, failure, or delay.

e-mail address

The combination of a username, followed by an `@` symbol, followed by a domain name. For example, `joe.schmoe@software.com` is an e-mail address, where `joe.schmoe` is the username and `software.com` is the domain name.

empty address

An envelope or header address field that contains no information.

entry

The basic unit of information stored in a directory. Entries consistsof a collection of attributes.

envelope

The portion of an InterMail message that contains the `RCPT TO` and `MAIL FROM` addresses, allowing the message to be delivered to its proper recipients. Unlike headers, the envelope is not a visible part of the message.

See also *header.*

environment variable

A variable that defines how the user's environment responds to commands. A user or a program may set an environment variable. Common examples of environment variables include `PATH` statements and library paths.

ESMTP (extended simple mail transfer protocol)

The protocol used by the server that processes mail queue requests when the ETRN command is executed.

ETRN (extended turn)

A mail-queue-processing option used in conjunction with SMTP. ETRN instructs remote mail hosts to attempt delivery of queued messages. This is useful if your server has a PPP or SL/IP connection or a similar intermittent connection to the Internet. ETRN is intended for use by connecting mail servers, but you can also execute the command manually to request queue processing.

failover strategy

A strategy in which, in the event of machine failure, an alternate machine assumes the network identity of the failed machine and accesses its disk array through a second port. MSS hosts should use a failover strategy, because they host message data information that must be accessible at all times.

filter

1. A program that accepts a certain type of data as input, transforms it, and then generates the transformed data. For example, a program that sorts names is a filter because it accepts the names in unsorted order, sorts them, and then generates the sorted names.
2. A pattern through which data is passed. Only data that matches the pattern can pass through the filter.

folder

A container for messages within a mailbox. By default, all InterMail mailboxes contain three folders: `INBOX, SentMail`, and `Trash`. Although POP mail users cannot see the folders, their messages automatically come from their `INBOX` folders.

Users with IMAP accounts can see folders within their mailboxes. They can create, delete, and move folders, as well as nest folders within other folders.

forwarding

See *mail forwarding.*

FTP (file transfer protocol)

A set of systems and interfaces for transferring files across the Internet. Typically, FTP downloads files from a host machine or from an archive site to a user's computer, or uploads files from a user's computer to a host machine.

hashing

1. The scrambling of a plain-text string (such as an account password) into an apparently random string from which the original plain text cannot be recovered.
2. The provision of rapid access to data items in a database through distinguishing keys. A hash function acting on the item's key produces a hash value that is an index to one of a number of "hash buckets" in a hash table.

header

An address-related line inside a message, such as `TO:`, `REPLY-TO:`, and `SENDER:`. Headers are a visible part of the message but are not used to route mail.

Header file

A file that contains the header information for a message, including `TO:`, `CC:` , `BCC:` , `FROM:`, `REPLY-TO:`, and `SENDER:` information.

header rewriting

The process of changing the content of message headers (such as the `TO:`, `CC:` and `FROM:` headers) without changing the addressing in the message envelope. Header rewriting does not affect mail routing. It is primarily used to clean up the addresses that readers see in messages, and to hide proprietary address information when necessary.

high availability

Availability of a system to users 24 hours per day, seven days per week, without any significant interruption of service. Achievement of high availability requires the implementation of a variety of hardware and software strategies.

host

A machine in a network. For example, in `venus.software.com`, `venus` is the host.

HTTP (hypertext transfer protocol)

The protocol that both Web clients and Web servers use to communicate and transfer data across the Internet.

IMAP (Internet mail access protocol)

A protocol that allows users to view and manipulate messages directly on the MSS without having to download them. IMAP provides multiple connections with simultaneous access to mailboxes, and allows users to request only portions of messages, such as headers or attachments. IMAP users can also create new folders in their mailboxes and move or copy messages between folders.

IMAP server

The InterMail server that handles requests for message retrieval from mail clients that support the IMAP protocol.

See also *IMAP.*

inbox

One of the default folders for an InterMail mailbox. All new messages go into this folder as the MSS receives them.

incoming mail

Every message, whether addressed to a local user or destined for a remote recipient. After applying the rules for incoming mail, InterMail completes additional checks to determine whether the message's final destination is local or remote.

See also *outgoing mail*.

index

In database design, a list of keys (or keywords), each of which identifies a unique record. Indexes make it faster to find specific records and to sort records by the index field (that is, the field that identifies each record).

InterMail user

A new UNIX user account created before installation on every host on which InterMail will be installed. InterMail files are in this user's directory, and all InterMail processes run in this directory. To execute any administrative commands, or to administer InterMail in any way, the administrator must log in as the InterMail user.

