InterMail® Kx

OPERATIONS GUIDE

Version 4.2

Published November, 1999

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 Kx Operations Guide, Version 4.2

Document number: KO-991110

Table of Contents

Preface

Welcome to InterMail Kx!

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

Intended Audience

This guide assumes that you have a technical background as well as a working knowledge of the Internet and high-end system issues.

Organization

This manual is organized as follows:

- Chapter 1, E-Mail Overview, provides an overview of electronic mail.
- Chapter 2, Introduction to InterMail, provides an overview of the InterMail Kx system.
- Chapter 3, 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 4, InterMail Data Models, discusses the data objects and data models used by the InterMail system to perform its mail and mail-related operations.
- Chapter 5, Site Management, describes the hierarchy of organizations and organization administrators, as well as the tasks you must complete to set up your site and organizations.
- Chapter 6, Account Management, tells you how to create and manage accounts using the InterManager interface, command-line utilities, and APIs.
- Chapter 7, System Configuration, tells you how to view and update the Configuration database in order to make changes to your InterMail system configuration.
- Chapter 8, Security, discusses InterMail security features. It describes 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 9, Mail Routing, describes the various InterMail mail routing options, designed to help control the direction of mail flow.
- Chapter 10, Managing Mail in Process, discusses reasons for the temporary storage of mail, the location and organization of temporarily stored mail, the handling of deferred mail, and mail queuing options.
- Chapter 11, Managing Mail Storage, discusses the persistent storage of mail, as well as the creation and deletion of mailboxes.
- Chapter 12, System Monitoring and Maintenance, describes monitoring and maintenance tasks as well as monitoring tools you can use to analyze and maintain an optimally performing InterMail system.
- Chapter 13, Backup and Recovery, familiarizes you with principles behind and general guidelines for system backup and recovery for you to apply to your individual InterMail environment.
- Chapter 14, LDAP Directory Synchronization, describes how to synchronize LDAP directory data between an InterMail directory and a foreign directory in a master and slave configuration.

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}`

Convention	Description	Example
`...` (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 Kx Reference Guide*, which contains reference information about InterMail's servers and databases, configuration keys, directory management utilities, administrative utilities, APIs, and event messages.
- *InterMail Kx Installation Guide*, which provides instructions for installing InterMail.
- *InterMail Kx 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

E-Mail Overview

This chapter presents a general overview of the basic concepts that apply to most e-mail systems. If you are new to e-mail administration, this information will be useful; if you already know what e-mail is and what kinds of programs constitute a mail system, you may want to skip ahead to Chapter 2.

E-mail provides a way for two or more people to exchange messages. Just as the postal service is used to send postcards, letters, and magazines, e-mail is used to send various kinds of messages. An electronic message can range from a simple memo or letter to a complex multimedia presentation.

E-mail offers a number of advantages over traditional paper-based communication. It is faster and cheaper, and it can be disseminated to a large number of people simultaneously. Responses are fast too, making e-mail almost interactive.

This chapter discusses how e-mail works, what constitutes an e-mail message, and the technology behind e-mail. It covers:

- Mail clients
- Mail servers
- Elements of a message
- The Domain Name System
- Misuses of e-mail

Mail Clients

Mail clients are programs that help users carry out tasks such as:

- Creating and submitting messages for delivery
- Checking for new incoming messages
- Reading received messages
- Organizing saved messages

Figure 1 shows a variety of mail clients in a network.

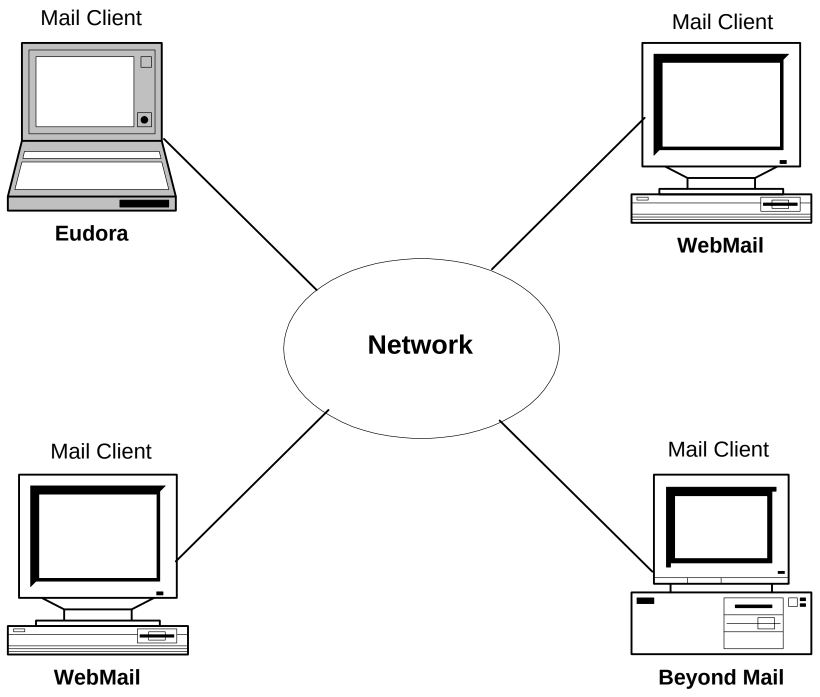


Figure 1 Mail clients

Creating and Sending Messages

Mail clients with a graphical user interface frequently provide message templates that you can use when creating new e-mail messages. These templates make it easy to enter the recipient’s address and text of the message. With most mail clients, it is also possible to include textual and multimedia information as message attachments.

Once a message is created, the mail client hands it off to a program called a *mail server* for delivery to its intended recipients. Mail servers are described in the section “Mail Servers” on page 3.

Retrieving Messages

Mail clients can receive messages using any of the following protocols:

- **Post Office Protocol** (POP). When you check for new mail on a client using POP, the client checks with the mail server. If the mail server has messages for you, it sends them to the mail client. One advantage of POP delivery is that it does not require you to have a system account on the same computer where the mail server is running.
- **Internet Message Access Protocol** (IMAP). If your client uses IMAP, messages do not need to be downloaded to the local file system in order for you to read them. IMAP allows you to view and manipulate messages directly on the mail server. It provides multiple, simultaneous access to mailboxes, allowing requests

to portions of messages, such as headers or attachments. IMAP also allows you to create new folders in your mailbox and move or copy messages between folders on the server.

- **HyperText Transfer Protocol** (HTTP). With a Web-based client using HTTP, you can send and receive e-mail messages over the Internet. Using HTTP, a client opens a connection to a server and then sends a request using a specific format. The server then responds and closes the connection.

Displaying Incoming Messages

Mail clients typically list incoming e-mail messages in a folder such as Inbox. This message list shows:

- Whom each message is from
- When it was sent
- What its subject is

You can save messages and look at them later. Many mail clients also allow you to organize messages into a set of folders or mailboxes.

Mail Servers

A mail server is designed to deliver messages across networks. Mail servers generally interact with other programs – primarily mail clients and other mail servers. The most common task of mail servers is accepting messages from and delivering messages to mail clients, as shown in Figure 2.

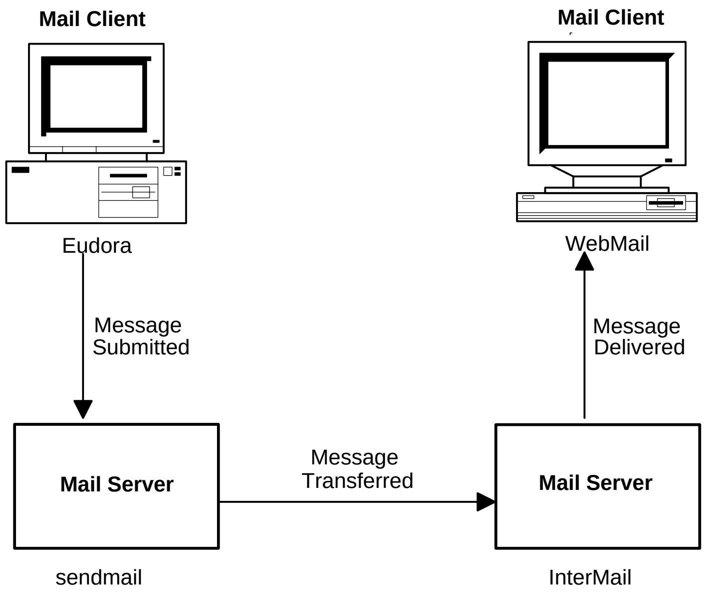


Figure 2 Mail transport

Mail server functions include:

- Receiving mail
- Sorting mail
- Forwarding mail
- Storing mail
- Delivering mail

The function of a mail server is analogous to that of postal workers who sort through mail before it is delivered.

A mail server uses a database to store information about your e-mail account, such as your e-mail address, password, delivery instructions, and so on. Mail servers typically run 24 hours a day for delivery and receipt of mail. However, it is typical for a server to be brought down for a regular maintenance window.

When mail is placed in a postal mailbox, its next stop is a post office. There it is sorted and a forwarding decision is made. The same method is used in e-mail delivery. As an e-mail message travels from the sender to the recipient by way of one or more mail servers, decisions are made as to how the message must be routed, depending on the addressing information provided on the message envelope (see “Electronic Envelopes” on page 5).

All mail servers wait for other e-mail programs to contact them and give them messages. Each e-mail transaction is a client-server transaction, a kind of call-and-response conversation in which one computer asks for something and another provides it. In this case the server is the mail server that is accepting the message, while the client is the mail client or another mail server that is transmitting the message.

Elements of a Message

When receiving a large number of messages, you would like to know several things about the message without opening it, such as:

- Who sent the message
- When the message was sent
- What the message is about

With postal mail this problem of message organization is solved with various conventions; for example, a letter always begins with an indication of whom it is for, when it was sent, and where it was sent from. With e-mail, this problem is solved with headers, which include information such as sender, recipient, and subject, thereby allowing the recipient to sort and prioritize mail before examining the body of a message.

With postal mail, a letter is placed in an envelope that contains the information required for delivery as well as a return address in case it cannot be delivered.

Similarly, e-mail programs use an electronic envelope, which is marked with a destination and return address and which contains the electronic message (header plus body).

Electronic Envelopes

When delivering a message from one person to another, an e-mail program needs to know only two things:

- Whom the message is going to
- Whom it is from, in case it needs to be returned

This information is used to create an electronic envelope. The envelope is labeled with To and From addresses and contains the message (headers and body) within. Only mail clients and servers use electronic envelopes.

Message Headers

In a business memo, key pieces of information are laid out in a series of headers at the beginning of the message. Header information enables mail services to deliver memos efficiently and gives memo recipients an initial idea of the message content.