InterMail user group

A UNIX user group created on every InterMail host. The InterMail user is the only member of this InterMail group.

IP address

The network address on a TCP/IP network.

ISD (Integrated Services Directory)

A set of services and databases used as a high-speed, scalable repository of account, administrative, and related information. The ISD consists of the Directory

Cache servers, Directory server, Directory Cache databases, and Directory database.

ISP (Internet service provider)

A company that provides connectivity to the Internet. Typically, an ISP provides user connections for the World Wide Web, FTP, news, and e-mail.

journaled file system

Any system that writes data to a journal. The journal is not lost in the event of a system crash and can therefore be used for recovery.

journaling

The process by which all Message File system transactions are saved in InterMail. A journal records every insertion, change, and deletion. Playing back the journal will re-create a full image of the Message File system. With a system of full backups, journaling provides a mechanism for full recovery of lost messages.

key

See *configuration key, public key.*

killing (of a server)

The most forceful means of shutting down a server. Unlike stopping or draining, killing a server causes its client connections to be terminated immediately.

See also *draining*, *stopping*.

LDAP (lightweight directory access protocol)

An Internet protocol that allows users to access and search a variety of otherwise incompatible directory systems for information such as names, telephone numbers, and addresses.

LDIF (LDAP data interchange format)

A standard way of representing directory data in a text file format, used to import and export data among LDAP directories. An LDIF entry consists of a series of lines of ASCII text, the first line specifying a distinguished name for the entry, and each subsequent line specifying an attribute of the entry.

leaf node

A location in a DIT that is at the end of a branch. A leaf node has no descendants.

local delivery

Delivery of mail to a local mailbox.

local mailbox

A mailbox created and maintained by the InterMail messaging system.

local mail domain

A mail domain over which the InterMail system has complete authority. All users within a local mail domain have accounts in the ISD.

See also *non-authoritative domain*.

local user

A user with an account in the ISD.

logical unit number (LUN)

A single unit made up of one or more disks.

See also *disk array, disk striping*.

login name

The name a user employs to start a POP or IMAP client session and begin the authentication process.

mail blocking

A method of preventing specific users or systems from sending mail to your site.

mailbox

A storage area in the Message Store database for messages that are sent to an end user's account. Typically, each account in the ISD is associated with a mailbox. Each mailbox has folders, which in turn contain the messages that have been received for the end user.

mailbox quota

A limit set on an account to control the size of the mailbox or the messages it contains. Quotas defined for each mailbox are the total size of messages in the mailbox, the maximum size of a single message in the mailbox, and the total number of messages in the mailbox.

mail forwarding

The sending of mail addressed to a particular destination to a different specified address. When a message arrives for an account for which mail forwarding is turned on, InterMail modifies the destination address to the one specified, and then sends the message to the modified location.

mail in delivery

All messages that the system is actively engaged in delivering. Messages that have been explicitly deferred are not considered mail in delivery.

See also *mail in process*.

mail in process

All messages that have not yet been delivered or explicitly rejected. Mail in process includes all mail in delivery, plus all mail that has been explicitly deferred for later delivery, sidelined for review, or held due to an error condition.

mail loop

A condition in which a message moves infinitely between two MTAs due to improper use of the empty address, or due to a lack of MTA hop specification.

mail routing table

A series of (typically) colon-delimited domains that can route mail if the specified destination domain for a message is unavailable.

Manager server

The InterMail server that allows administrators to log on to a single InterMail host and start or stop any of the servers running on that host or on any other InterMail host. The Manager server runs on each InterMail host.

master agent

An SNMP agent that collects information from the subagents in the system for exchange with an SNMP network management system using the SNMP protocol. There is one SNMP master agent per system.

master Configuration database

A database containing the authoritative list of InterMail system settings. The master Configuration database is used by the Configuration server to provide all InterMail servers with current configuration information.

master configuration host

The host machine on which the master Configuration database and Configuration server are located.

message

1. In an e-mail system, an individual piece of mail.
2. In computer systems in general, an information unit that the system sends back to the user or system operator with information about the status of an operation, an error, or other condition.

message aging

A feature that permits deletion of old mail from the Message Store database, even if recipients have not read the mail. Message aging is useful for controlling the size of mailboxes and preventing over-quota conditions.

Message File system

The file system located on the MSS that is responsible for storing the physical contents of a message, including the header (summary of the contents of the message) and the body (the text of the message and any attachments).

message relaying

The process of sending messages from one MTA to a second MTA for any of a number of reasons, including migration.

See also *relay host*.

Message Store database

A database containing state information about messages stored in the Message File system.

MIB (Management Information Base)

The database of information maintained by an SNMP agent that the network management system can query or set. InterMail supports querying only.

migration

The process by which users' mailboxes are moved from a "legacy" mail system (for example, Post.Office) to an InterMail system. The process of migrating mailboxes involves exporting account information from the old system, creating new accounts in InterMail, and then moving mailboxes to InterMail.

MIME (multi-purpose Internet mail extensions)

A standard for multi-part, multimedia electronic mail messages and World Wide Web hypertext documents on the Internet. MIME provides the ability to transfer nontextual data, such as graphics, audio and fax. RFC 1341 defines the MIME standard.

mirroring

The writing of duplicate data to multiple devices (usually two hard disks) in order to protect against loss of data in the event of device failure. There are hardware implementations (sharing a disk controller and cables) and software implementations of this technique, which is a common feature of RAID (redundant array of independent disks) systems.

MSS (Message Store Server)

The InterMail server responsible for hosting mailboxes and storing incoming messages.

MTA (Message Transport Agent)

The InterMail server responsible for delivering messages. The MTA determines whether a message is destined for local or remote delivery, and then performs those tasks required to complete the delivery process.

multi-threading

The ability of an individual server process to perform multiple tasks simultaneously. Multi-threading enhances system performance, resulting in higher message throughput.

mutex

Short for *mutual exclusion lock*. Use of this type of lock excludes all threads other than the lock holder from any access to the locked resource.

MX record (mail exchanger record)

An Internet MX record specifies a *mail exchanger* for a specific domain. The mail exchanger is the host machine whose task it is to deliver or forward mail for this particular domain. When the MTA attempts to deliver mail to a remote domain, it must first find the MX record for the machine that delivers mail for that domain.

non-authoritative domain

A mail domain over which the InterMail system has partial authority. InterMail accepts messages for all users in a non-authoritative mail domain, but only some of those users have accounts in the ISD. Also called *semi-local domain*.

See also *local mail domain*.

object class

A collection of required and optional attributes in a directory that defines a type of data. For example, `person`, `organization`, and `domain` are standard object classes in an LDAP directory.

object identifier (OID)

A string of numbers separated by dots, used to uniquely identify objects in the directory. Each part of an OID represents a node in a hierarchical OID tree. This allows blocks of namespace to be delegated to individual organizations for their own use. For example, the InterMail object `class mailUserPrefs` is the string of numbers assigned to Software.com followed by a string of numbers designated by us: `SWCOM.1.2.2.6`

orphan

1. A child process left when a UNIX parent process creates it and then terminates.
2. A message file that exists in the Message File system without a corresponding referential pointer in the MSS.

See also *widow.*

outgoing mail

Mail whose final destination is a remote mail server.

See also *incoming mail*.

parent process

A process that creates another process. The created process is a child process.

partitioning

The replication of Directory data to two or more Directory Cache servers. The servers typically have mutually exclusive sets of entries, which together constitute all of the data required for the application served by the Directory caches. Dividing accounts by region (for example, east and west) is an example of partitioning.

password

The string that a user enters when starting a POP or IMAP client session, after entering a login name.

path

The specification of a file or directory in a hierarchical file system. The path usually lists the directories top-down, separating the directories by the pathname separator ("/" in UNIX, "\" in DOS).

permission

An access privilege (for example, read and write) associated with a file or directory. Depending on the operating system, each file may have different permissions for different kinds of access and different users or groups of users.

ping

A program that "bounces" a request off another computer over a network to see if the remote computer responds. If the ping comes back, the remote computer is alive.

Since ping works at the IP level, its server side is often implemented entirely within the operating system kernel and is thus probably the lowest-level test of whether a remote host is alive. A ping often produces a response even when higher-level TCP-based services cannot.

POP (post office protocol)

The protocol used by most e-mail clients to retrieve e-mail from a mail server. There are two versions of POP. The first, POP2, became a standard in the mid-1980s and requires SMTP to send messages. The newer version, POP3, does not require SMTP.

POP server

The InterMail server that handles requests for message retrieval from any client that supports the POP3 protocol. It communicates with the Directory Cache server to validate the user’s login name and password, and to obtain the name of the MSS host on which the user’s mailbox resides. The POP server also communicates with the MSS to service requests for message retrieval from the client.

port

In a communications network, an endpoint to a logical connection, with a unique port number. For example, port 110 is typically used for POP traffic.