E-mail message header information (Figure 3) also provides an indication about whom a message is for, what the message is about, and who sent the message and when. As a recipient, you can also use header information to sort messages according to sender, subject, date, and so on.

```
Date: Wed, 17 Feb 1999 10:19:32 –0800
To: "John Doe" john.doe@software.com
From: Jane.Smith@software.com (Jane Smith)
Subject: Planning meeting
Cc: susie.smith@software.com
---------------------------------------------------------
```

Figure 3 Message headers

There are standard protocols for the use and format of headers that allow users with different e-mail clients to send messages to each other.

The Message Body

In general, the body makes up the bulk of an e-mail message. You can send formatted text, graphics, sound, and video clips (multimedia) in the body of a message. If you have a Multipurpose Internet Mail Extension (MIME)-enabled client, you can also send and receive multimedia files.

The Domain Name System (DNS)

A mail server can function as a local post office for an entire network or a portion of a network. When used in this manner, the mail server has a list of local recipients for whom it receives messages. This list translates, from the mail server's perspective, into a list of addresses that includes everybody in that mail server's electronic neighborhood (often called a domain). A mail server accepts a message when it recognizes the address as belonging to its domain.

Addressing protocols are the key to enabling users and computers to contact each other across networks. This section describes the most commonly used addressing protocol, the domain name system (DNS).

With the DNS each computer has a hierarchical name, such as `paris.software.com`.

In this example, the parts of the address are:

`paris`	The name given to the computer. This name must be unique within the organization.
`software`	The name of the organization that the computer is in. This name must be unique within the domain (for example, there is also a `software.org`, but there can be only one `software.com`).
`com`	An abbreviation indicating the type of organization (or sometimes the country) in which the computer is located. The abbreviation `com` indicates a commercial organization.

Note: Other common organization codes include “gov” for government, “edu” for education, “mil” for military, “net” for network resources, and “org” for other organizations. Country codes are generally two digits: “ca” for Canada, “us” for the United States and so on.

A message that is addressed to user `Jane` who uses the computer `paris` could be addressed to:

```
Jane@paris.software.com
```

A large organization may decide to further divide its network by departments such as Sales or Support. With such a division, the address for `paris` could be:

```
paris.sales.software.com
```

In this case, Jane's address would become:

```
Jane@paris.sales.software.com
```

Often, specific computer names are not even included in the e-mail addresses that people use. For example:

```
Jane@Software.com
```

The DNS directory service allows the transformation of the above e-mail address into the address of a specific computer.

Misuses of E-Mail

E-mail is not without its problems. Although these issues are rapidly changing, anyone who is going to administer an Internet e-mail system should be aware of the following:

- Unsolicited commercial E-mail
- Mail relay
- Denial-of-service attacks

Chapter 8 in this manual discusses InterMail features that help deal with these problems.

Unsolicited Commercial E-Mail

The most common misuse of e-mail systems is junk mail, including commercial advertising and other unsolicited material. On the Internet, sending out junk mail is popularly known as *spamming*. Individual pieces of junk mail are called spam or unsolicited commercial e-mail (UCE).

Mail Relaying

Mail relaying happens whenever messages are given to one mail server but are destined for some other mail server. You use mail relaying every time you send a message to someone whose e-mail account is not stored on the same mail server as yours.

In principle there is nothing wrong with mail relaying; it is one of the primary functions of the mail server. However, this function has been misused by distributors of junk e-mail, who often use mail servers to relay large volumes of junk e-mail to users throughout the Internet. This large message volume can disrupt legitimate mail operations.

Denial-of-Service Attacks

A particularly destructive misuse of e-mail systems is denial-of-service attacks. Unlike spamming and abusive relaying, which create problems for mail servers as a side-effect of the distribution of junk e-mail, denial-of-service attacks are created solely for the purpose of disrupting or stopping e-mail activity.

The most common type of denial-of-service attack opens many network connections to a mail server and maintains these connections, thereby disrupting normal activity on the server.

2

Introduction to InterMail

The InterMail product is a high-performance native Internet messaging system. This chapter offers an overview of the InterMail system, including:

- Features and benefits
- Servers
- Data storage
- Model installation

Product Features and Benefits

InterMail offers flexibility, reliability, and ease of use through many features and benefits:

- Multiple mail retrieval options—InterMail supports three popular mail retrieval methods:
  - POP3
  - IMAP4
  - Web-based

  Service providers can allow individual users or groups of users access to any combination of these mail retrieval methods. The web-based application enables users to retrieve mail from any location using a standard Web browser.
- Advanced security features—InterMail provides a variety of state-of-the-art security features:
  - Password protection
  - Mail blocking
  - Message filtering
  - Relay prevention

  - Secure socket layer (SSL) authentication

- Classes of service—InterMail creates a variety of new revenue opportunities by enabling you to sell and manage accounts with different feature sets and price points. The class-of-service feature supports the development of highly differentiated business-specific packages.
- Outsourced administration—InterMail allows you to delegate account administration from the mail system administrator to one or more organizations, each with its own administrator. This allows corporate customers to manage, allocate, and provision accounts for their own users. The resulting independence is attractive to corporate customers, and means less administrative work for service providers.
- Web-based account management—InterMail provides an easy-to-use account management tool, featuring Web forms that enable you to create user accounts and edit account information from an intuitive graphical user interface.
- User-configurable account data—InterMail allows end users to configure a variety of accounts settings. For example, end users can:
  - Change passwords
  - Define aliases
  - Set forwarding addresses
  - Create automatic responses to incoming mail
- Transaction-based communications—Communication between servers is transaction-based; an operation is not considered complete until the initiating server receives confirmation from the server to which it communicated. For example, when the Message Transport Agent (MTA) passes a message to the Message Store Server (MSS), the operation is not considered complete until the MSS signals the MTA that it has received the message. This feature greatly reduces the risk of message loss.
- Native Internet standards support—InterMail supports Internet messaging protocols without the use of gateways or connectors that require conversion to proprietary protocols or message formats.
- Multi-threading—Most InterMail servers are multi-threaded. This greatly speeds up message delivery on the system.

InterMail Servers

The InterMail messaging system includes nine servers. Each server is dedicated to a specific function; however, it can be useful to think about them in the following four categories:

- Message delivery
- Message storage
- Message retrieval
- Management and administration

The rest of this section describes server functions and relationships in greater detail.

Message Delivery

The MTA handles the receipt of all incoming messages and determines whether the recipients are local users or belong to a domain outside the InterMail system. When a recipient is a local user, the MTA obtains account information from the Integrated Services Directory (ISD), then delivers the message to the MSS, which in turn delivers it to the appropriate mailbox. When the recipient is in an outside domain, the MTA sends the message to the appropriate remote location.

The MTA is also responsible for:

- Enforcing security
- Forwarding mail
- Managing delivery process errors

Message Storage

The MSS is responsible for persistent storage of messages in a user’s mailbox. It takes delivery of messages from the MTA and fills requests for message retrieval from the POP, IMAP and Web servers.

The MTA also plays a role in message storage, since it temporarily stores any messages that it cannot deliver immediately.

Message Retrieval

The POP server, IMAP server, and Web server handle message retrieval by the end user. First, they communicate with the ISD to validate the user’s login name and password, and to obtain information about the location of the user’s mailbox and class of service. Then, they communicate with the MSS to retrieve the messages.

Management and Administration

The following servers are involved in the management and administration of the InterMail system:

- The ISD is the definitive source of InterMail account information. It supplies this information to the MTA when the MTA needs to deliver a message to a local user. The ISD also supplies account information to the POP, IMAP, and Web servers when users log on to download their messages.
- The Configuration server manages the Configuration database, which contains settings that control the behavior of all the InterMail servers.
- The SNMP (Simple Network Management Protocol) server communicates with the other InterMail servers and gathers system information such as present server status, number of connections since the server started, number of messages stored in the MSS, and so on.

Note: You must supply your own monitoring system.

- The Manager server is responsible for issuing startup and shutdown commands to the other InterMail servers.
- The Web server, in addition to it role in message retrieval, supports the graphical user interface that allows administrators to enter and change account information.

InterMail Data Storage

The following components of the InterMail system are involved in data storage:

- The Directory database (ISD) stores InterMail account information. The MTA uses this information when it needs to deliver a message to a local user, and the POP, IMAP, and Web servers use this information when end users log on to download their messages.
- The Message Store database holds information about messages, including the users for whom they are being held, the current state of each message (read, unread, deleted, and so on), and the location of each message’s associated message file.
- The Message File system stores actual message data. Messages are static information that is not subject to change. Each message exists as a single file that includes all headers, body text, and attachments.
- The Configuration database stores the configuration information that controls the behavior of each InterMail server. Every InterMail server reads the information in this database when it starts up.

Sample InterMail Configuration

Throughout this manual are discussions of various InterMail features, along with step-by-step instructions on their use.

For ease of understanding, the sample configuration described in this section will be used as the basis of all later examples.

- Host machine—InterMail host machines in this configuration are named for cities such as `paris`, `london`, `rome`, and so on.
- Classes of service—When InterMail is installed, the `default` class of service is created automatically. The sample InterMail configuration uses three additional classes of service:
  - Economy
  - Standard
  - Premium
- Organizational structure—InterMail offers service providers the ability sell outsourced e-mail service to corporate customers and other organizations. To illustrate how you can partition an InterMail system in order to provide delegated administration for a variety of customers, the sample configuration uses the following hypothetical institutions. With the exception of Software.com, Inc., all are fictitious.
  - A service provider, YourISP, Inc.
  - Three corporate customers of YourISP, Inc.: Software.com, Inc., Megacorp International, and Minorcorp Ltd.

Megacorp International has two divisions, Sales and Manufacturing. The Sales division has two subdivisions, Western Division and Eastern Division. The Manufacturing division has two plants, Plant 1 and Plant 2.

Figure 4 shows the organizational relationship between YourISP, Inc., and its customers.