See also *reserved port*, *well-known port.*

primary SMTP address

The main address for a user's account. Options such as forwarding and SMTP aliases use the information in this primary address as a source of forwarding information.

private key

A key used in public-key cryptography to perform a type of decryption. Private and public keys come in corresponding pairs.

A private key belongs to a single user only. Unlike a public key, which can be available to anyone, a private key remains secret to everyone except its user. It is used to decrypt data that has been encrypted with that same user's public key.

See also *public key.*

process

A single stream of instructions that may in turn spawn other processes.

proxy

A server that acts as a surrogate for another server. The proxy receives messages, then relays them to another server where the end user's account may actually reside. Proxying is used primarily during the migration process (when user accounts are moved from a legacy system to InterMail).

public key

A key used in public-key cryptography to perform a type of encryption. Public and private keys come in corresponding pairs.

User A may send a secure message to User B by obtaining User B's public key. User A then encrypts all or part of the message with User B's public key. After receiving the encrypted message, User B decrypts it with a corresponding private key.

A user's public key can be made available to anyone who wishes to initiate an encrypted transaction with that user.

See also *private key.*

queue directory

The root directory for the Queue server. The `queue` directory contains a series of subdirectories that store messages that the MTA cannot deliver immediately.

Queue server

The InterMail server that is responsible for temporary storage of messages that the MTA cannot process immediately.

quota

See *mailbox quota*.

quota threshold

A percentage of a quota value at which a user is warned about an impending quota violation.

redundancy (server)

A condition in which multiple servers of a particular type exist to perform required tasks in an InterMail system. In such a system, the loss of any one server has a minimal effect on system operations.

relative distinguished name (RDN)

An LDAP term for the most specific component of a distinguished name. A relative distinguished name must be unique among entries of the same parent in the DIT.

relative pathname

The path to a file relative to a given starting point. The starting point is typically the InterMail operational directory, which you can identify by typing:

```
echo $INTERMAIL
```

See also *absolute pathname*.

relay host

A backup host that functions as the MTA. A relay host can be a proxy host used in the migration process or an MTA that takes over for the primary MTA in a system.

relaying

See *message relaying.*

remote delivery

Delivery of a message to another mail server (as opposed to a local mailbox).

remote mail server

A mail server outside the InterMail system.

remote recipient

A recipient who does not have an account in the ISD.

replica set

A set of identical Directory Cache servers that are interchangeable for the purpose of satisfying client requests. If one of Directory Cache server is unavailable, a

client tries to communicate with the others, one at a time. Replica sets thereby function as a failover mechanism.

replication

The process by which directory entries are automatically copied from the Directory server to one or more Directory Cache servers for local storage to provide greater performance, scalability, and reliability.

replication agreement

A filter that specifies which directory entries and which of their attributes are copied and maintained by one Directory Cache server or by a replica set formed by several Directory Cache servers.

replication area

A set of objects and attributes to be replicated to a consumer server.

reserved port

A port that, although not explicitly defined to carry a particular protocol or service, is within the range of well-known port numbers (0–1023). A reserved port is reserved for future use, and users should take care not to assign port numbers arbitrarily to reserved ports.

See also *port*, *well-known port*.

resource file

A file that defines a set of environment variables and user preferences to determine behavior during a session. Examples of resource files are `.cshrc` and `.profile`.

rewrite domain

A domain name that serves as an alias for a local or non-authoritative domain.

RFC (request for comments)

A series of notes about Internet standards and protocols. An RFC number designates each RFC. Once published, an RFC never changes. Any changes that are necessary become a new RFC with a new RFC number. Anyone may submit an RFC, but an RFC gains acceptance as an Internet standard only if it generates enough interest.

RME (remote method execution)

The protocol that InterMail servers use to communicate the results of transactions between themselves.

rollover

In an InterMail system, a means of conveniently archiving large log files to secondary storage without disrupting running applications. In log file rollover, the system closes the old log file, creates a new log file, and continues logging to the new file without interruption.

root directory

The top-level directory on a disk.

routing host

A host machine that performs the functions of a router.

scalability

The ability to change the size or capacity of a system in proportion to the increase in resources used; for example, the ability to handle a larger number of users based on a proportionately greater hardware base.

schema

A description of the Directory database that sets the rules for what can be stored in it, as well as how the Directory server and its clients handle information during search, modify, add, or delete operations. In LDAP-based directories, a schema is composed of object classes and attributes.

sidelining

The automatic placement of incoming mail suspected of being spam to a designated directory for special review and handling. Mail may qualify for sidelining if it originates from certain domains or addresses, if it is addressed to too many recipients, or if its origin is incomplete or missing. Sidelined mail that is found to be legitimate can be moved back into ordinary mail processing.

signature

Information (such as name, telephone number, and address) automatically appended to outgoing e-mail messages.