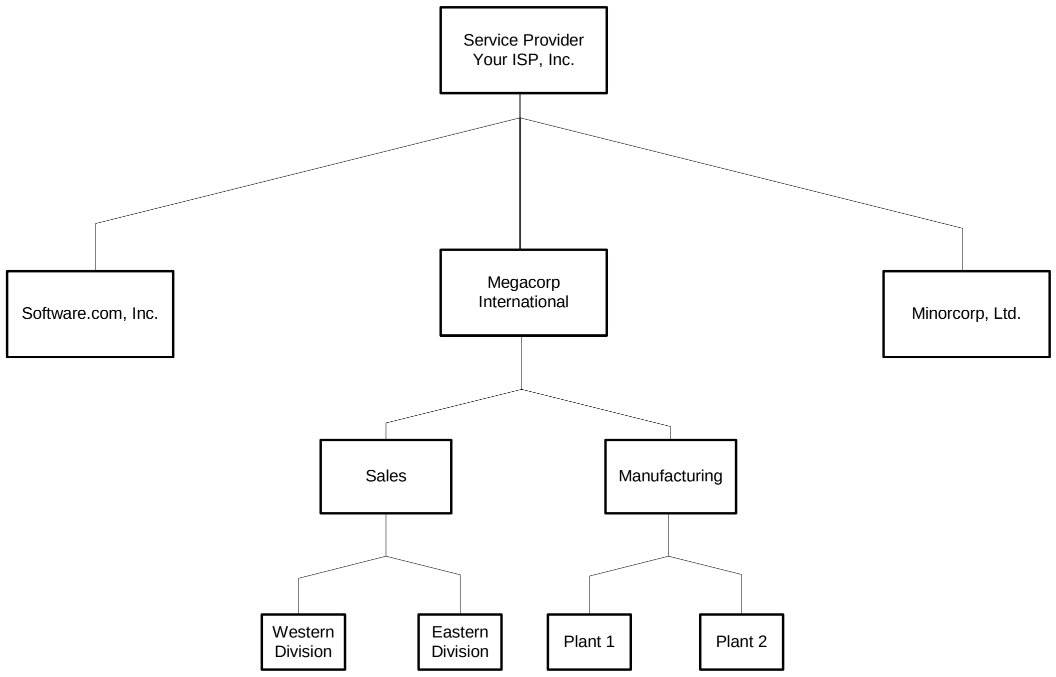


Figure 4   Sample InterMail configuration

3

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
- Complete preproduction planning tasks

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. These examples assume that you have already 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 Integrated Services Directory (ISD) and the configuration of the system are already running.

To perform an operation on a server, you need to 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`
Manager server	`immgrserv`
Configuration server	`imconfserv`
Web server	`httpd`
Directory server	`imdirserv`
SNMP server	`snmpd`

There are two commands that you can use to start and stop InterMail servers: `imctrl` and `imservctrl`.

- `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 useful in 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 stop 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`. Otherwise, the server is not configured to run. You can read this key either by running `imconfedit` and searching for this variable 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 current machine. To start a particular server and no others, you need to 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 Kx 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 the use of `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** 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** 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 exits. For example, when a POP server drains, it does not accept any new connections from POP3 clients. It 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 the 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 shutdown is typically not practical for the IMAP server.

- **Killing** a server causes the process to exit at once. This method is equivalent to the UNIX `kill -9` command. It causes the process to exit immediately, with no error or notification messages to the connected clients. It is recommended that you not use this method of shutdown unless absolutely necessary.
- **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 executes, 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 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 backup and recovery policies.
- Defining the structure of your site. These tasks are necessary to add account and domain information to the system.
- InterMail configuration tasks. Completion of these tasks results in a custom configuration specifically suited to the needs of your organization.

Integration with Existing Systems

When integrating InterMail into your existing operations 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 site-specific, they are described here only in general terms.

Establishing a Connection to Your Provisioning System

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

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.

See Chapter 12 for an overview of InterMail logging and Chapter 9 of the *InterMail Kx Reference Guide* for a complete listing of InterMail log events.

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 MTA, and the total number of messages stored on the 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 12 of 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; among other things, you must decide:

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

Your decisions 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 enforcing it.

Chapter 13 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 those 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 Kx Reference Guide.*

Defining Site Structure

When an InterMail site is configured, you are ready to model your site and then create organizations, domains, classes of service, and an initial set of accounts. This section discusses key considerations in modeling your site followed by the setup of:

- Organizations
- Domains
- Accounts
- Classes of service

Key Considerations in Modeling Your Site

Because requirements vary from site to site, it is impossible to document a recommended model for InterManager implementation. Instead, here are guidelines to assist you in determining the appropriate model for your site. For a detailed discussion of site management tasks, see Chapter 5.

Issues to consider include:

- Identification of the host organization. The first organization created is designated as the site. Administrators linked to this organization have access to all account and class-of-service operations. The identity of the site should reflect the identity of the service provider.
- Identification of clients requiring support. Separate organizations with unique names should be established for each client. A client would typically be one of the ISP's large customers, or else a group of consumers whose accounts will be administered together. By segregating client data into distinct organizations, you can limit client access to the appropriate data set.
- Support for existing client structures. The service provider must establish the top level of the organizational model (the organizations). The clients themselves typically define lower levels. However, if you wish to populate the model with initial user data according to a more detailed model, those additional levels of detail should be defined.
- Desired handling of unaffiliated users. The host organization (the service provider) may provide mail service for users who are not affiliated with another client company. By default, such users are grouped into a single *consumer*

organizational unit. If you intend to support a large number of consumer users, you may wish to subdivide those users into more manageable groups by establishing additional organizational units within the consumer unit.

Defining the appropriate model for delegated administration is a critical first step in implementing InterManager. Once that model has been established, the following additional issues should be addressed:

- Creation of organization administrator accounts—To delegate responsibility for account management back to the client companies, the service provider must create organization administrator accounts for each of the client companies or organizations.
- Naming conventions—For ease of reference and to guarantee uniqueness, conventions should be established for user addresses and login strings. In addition, recommended naming conventions for organizational units should be created and distributed to the administrators of all client organizations.

Setting Up Organizations

InterMail supports delegated administration through a flexible hierarchy of organizations, organizational units, and users. The specific implementation of the hierarchy depends on the unique requirements of each site.

The top level in the hierarchy is the organization. Typically, each organization represents a single company. The first organization established represents the site itself (the service provider running the InterManager application).

Note: There is no requirement that an organization represent a company. An organization is simply a logical container for a group of accounts that should be administered as a distinct entity.

Organizations can contain organizational units, which are smaller groups of users that have common characteristics. Organizational units can be nested within other organizational units. For example, a large company may be set up as a single organization containing separate organizational units for each division, and each of these organizational units may in turn contain another organizational unit for each department within the division.

Users must be associated with an organization or an organizational unit, but they can be linked with a container at any level in the hierarchy. Users can be moved between levels in a given organization; however, they cannot be simply moved from one organization to another.

For a detailed description of site hierarchy and site management, see Chapter 5.

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 system 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.

External recognition is gained through establishment of MX records in the domain name system (DNS). Internal recognition is achieved through the addition of domain records to the mail server.

For a discussion of domains, see Chapter 4, InterMail Account Structure.

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 four methods by which a provisioning system can interface with InterMail: the InterManager Web interface, a C API, a Perl API, or a command-line interface.

Migrating Existing Accounts

If you have a database of account information from a legacy mail system, one of your most important pre-production tasks is to “migrate” existing account information to the ISD. The new InterMail system can 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 Kx Migration Guide.*

Creating Class-of-Service Options

A class of service comprises 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, mailbox quotas, and end user access to the InterManager Web interface. 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 and a different rate for e-mail accounts offering a larger set of services.

The number of classes of service defined is at the discretion of the service provider. To determine an appropriate number of classes of service to offer and the characteristics to be associated with each, you should review the requirements of all expected clients.

Note: For a complete description of attributes that can be associated with a class of service, see Chapter 4, InterMail Account Structure.

Modifying Configuration Options

When you installed InterMail, each available configuration option received a value. You are probably unaware of the settings of most values, since very few of them required direct selection on your part. This method simplifies the installation process, but does not necessarily result in the ideal configuration for your site.

This section identifies 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 routing, storage, and security policies specific to your site.

For instructions on editing configuration keys, see Chapter 3, System Configuration. For details on specific configuration keys, see the *InterMail Kx Reference Guide.*

Determining a Security Policy

InterMail provides a number of security-related features you should thoroughly understand before going into production. Chapter 8 of this guide explains each security option in detail and discusses the potential impact of different security settings on your mail service.

Once you are 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 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 8 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 you are transferring accounts from a legacy mail system to InterMail. For more information on mail proxying, see the *InterMail Kx 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 preproduction. For a full discussion of routing rules, see Chapter 9, Mail Routing.

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 that can be accepted.

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 preproduction issue. In contrast, the system-wide settings are important considerations before production, since the choices you make will affect your entire system right from the start. For additional information about setting message quotas, see Chapter 4, InterMail Account Structure.

Setting a 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 decide how long deferred mail can remain in the queue before the system considers it undeliverable and returns 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; however, 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 10, Managing Mail in Process.

Setting the Welcome Message

If specified, a welcome message can arrive automatically in every new mailbox. A well-written welcome message can create a positive first impression and also convey important information, such as contact numbers and e-mail addresses for technical support, information on quota policies, and the URL for your organization’s Web site.

You use the `immsinit` command to create an administrative mailbox containing the welcome message. For details on the `immsinit` command, see the *InterMail Kx Reference Guide*.

Setting Up cron Jobs

Certain InterMail processes should run periodically to perform routine system monitoring or maintenance tasks. Most recommended `cron` jobs begin automatically at the 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 archiving mail-related log files.

The `ServerAdmin` directive in the `httpd.conf` file specifies an actual account for a system administrator or network operator who will receive administrative mail for the `httpd` server. The value for this directive is set by default to the `admin` user specified during installation. You can modify this value if you need to need to change the administrator's e-mail address for the `httpd` server.

4

InterMail Account Structure

This chapter describes the structure of InterMail accounts and how accounts relate to domains and classes of service.

It covers:

- Accounts
- Domains
- Classes of service
- Mailbox quotas

For information on managing accounts and classes of service, see Chapter 5, Site Management, and Chapter 6, Account Management.

About Accounts

Account information, which is stored in the Integrated Services Directory (ISD), has a significant effect on mail handling, given that entries in the account record control message delivery, message storage, and message retrieval.

There are eight types of information related to an InterMail account:

- General account data
- E-mail addresses
- Delivery information
- Auto-reply options
- Mailbox quotas and related options
- Mail system access
- Web interface access
- Class-of-service identification

General Account Data

The basic information for each InterMail account includes the following:

- **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 one e-mail address, usernames must be unique within a domain.

- **Domain**—Each 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**—An e-mail client uses the account password when connecting to the POP or IMAP server to retrieve mail from the account's mailbox. Finally, the password is required for mail delivery when secure or authenticated SMTP transmission is requested.

- **Account type**—There are two types of InterMail accounts: *standard* and *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. However, the type of the account can indicate certain information in conjunction with a billing or provisioning system. For example, you may create a rule in your billing system to ignore all administrative accounts when generating usage and billing information.

  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.

- **Account 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 then to deliver them normally when the status of the account returns to Active. Client 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.
- **Locked status** is identical to Suspended status. 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 InterMail hosts as their SMTP, POP, or IMAP servers while their accounts remain on a legacy mail system.

Note: 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.

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*.

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 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
```

Each account includes an optional limit on the maximum number of SMTP aliases. Although users may add their own alias addresses, administrators 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 also has an associated *mailbox*.

The most common method of mail delivery is local delivery. When local delivery is enabled, messages are stored in the account's local mailbox, which is simply a storage area for mail sent to the account. Messages can be retrieved from the mailbox using any of InterMail's standard retrieval mechanisms, including the POP, IMAP, and Web servers. Mailboxes are discussed in greater detail in Chapter 11.

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.

Note: Accounts that use forwarding delivery only do not require or make use of a mailbox.

Each account includes an optional limit on the maximum number of forwarding addresses. Although users may add their own forwarding addresses, administrators control this limit.

Each account also includes an optional control on mail filtering. If this option is enabled, mail addressed to this account is subject to system-wide filtering rules. For more information on mail filtering, see Chapter 8, Security.

Auto-Reply Options

InterMail accounts offer end users the option of automatically responding 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 automatically 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 period (by default, one week). This means that if a person sends ten messages within a week to an account that is using the Vacation mode of auto-reply, that person will receive only one copy of the account’s auto-reply message.

Mailbox Quotas and Related Options

Each InterMail account that uses the local delivery method has 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 messages that can be delivered to the account.
- **Maximum mailbox size** limits the storage size of the account’s mailbox.
- **Maximum number of messages** limits the number of messages that can be in the account’s mailbox at any time.

If delivery of mail to an account’s mailbox would violate 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. The InterMail system can be configured 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 account’s mailbox quotas.
- A **quota threshold warning** advises the recipient that the mailbox is nearing one or more of its established quotas.

Mail System Access Options

The following options define user access to InterMail services:

- **Login name**—Each account that uses local delivery has an associated login name. When retrieving messages with the POP, IMAP, or Web server, 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**—When this option 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**—When this option 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**—This option 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.
- **Authenticated SMTP**—This option specifies that the user must provide authentication information when sending mail using SMTP. When it is enabled and the user submits mail to an MTA, the system accepts the 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**—This option controls the user's ability to send e-mail using the standard (non-secure) SMTP port.
- **SMTP with SSL**—This option controls the user's ability to send e-mail using the SSL (secure) SMTP port.

Web Interface Access Options

Each account includes optional characteristics that define and control user access to the following available Web interfaces in InterMail:

- **SelfCare** provides end user control over a limited set of account information.
- **InterManager** enables administration of grouped accounts by enabling separate administrators for each group of accounts and the ability to delegate administrative privileges from an Internet Service Provider (ISP) to a company.

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.

Note: For a discussion of classes of service, see "About Domains" on page 36.

Special-Purpose Accounts

An InterMail system can include two types of special-purpose accounts: wildcard accounts and reserved accounts.

Wildcard Accounts

Each local domain can have an optional wildcard account, which is simply an account within a local domain that receives all mail sent to non-existent addresses in the domain. This feature allows you to collect in a single account all mail sent to a particular domain when the destination address 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`, 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 one of 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 at time of 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:

- The **imail** account can receive output from `cron` jobs.
- The **postmaster** 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.
- The **mailer-daemon** 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.
- The **root** account is similar to the imail account. It may also receive `cron` job and other system messages generated by InterMail.

Note: Reserved accounts such as imail and root are InterMail accounts and should not be confused with UNIX accounts.

About Domains

Domains are a general form of Internet addressing. All SMTP-compliant e-mail addresses include a domain to the right of the `@` sign. To be known to the InterMail system, a domain must be defined in the ISD. Every InterMail account and address in the system must be part of a known domain.

Types of Domains

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 will not be 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 and to 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 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) or when a domain is an outsourced domain, such as a domain serviced for a remote customer.

Rewrite Domains

A rewrite domain is very different from a local or non-authoritative domain. A rewrite domain simply defines a rule for rewriting of the domain portion of the recipient address on incoming mail; it is always 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 9, Mail Routing.

Deleted Domains

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

Relationship of Domains and Accounts

Domains play the following roles in InterMail accounts:

- Domain data in the ISD defines the mail domains (that is, Internet domains) known to InterMail.
- All accounts and addresses for every InterMail user are associated with a known domain.
- When mail arrives for an address in a known domain, InterMail searches its directory data for the associated account, reviews the delivery options associated with that account, and delivers the mail accordingly.
- Additional information in the account record (password information and login name) allows retrieval of the message by the appropriate user.

Figure 5 shows the relationship among domains, addresses, accounts, and mailboxes.

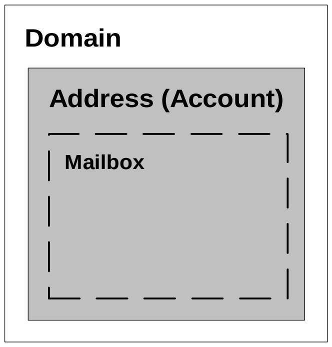


Figure 5 Mail handling

The dashed rectangle represents the logical relationship between the account in the ISD and its associated mailbox (if any). Mailboxes and messages are managed separately by the InterMail MSS, but the location of the mailbox is stored with the account information in the ISD.

About Classes of Service

Classes of service allow you to establish a variety of standard service offerings. The characteristics of a class of service can be inherited by all accounts assigned to that class of service. Except for the general account data described under “Relationship of Domains and Accounts” on page 37, administrators can configure most account attributes at the class-of-service level.

Figure 6 depicts one possible implementation of classes of service. In this example, there are three classes of service: economy, standard, and deluxe. Each class of service corresponds to a different set of features and user privileges. Pricing policies vary based on the feature set selected.

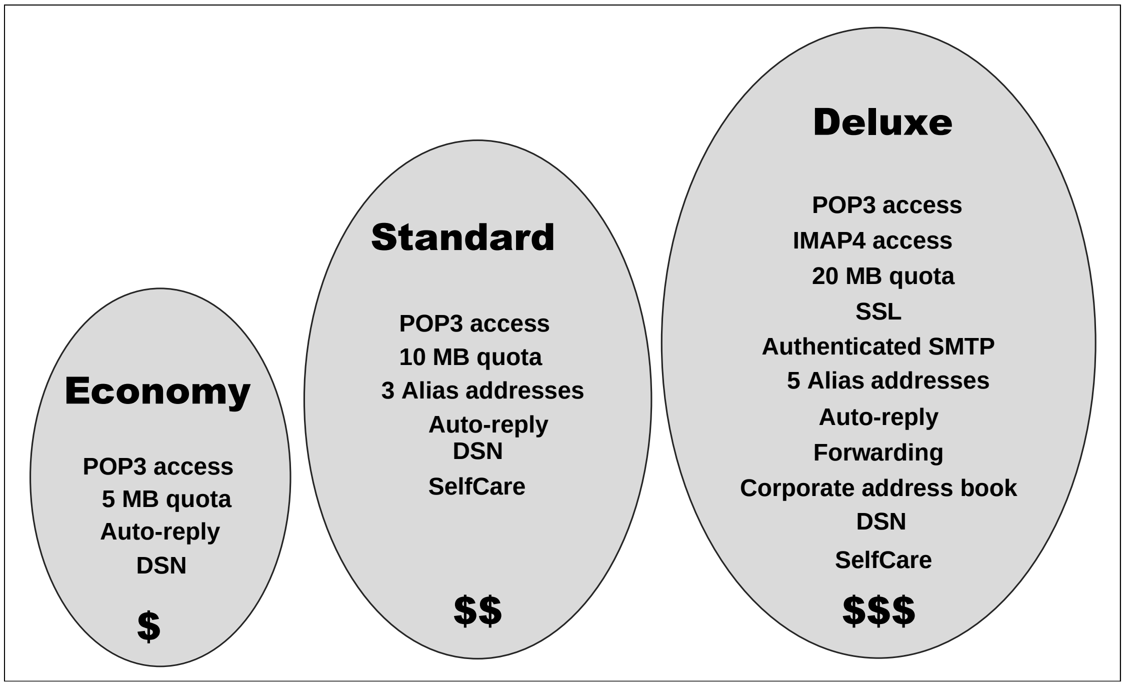


Figure 6 Sample classes of service

Classes of service define standard feature sets that can be easily associated with multiple accounts. The number of feature sets defined is up to the discretion of the service provider.

For example, you may want to have three classes of service:

- **Economy** for users who read mail infrequently and do not require any forwarding.
- **Standard** for users who frequently read mail; require the ability to forward mail, send auto-reply messages, and access InterMail client Web applications; and who require a higher mailbox quota than Economy users.
- **Deluxe** for users who require all the features of the Standard user, as well as IMAP access and the ability to filter messages that appear to be junk mail.

To determine the appropriate number of classes of service for your site, and the characteristics associated with each, you should review the requirements of all the organizations and end users you plan to create.

Class-of-Service Attributes

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

There are two types of class-of-service attributes:

- **Preferences** define the values for account options. For example, preferences define whether a feature is in use (as in the case of mail forwarding) and set account-related limits (such as mailbox quotas).
- **Permissions** define whether end users may set preferences for their accounts. For example, the option to enable or disable user access to mail delivery options (such as forwarding delivery) in the SelfCare Web interface is a permission.

Although each permission attribute has an associated preference attribute, the opposite is not true; preferences such as mailbox quotas and Web interface access, which an end user cannot set, have no explicitly defined permissions.

Note: You can define additional preference and permission attributes, allowing you to create new user services that are outside the scope of InterMail.

Attribute Precedence Rules

It is possible to define class-of-service attributes at both the account level and the class-of-service level. The administrator always sets attributes for a class of service, whereas the user typically sets individual account attributes through the SelfCare Web interface. Although an administrator can set attributes at the account level, this typically occurs only if an end user requires special access or privileges (for example, a mailbox quota larger than what the account's class of service defines).

The InterMail system uses precedence rules to determine whether values for a particular attribute are to apply at the class-of-service or account level. For example, if the mailbox quota of an account is 3 MB, but the mailbox quota for the associated class of service is 1 MB, a precedence rule for the mailbox quota attribute determines which of the quotas applies.

The four precedence rules are:

- **Class-of-service value instead of account value**—This rule is typically for attributes that the end user cannot modify.
- **Account value instead of class-of-service value—**This rule is typically for preferences that are set at the discretion of the end user.
- **Lesser of the account and class of service values—**This rule is for preferences that the user can set only to a lower value. It establishes the class-of-service entry as the maximum allowable value.
- **Greater of the account and class of service values**—This rule is for preferences that the user can set to a higher value. It establishes the class-of-service entry as the minimum value.

These rules depend on whether a value exists for the attribute at the class-of-service level. The following table defines the interaction between attribute values and the resultant values that apply to accounts.

For this precedence rule:	If attribute values are defined at this level:	This is the value applied to the account:
C (class-of-service value used)	For neither class of service nor account	Default value
	For class of service only	Class-of-service value
	For account only	Default value
	For class of service and account	Class-of-service value
A (account value used)	For neither	Default value
	For class of service only	Class-of-service value
	For account only	Account value
	For class of service and account	Account value
L (lesser of account and class-of-service values)	For neither	Default value
	For class of service only	Class-of-service value
	For account only	Default value
	For class of service and account	Lesser of the two values
G (greater of account and class of service values)	For neither	Default value
	For class of service only	Class-of-service value
	For account only	Default value
	For class of service and account	Greater of the two values

Note: For Boolean attributes, the system considers the value `true` greater than `false`.

The default values associated with class of service attributes are:

- For Boolean attributes: `false`
- For number attributes: `0`
- For character attributes: `null`

In the following example, the mailbox quota attribute is 3 MB for an account but 1 MB for this user's class of service. The quota value enforced on this account's mailbox for each precedence rule is as follows:

Precedence Rule	Value Applied to Account
C (class-of-service value used)	1 MB
A (account value used)	3 MB
L (lesser value used)	1 MB
G (greater value used)	3 MB

Preferences

The following table lists the available class-of-service preference attributes. You can also add custom attributes to this list if you want to increase the available class-of-service options.

Attribute	Rule	Type	Description
`pref_forwarding`	G	Boolean	Enables mail forwarding.
`pref_imap`	A	Boolean	Allows IMAP access.
`pref_imapssl`	A	Boolean	Allows IMAP access via SSL
`pref_intermanager`	A	Boolean	Allows administrator Web access through InterManager.
`pref_localdelivery`	G	Boolean	Allows the user to disable local delivery with forwarding enabled.
`pref_mtafilter`	A	Boolean	Causes the system to reject mail sent from a system-wide list of addresses.
`pref_numaliases`	A	Numeric	Set the maximum number of SMTP aliases.
`pref_pop`	A	Boolean	Allows POP access.

Attribute	Rule	Type	Description
`pref_popssl`	A	Boolean	Allows POP access with SSL.
`pref_quotabouncenotify`	L	Boolean	Notifies the recipient when the system bounces a message because delivery would exceed one of the recipient’s mailbox quotas.
`pref_quotamsgkb`	A	Numeric	Sets the maximum message size quota (in KB).
`pref_quotathreshold`	G	Numeric	Specifies teh quota warning threshold (set as a percentage).
`pref_quotatotkb`	A	Numeric	Sets the total mailbox size quota (in KB).
`pref_quotatotmsgs`	A	Numeric	Sets the quota for the maximum number of messages.
`pref_replymode`	A	Numeric	Sets the auto-reply mode (Vacation, Reply, or Echo).
`pref_selfcare`	A	Boolean	Sets Web access to the SelfCare interface.
`pref_smtp`	A	Boolean	Allows SMTP access.
`pref_smtpauth`	A	Boolean	Specifies that SMTP connections must use SMTP authentication.
`pref_smtpssl`	A	Boolean	Allows SMTP access with SSL.

Permissions

The following table lists the account permissions associated with a class of service. In all cases, permissions are numeric data types, with a value of `1` indicating `true` (access allowed) and a value of `0` indicating `false` (access denied).

Attribute	Rule	Type	Description
`perm_autoreply`	L	Numeric	Controls the end user's ability to select the Reply mode of auto-reply.
`perm_bypassauthentication`	L	Numeric	Controls the end user's ability to bypass future requests for authentication when using Web-based applications.

Attribute	Rule	Type	Description
`perm_echo`	L	Numeric	Controls the end user's ability to select the Echo mode of auto-reply.
`perm_forwarding`	L	Numeric	Controls the end user's ability to set `pref_forwarding`.
`perm_localdelivery`	L	Numeric	Controls the end user's ability to set `pref_localdelivery`.
`perm_mtafilter`	L	Numeric	Controls the end user's ability to set `pref_mtafilter`.
`perm_quotabouncenotify`	L	Numeric	Controls the end user's ability to set `pref_quotabouncenotify`.
`perm_quotathreshold`	L	Numeric	Controls the end user's ability to set `pref_quotathreshold`.
`perm_vacation`	L	Numeric	Controls the end user's ability to select the Vacation mode of auto-reply.

Note: The name of each defined permission attribute comes from the preference attribute to which the permission allows access. For example, the `perm_forwarding` permission controls the end user's ability to set the `pref_forwarding` preference attribute. It is recommended that you follow this convention when defining new attributes.

Default Class of Service

When InterMail is installed, a single class of service, called Default, is created automatically. The preference and permission attributes associated with the Default class of service are listed below.

Attribute	Value	Description
`pref_forwarding`	0	Disables mail forwarding.
`pref_imap`	1	Allows IMAP access.
`pref_intermanager`	1	Allows InterManager access.
`pref_localdelivery`	1	Enables local delivery.
`pref_mtafilter`	0	Disables MTA filtering.

Attribute	Value	Description
`pref_numaliases`	3	Sets a maximum of 3 SMTP aliases.
`pref_pop`	1	Allows POP access.
`pref_popssl`	1	Allows POP access with SSL.
`pref_webmail`	1	Allows access to WebEdge.
`pref_quotabouncenotify`	1	Notifies the recipient when the system bounces a message because delivery would exceed one of the recipient's mailbox quotas.
`pref_quotamsgkb`	10000	Seta a maximum message size quota of 10000 KB.
`pref_quotathreshold`	75	Sets a quota warning threshold of 75%.
`pref_quotatotkb`	100000	Sets a maximum mailbox size of 100000 KB.
`pref_quotatotmsgs`	1000	Sets a message quota of 1000.
`pref_replymode`	0	Turns off auto-reply.
`pref_selfcare`	1	Allows Web access to the SelfCare interface.
`pref_smtp`	1	Allows SMTP access.
`pref_smtpauth`	0	Does not require authentication for SMTP connections.
`pref_smtpssl`	1	Allows SMTP access with SSL.
`pref_webdisplay`	80	Sets a width of 80 for the Web interface column display.
`perm_autoreply`	1	Access allowed.
`perm_bypassauthentication`	1	Access allowed.
`perm_echo`	1	Access allowed.
`perm_forwarding`	1	Access allowed.

Attribute	Value	Description
`perm_localdelivery`	1	Access allowed.
`perm_mtafilter`	1	Access allowed.
`perm_quotabouncenotify`	1	Access allowed.
`perm_quotathreshold`	1	Access allowed.
`perm_vacation`	1	Access allowed.

As with other classes of service, you can modify the default class of service to provide additional services to all accounts within it.

Warning! You must not delete the Default class of service.

Note: When you create additional classes of service, they do not automatically include class-of-service attributes. For information on how to create classes of service and set attributes, see Chapter 6, Account Management.

Mailbox Quotas

Each InterMail mailbox is subject to limits, or quotas. Mailbox quotas are used to control the size of mailboxes and the size of messages that can be stored in these mailboxes. You typically define quotas at the level of a class of service to apply to all accounts associated with that class of service. However, you can also set quota values at the account level to override the quota values defined for the account's class of service.

This section describes over-quota policies that you can set. For information on setting mailbox quotas themseves, see the *InterMail Kx Reference Guide* if you are using `imdbcontrol` or the InterMail API libraries or Chapters 5 and 6 if you are using InterManager.

Setting the Over-Quota Policy

The configuration key `bounceOnQuotaFull` defines the overall quota bounce policy. The value of this configuration key determines the action taken by the InterMail system when delivery of a message would exceed one or more of the recipient's mailbox quotas. If the value is `true`, the system returns messages to sender if they would exceed the recipient's mailbox quotas. If the value is `false` (the default setting), it holds such undeliverable messages for subsequent delivery attempts.

Setting an over-quota policy is an important step to take before going into production, so you should review the value of `bounceOnQuotaFull` before beginning full mail service.

Setting Over-Quota Notifications

When the system bounces a message because it would have caused the recipient's mailbox to exceed one of its mailbox quotas, the intended recipient receives an automatic notification of the event. This informs the user that the system could not deliver a message and recommends that the user resolve the situation by removing other mail that is currently in his or her mailbox.

To log a message when a warning notification is sent to a user, you set the following configuration keys to `[log]`:

```
/*/mta/Error-Actions/MsLimitMsgSize:
             [return]
             [log]

/*/mta/Error-Actions/MsLimitNumMsgs:
             [return]
             [log]

/*/mta/Error-Actions/MsLimitTotalSize:
             [return]
             [log]
```

Note: Over-quota notices themselves do not bounce due to mailbox quotas. An account can receive bounce notifications even if its mailbox is beyond any of its quotas.

The configuration key `bounceQuotaNotice` defines the (configurable) text of the over-quota notification message. There is a default message for this configuration key, but you may alter this message as you wish. For example, you may choose to add a customer service phone number or other information specific to your site.

The default notification defined in `bounceQuotaNotice` is as follows:

```
Return-Path: <>
From: <Bounce_Notice_From>
Subject: <Bounce_Notice_Subject>
Date: <Bounce_Notice_Date>

A message was sent to you that was returned to
the sender(bounced) because it would have caused
your mailbox quota to be exceeded.

The following is the reason that the message was
over quota:

Quota Type: <Requested_Resource>
Quota Available: <Available_Resource>
Total Quota: <User_Quota>

The following is the information on the message
that was bounced:

Sender: <Bounced_Message_From>
Subject: <Bounced_Message_Subject>
Size: <Bounced_Message_Size>
Message ID: <Bounced_Message_ID>
Date: <Bounced_Message_Date>
Reply_To: <Bounced_Message_Reply_To>

To fix this problem, delete some messages from
your mailbox, and contact the sender to resend
the message.