SMTP (simple mail transfer protocol)

An application-level TCP/IP protocol for e-mail on the Internet. SMTP defines the message format and message transfer agent. Although SMTP is text-based, MIME (Multipurpose Internet Mail Extension) and other encoding methods allow nontext attachments.

SMTP address

See *e-mail address*.

SNMP (simple network management protocol)

A widely used application-level protocol for network management. Network management applications can query management agents in network devices such as hubs, bridges, and routers. The hardware- or software-based management agents can report information about the device stored in the device's MIB (Management Information Base). Although SNMP is a TCP/IP standard protocol, other non-TCP network types, such as Ethernet, also have SNMP implementations.

SNMP server

An InterMail server that communicates with other InterMail servers (all except the Manager and Configuration servers) to gather useful system information and pass it to an SNMP monitoring station, which can be any remote host. Each InterMail system includes a single SNMP server. (You must supply your own monitoring system.)

spool directory

A directory on the MTA for messages that would normally go into the `queue` directory. The MTA normally operates in a stateless mode. However, if the system is configured to handle this operation and the Queue server becomes unavailable, the `spool` directory on the MTA stores all messages that are not immediately deliverable.

SSL (secure sockets layer)

A transport-level security protocol for authentication and data encryption on the Internet. SSL negotiates security between clients and servers.

stateless mode

A condition of independence. A stateless server request, for example, is an independent transaction, unrelated to any previous request. This simplifies the server design, because it does not require the server to allocate storage to deal with conversations in progress, or to free storage if a client fails in mid-transaction.

The MTA operates in a stateless mode. The Queue server handles all storage of mail in process.

statistics file

A file containing information about system performance.

stopping (of a server)

The process of shutting down a server so that it stops immediately, regardless of the presence of client connections, but terminates client connections with meaningful error or status messages. This is the most common method of shutdown.

See also *draining*, *killing*.

sub-agent

A component of a system that can exchange information with the system's SNMP master agent. There can be one or more sub-agents per system.

supplier server

A server that supplies information to be replicated. In InterMail, the supplier server is the Directory server.

tablespace

A logical portion of an Oracle database for allocating storage for table and index data. Databases have one or more tablespaces, each made of one or more data files. Tables and indexes must exist within a tablespace. A tablespace can have many tables and indexes.

TCP/IP (transmission control protocol/internet protocol)

A networking protocol for communicating between heterogeneous networks and hardware and software platforms. Most of the Internet operates using TCP/IP. Practically all platforms offer TCP/IP support.

TCP, a transport-layer protocol, first establishes a connection between two systems. Then the sending system breaks transmitted data into bundles, called *packets*. Each packet carries a distinguishing sequence number that allows the receiving system to reassemble the packets into the original data. If the `checksum` of the result matches the original `checksum`, the receiving system acknowledges proper receipt of the packet. TCP attempts to retransmit lost or damaged packets.

IP is a network-layer protocol, and uses the 32-bit IP address, subnet mask, and default gateway to address and send packets to their proper destinations.

throttling

The process of controlling the pace of mail flow by adjusting the threshold size of mail to be automatically deferred, the threshold size of mail to be automatically rejected, or the threshold size of mail to automatically cause work to stop on deferred mail (in order that current mail can be processed).

trace file

A file containing diagnostic information. Trace files track the flow of messages through the InterMail system and are useful for debugging and measuring system performance.

username

The portion of an e-mail address that precedes the `@` symbol. For example, in the e-mail address `joe.schmoe@software.com`, the *username* is `joe.schmoe`.

See also *e-mail address*.

welcome message

A message to greet new users and inform them of system policies, quotas, and so on. The welcome message is located in a special administrative mailbox, where the site’s message aging policy cannot delete it.

well-known port

A port that is explicitly designated for a particular protocol or service. For example, port 110 is designated for the POP3 protocol. The well-known port numbers are in the range 0–1023.

See also *port*, *reserved port*.

widow

A message reference which appears in the Oracle tables on the MSS but for which there is no corresponding message file in the Message File system.

See also *orphan*.

wildcard account

An administrative account with an address such as `*@domain.com`. This account is designed to receive all mail that does not exactly match some actual e-mail address.

Index

B

C

D

E

F

G

H

I

J

K

L

M

Q

R

S

T

U

V

W