If the size of the message is too big, contact
the sender to reduce the size of the message and
resend the message.
```

To change this over-quota notice, modify the value of `bounceQuotaNotice`.

You can also control the maximum number of unread over-quota notification messages to any mailbox. This option is useful for preventing extremely large numbers of messages from accumulating in the mailboxes of end users who read their mail infrequently. The configuration key `maxBounceNotices` defines this maximum number; the initial value is 20.

Note: For more information about the `bounceQuotaNotice` and `maxBounceNotices` configuration keys, see Chapter 2 of the *InterMail Kx Reference Guide*.

5

Site Management

After InterMail has been installed and configured, you as a service provider are ready to set up site characteristics and the structure of the delegated administration model that you wish to use. The process of setting up a site includes adding domains, organizations, and classes of service. Typically, these functions are performed through the InterManager interface; however, you can alternatively use one of the InterMail system’s other administrative interfaces.

This chapter describes the steps involved in setting up and managing a site in order to prepare InterMail for a production environment. It covers:

- Site structure
- Planning your site
- Site setup overview
- Using the InterManager interface
- Setting up domains
- Setting up classes of service
- Setting up organizations
- Adding users

About Site Structure

An InterMail site is a flexible hierarchy that you can define according to the unique requirements of your system. You manage this hierarchy using delegated administration.

Levels in the Hierarchy

An InterMail site hierarchy consists of the site, organizations, organizational units, and users.

Site

The site corresponds to the service provider. The site has all of the attributes of a standard organization but is unique in several respects:

- It is created when you initialize your InterManager installation and cannot be deleted.
- There is only one site per installation; it usually comprises multiple organizations.
- It is the only organization that can contain organizational units of the Consumer type.

Organizations

An organization corresponds to an entire company or other institution. Organizations are typically created when a company purchases an initial set of e-mail accounts and services. They are containers for organizational units and users.

Each organization has one or more associated domains. These are the domains that are used in the e-mail addresses of the organization's accounts. Because an organization corresponds to a company, the primary domain of an organization is typically a top-level domain that does not include a subdomain or host name. For example, the organization Software.com would have the primary domain `software.com` and possibly additional domains, such as:

```
hardware.com
jupiter.software.com
```

If the organization Software.com includes all of these domains, an e-mail account in this organization can have addresses that include any of these domains, such as:

```
john.doe@software.com
john.doe@hardware.com
john.doe@jupiter.software.com
```

Organizational Units

Organizational units are subsets of organizations, and can be nested within other organizational units. For example, a large company may be set up as a single organization containing a separate organizational unit for each division, each of which in turn contains distinct organizational units for each department within the division. An organizational unit can be one of two types:

- **Business organizational units** define their users as members of a particular organization. These users are typically the employees of a company that has purchased e-mail from the service provider.

- **Consumer organizational units** define groups of end users who are not associated with a particular company. These are the users who purchase personal or family e-mail access from the service provider.

  Note: Consumer organizational units can be created only in the site organization. When organizational units are created in other organizations, they are automatically defined as Business organizational units.

  Although Consumer organizational units are members of the service provider's organization (the site organization), the service provider's organization administrator cannot view or modify Consumer organizational units. This creates a distinction between users who are employees of the service provider (who are members of a Business organizational unit) and the individual consumer users who purchase e-mail access from the service provider (members of the Consumer organizational unit).
- Each organizational unit has one or more administrators.

Users

Users are associated with an organization or an organizational unit, but at any level in the hierarchy. Users can simply be moved between levels in a given organization; in contrast, to be moved from one organization to another, they must be deleted from one and re-created in the other.

Sample Site

Software.com is a service provider that wishes to host e-mail service for two client companies: Majorcorp.org and Minorcorp.com.

Majorcorp.org is a large organization that requires independent administration of accounts within its two major departments (Accounting and Marketing). Minorcorp.com is a smaller company that plans to administer all of its accounts in a single group.

In addition, Software.com has its own base of consumer users who purchase e-mail service directly from Software.com.

Figure 7 depicts a site capable of supporting these requirements.

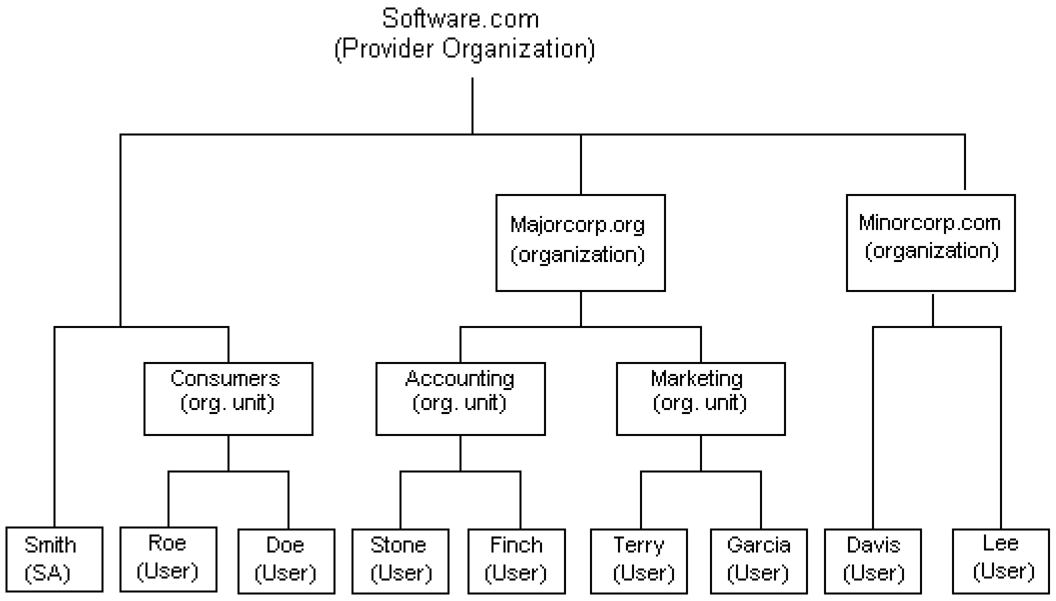


Figure 7 Sample site hierarchy

The model installation has three organizations: one for Software.com, one for Majorcorp.org, and one for Minorcorp.com.

In this hierarchy, the Software.com organization is the provider. It contains a single user, and an organizational unit that contains all consumer users.

The Majorcorp.org organization contains two organizational units, one for each of the company's two departments. Majorcorp.org's users are distributed between these two organizational units.

There are no organizational units associated with Minorcorp.com. Instead, all Minorcorp.com users are directly associated with the Minorcorp.com organization.

About Delegated Administration

With delegated administration, you manage your InterMail system by defining administrators at each level of the organizational hierarchy who can create and administer the system at their level and below. The administrators are:

- The **site administrator** (SA), a "super user" and a member of the service provider's organization. This user is typically a system administrator who is responsible for maintaining the InterMail system. Site administrators have the ability to create and modify all objects used in delegated administration and to assign all levels of administrative access. However, they typically limit their activities to the operations that only they can perform, such as creating new classes of service and assigning customer service administrators.
- The **customer service administrator** (CSA), who is typically an employee of the service provider and also a member of the service provider's organization.

Customer service administrators are responsible for creating new organizations and assigning organizational administrators. Customer service administrators also have the ability to manage organizations and the e-mail accounts associated with them, but typically delegate these tasks to the organization administrator.

- The **organization administrator** (OA), who is typically the mail administrator for a company that has purchased internal mail from the service provider and is responsible for managing the organizational units and e-mail accounts within an organization. Each organization administrator must be a member of the organization that he or she manages.

Note: Organization administrators can access only data associated with their assigned organizations. They have no control over the mail system as a whole, nor do they have access to data for other organizations to which they have not been assigned.

- The **organizational unit administrator** (OUA), who is typically the manager of an individual department within a company, and is responsible for managing e-mail accounts within that department (an organizational unit). An organizational unit administrator must be a member of the organizational unit's parent organization.

The following table depicts the relative authority that each administrator has in the site hierarchy.

Function	**SA**	**CSA**	**OA**	**OUA**
Create/delete/edit an SA	Y	N	N	N
View an SA	Y	N	N	N
Create/delete/edit a CSA	Y	N	N	N
View a CSA	Y	Y	N	N
Create/delete/edit an OA	Y	Y	N	N
View an OA	Y	Y	Y	N
Create/delete/edit an OUA	Y	Y	Y	N
View an OUA	Y	Y	Y	Y
Create/delete/edit a class of service	Y	N	N	N
View a class of service	Y	Y	N	N
Create/delete/edit a domain	Y	Y	N	N

Function	SA	CSA	OA	OUA
View a domain	Y	Y	Y	Y
Create/delete/edit an organization	Y	Y	N	N
View an organization	Y	Y	Y	Y
Create/delete/edit an organizational unit	Y	Y	Y	N
View an organizational unit	Y	Y	Y	Y
Create/delete/edit an account	Y	Y	Y	Y
View an account	Y	Y	Y	Y

Planning Your Site

Site implementation begins with a review of site requirements. Once you have defined your requirements, you can initialize the site and establish the structures to support your implementation.

During review you should do all of the following:

- List the specific organizations that you need to create, and consider what types of organizations you may need to create in the future. Identify each organization's service requirements, including mailbox quantities and quotas, domain names, addressing conventions, and so on. Determine the mail system modifications that will be necessary to support these requirements, such as additional classes of service, new DNS records, and so on. For detailed information about classes of service, see Chapter 4.
- Consider which organizations will require support from you, which will be creating their own organizational units, and which will need room for substantial growth.
- Identify the individuals within each organization who will act as organization administrators.
- Evaluate the method you will use to classify large numbers of users not affiliated with a client company. By default, these users are grouped into a single organizational unit, but you may want to subdivide them into more manageable organizational units within the default organizational unit.
- Identify required domain names. If you expect that companies or end users will use e-mail addresses with unique domain names, you as the service provider must be recognized as the party responsible for mail service to those domains. Accepting responsibility for additional mail domains is a two-step process. You gain external recognition by establishing MX records in the Domain Name

System (DNS), and you gain internal recognition by adding domain records to the mail server.

- Establish a naming convention that guarantees uniqueness among user addresses and login strings. This convention should be distributed to all your client organizations.
- Draft a site structure based on the above information. Your site structure should:
  - Separate organizations with unique names for each organization and organization unit, which limits customer access to the appropriate data.
  - Allow flexibility for expansion within each organization.
- Consider whether user account and user information should be batch-loaded. Typically, this information is entered one user or account at a time through the InterManager Web interface. However, if you need to create or import a large number of accounts into InterMail, you can use the `imbatchload` utility. To learn more about `imbatchload`, see Chapter 6.

Site Setup Overview

You use the InterManager interface to add site components such as organizations and accounts. Because of dependencies among the various structural elements, it is recommended that you create these components in the following order:

1. Add domain records required by all customers whose mail service you intend to host.
2. Create necessary classes of service and adjust configuration keys as required to avoid potential conflicts.
3. Create all top-level organizations and assign organizational administrators.
4. Create organizational units (optional).
5. Create users within the organizations and organizational units (optional).

Note: Before you can use InterManager to create these components, your site must already have one domain, organization, and class of service. These are normally created during the InterMail installation process.

Using the InterManager Interface

This section describes how to use InterManager, including how to log in and out and how to use the InterManager forms.

Logging In and Out

To access the interface, launch your Web browser and point to the host on which the InterManager application is running. Enter the login name and password for the site administrator, and click the Authenticate button.

Note: The login name and password were created at the time of installation during the InterManager configuration process. If you do not remember them, review the script file from the installation.

Once you have logged in to InterManager, your session remains active until you log out, or until 30 minutes have elapsed since you last requested a form. For security purposes, administrators should always terminate their sessions by manually logging out; this prevents non-administrators from accessing InterManager through a session that was left running by an administrator.

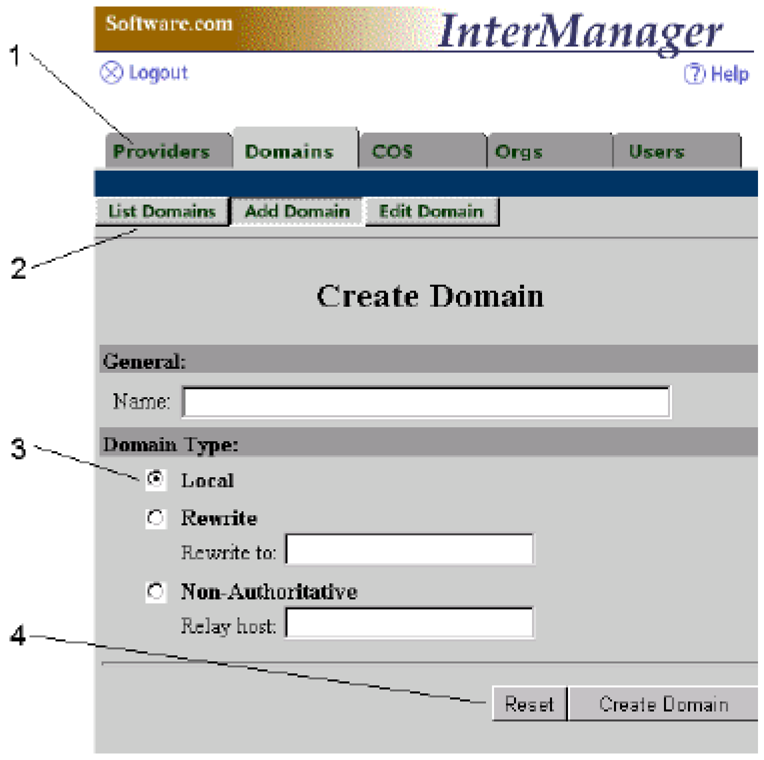
To log out of the InterManager interface, click the Logout button at the top of any form:

⊗ Logout

When you click the Logout button, the InterManager session is immediately terminated and the Authentication form is displayed. To execute subsequent InterManager operations, you must re-authenticate.

Using Administrative Forms

Administrative forms allow administrators to create, modify, and delete objects under their jurisdiction. Figure 8 shows a typical InterManager administrative Web form.



Create Domain

General:

Name:

Domain Type:

(● Local) Rewrite
Rewrite to:

Non-Authoritative
Relay host:

Figure 8 Site administrator's Create Domain form

To create or modify an object on an administrative form:

1. Select the tab for the type of form you want.
2. Click the button at the top of the tab for the particular form you want.
3. Enter and modify data in fields in the displayed form.
4. Click the execution button at the bottom of the form to commit your changes.

Using Search Forms

Search forms allow you to search and display lists of objects (such as organizations and users) that meet specified search criteria. Figure 9 shows the List Domains form, which can be used to generate a list of domains within a specific alphabetical range.

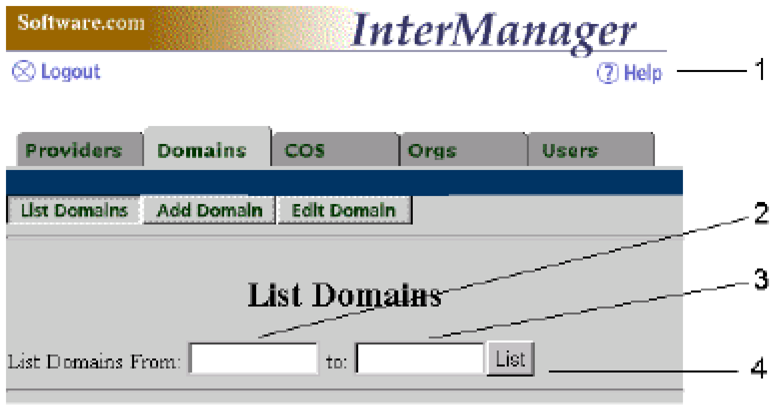


Figure 9 Site administrator’s List Domains form

To use a search form:

1. Click the Help button at any time to get information about the form and the operations that are performed with it.
2. In the first search criteria field, define the start of the search range. You can specify the name of a particular object, or use a wildcard (*).
3. In the second search criteria field, define the end of the search range.

> ***Note:*** Not all search forms have two fields for defining search criteria. Some, such as the List Organizations form, have only one search field.

4. Click the List button to execute the search with the given criteria. InterManager generates a list of objects whose names match the specified search criteria.

Although some InterManager objects will be few in number, most—including domains, organizations, and users—will number in the hundreds or thousands. Because building a list of thousands of objects can take a long time in the InterManager interface, it is recommended that you always make your search criteria as specific as possible.

Note: To search for users across all organizations, use the Users Form.

Setting Up Domains

To start setting up your site, you must add initial domains. Because domain identification plays a critical role in mail delivery, it is extremely important to identify domains correctly. 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.

You can create domains using InterManager, the InterMail `imdbcontrol` utility, or the C API.

Creating Domains with InterManager

To set up a domain in InterManager:

1. At the top of the Domains tab, click the Add Domain button.
2. In the Name field, enter a unique domain name of no more than 64 characters.
3. Specify whether the new domain is local, non-authoritative, or rewrite. (For help on these terms, see the InterManager online help.)
4. Click the Create Domain button at the bottom of the tab.

Creating Domains with imdbcontrol

The syntax you use with `imdbcontrol` depends on whether your are creating a local, non-authoritative, or rewrite domain.

Creating a Local Domain

Unless otherwise specified, `imdbcontrol` creates all domains as local domains. To create a local domain, enter:

```
imdbcontrol CreateDomain <DomainName>
```

where `<DomainName>` is the name of the local domain.

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

```
imdbcontrol CreateDomain majorcorp.com
```

Creating a Non-Authoritative Domain

To create a non-authoritative domain, enter:

```
imdbcontrol CreateDomain <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 in the non-authoritative domain.

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

```
imdbcontrol CreateDomain minorcorp.com nonauth pluto.minorcorp.com
imdbcontrol CreateDomain minorcorp.com nonauth pluto
```

Note: As a rule, you should use a fully-qualified domain name as the relay host. The only exception is if the relay host is a “pingable” host on your local network, in which case just the host name (in this example, `pluto`) is sufficient for the `relayHost` argument.

Creating a Rewrite Domain

The syntax for creating a rewrite domain is:

```
imdbcontrol CreateDomain <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 `majorcorp.com`, enter:

```
imdbcontrol CreateDomain minorcorp.com rewrite majorcorp.com
```

Creating Domains with the C API

To create a domain using the C API:

1. Write a source file. For example, to create a local domain called `majorcorp.com`:

```
extern "C" {
#include "im_domain.h"
}
#include <stdio.h>

int main(int argc, char **argv)

{

 if (IM_InitLibrary()) {
 printf("Init failed\n");
 }

 IM_Domain d;
 IM_Error err;

 IM_InitDomain (&d);
 IM_InitError (&err);

 d.name = "majorcorp.com";
 d.type = IM_DOMAIN_LOCAL;

 int ret = IM_CreateDomain(&d, &err);

 if (ret)
 {
 printf("IM_CreateDomain returned an error\n");
 printf("Errorcode = %d ErrorString = %s\n", err.number, err.string);
 }
}
```

2. Compile the source file. This involves invoking the C compiler, linking all necessary libraries, specifying the input file to be compiled, and specifying an output file to represent the executable file to be invoked in Step 3.

Note: To save time, it may be advantageous to edit the `.cshrc` or `.profile` file, adding an alias for the C compiler, all necessary libraries, and a user-defined source file. For example, to invoke the Sun compiler:

```
/usr/local/SUNWspro/bin/CC -o createAccount test.cxx -mt
-L@INTERMAIL/lib -lomu -lim -L$INTERMAIL/lib -lncr -
lcommon -lgeneric -lepc -lnlsrtl3 -lcore3 -lc3v6 -ldl -
lsocket -lnsl -lm –I$INTERMAIL/include
```

---

3. Execute the compiled source file. To do this, simply enter the name of this file.

Setting Up Classes of Service

After creating initial domains, you are ready to set configuration keys and then create classes of service for different types of users using either InterManager or the Perl API.

Setting Configuration Keys

Depending on the class-of-service attributes you want, you should make the following configuration key settings:

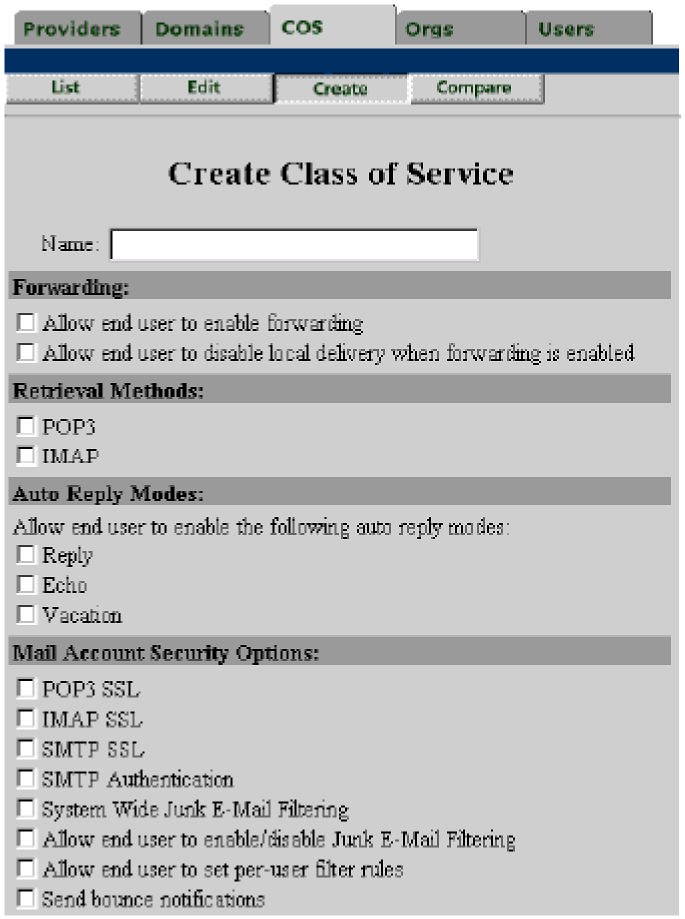
- Set the `blockPerAccount` configuration key to `true` if you want to create a class of service that includes mail blocking as an option. If the key is disabled (set to `false`), all users will have access to mail blocking, regardless of class of service.
- Set the value of the `maxMessageSizeInKb` configuration key to exceed the value you plan to 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 the value of this key will be ignored.

Note: For detailed information on these configuration keys, see the *InterMail Kx Reference Guide*. For instructions on changing configuration key settings, see Chapter 7 of this manual.

Creating Classes of Service with InterManager

To create a class of service:

1. At the top of the COS tab, click the Create button.
2. In the Name field, enter a name for the class. Keep the name simple; you may include alphanumeric characters.
3. Select the attributes you want to include in the class (Figure 10).



Create Class of Service

Name:

Forwarding:

☐ Allow end user to enable forwarding
☐ Allow end user to disable local delivery when forwarding is enabled

Retrieval Methods:

☐ POP3
☐ IMAP

Auto Reply Modes:

Allow end user to enable the following auto reply modes:
☐ Reply
☐ Echo
☐ Vacation

Mail Account Security Options:

☐ POP3 SSL
☐ IMAP SSL
☐ SMTP SSL
☐ SMTP Authentication
☐ System Wide Junk E-Mail Filtering
☐ Allow end user to enable/disable Junk E-Mail Filtering
☐ Allow end user to set per-user filter rules
☐ Send bounce notifications

Figure 10 Creating a class of service (partial screen)

4. When you have finished, click the Create Class of Service button at the bottom of the tab.

Creating Classes of Service with the Perl API

To create a class of service using the Perl API:

1. Create a source file. This involves setting a `$newCos` variable for an array containing class-of-service attributes that will be added to a new class of service called `standard`. The `$cos` routine defines the class-of-service name and passes

in the attributes specified in `$newCos`. For example, to create a class of service called “standard:

```
use SwCom::Mail::All;

im_init() or die “Call to initialize library failed”;

$newCos =
    {
        pref_pop                      => '1',
        pref_popssl                   => '1',
        pref_smtp                     => '1',
        pref_smtpssl                  => '1',
        pref_imap                     => '1',
        pref_smtpauth                 => '1',
        pref_mtafilter                => '1',
        pref_quotabouncenotify        => '1',
        perm_forwarding               => '1',
        perm_vacation                 => '1',
        perm_autoreply                => '1',
        perm_echo                     => '1',
        perm_localdelivery            => '1',
        pref_intermanager             => '1',
        pref_webmail                  => '1',
        pref_quotatotkb               => '100000',
        pref_quotamsgkb               => '100000',
        pref_quotathreshold           => '10',
        pref_numaliases               => 5
        };

my $cos = new Cos(
         name => 'standard',
         serviceDef => $newCos
         );

my $err = $cos->Create();
print "Create Failed\n" unless $err;

1;
```

2. Call and initialize the Perl library. All InterMail Perl modules must declare the use of the InterMail library `SwCom::Mail::All`. Before making any calls to that library, you must initialize it with a call to `im_init()`.

3. Create a Perl routine for the desired function.

4. Create a structure for error reporting so you can make sure that the source file you create runs properly.

Note: For more information on using the Perl and C APIs, see the *InterMail Kx Reference Guide*.

5. Execute the Perl function. For example, to create the new class of service named in Step 1, issue the following:

```
perl MakeCos
```

Note: The successful execution of a Perl source file depends on correct environment settings. These settings should be correct upon installation; however, if problems occur, make sure that your `PATH` and `PERLHOME` variables are set to the proper location.

Setting Up Organizations

After creating necessary domains and classes of service, you are ready to add organizations. This operation is typically carried out by a customer service administrator at the company purchasing e-mail access from the service provider, but it can be performed by the service provider's site administrator.

Adding an organization is a three-step process, involving:

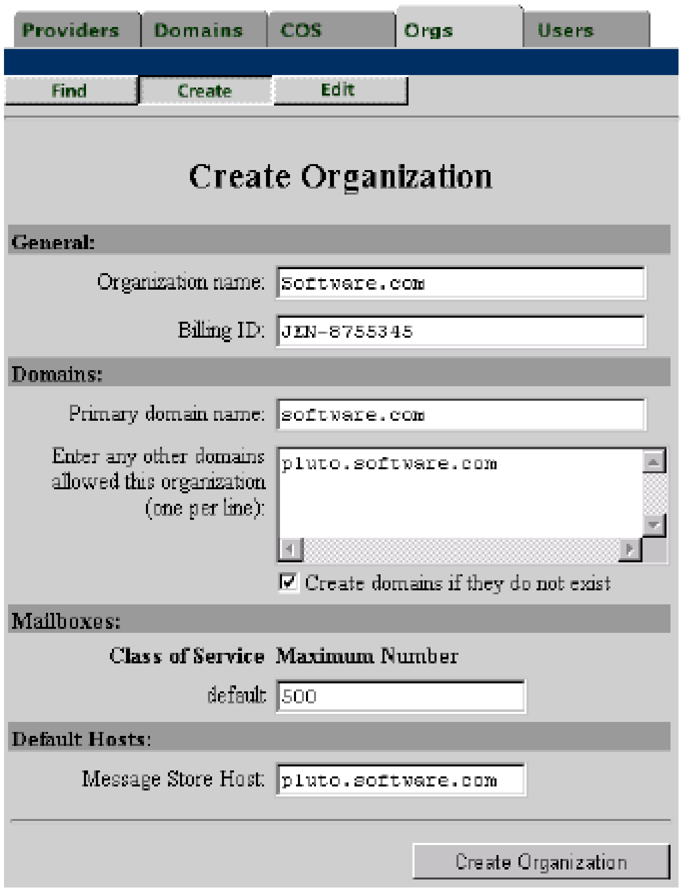
1. Creating the organization.
2. Creating the user who will administer the organization.
3. Assigning the new user as the organization's administrator.

This section describes each of these steps in detail.

Creating an Organization

To create an organization:

1. Log in to InterManager.
2. On the Orgs tab, click the Create button.
3. Fill in the fields for the name, billing ID (if any), domain, class of service, and Message Store host associated with the organization. Figure 11 depicts the creation of an organization called software.com associated with the domain software.com.



Create Organization

General:

Organization name: Software.com

Billing ID: JIN-8755345

Domains:

Primary domain name: software.com

Enter any other domains allowed this organization (one per line): pluto.software.com

☑ Create domains if they do not exist

Mailboxes:

Class of Service	Maximum Number
default	500

Default Hosts:

Message Store Host: pluto.software.com

Create Organization

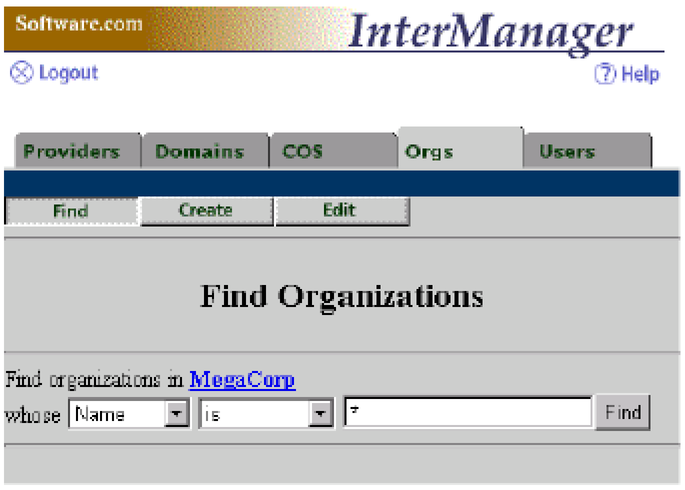
Figure 11 Create Organization form

4. Click the Create Organization button at the bottom of the form to create the organization in the Integrated Services Directory. If you selected the check box Create Domains If They Do Not Already Exist, any new domains specified for the organization are also created.

Creating the Administrator's Account

To create an administrator within the organization:

1. On the Orgs tab, click the Find button to display the Find Organizations form, as shown in Figure 12.



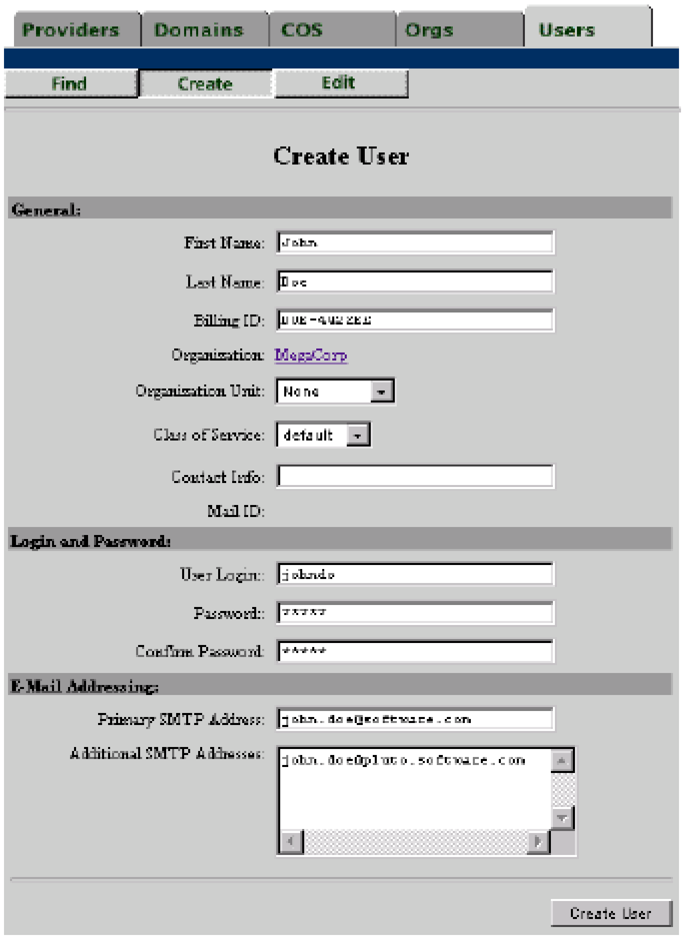
Find Organizations

Find organizations in MegaCorp

whose Name is * Find

Figure 12 Find Organizations form

2. Type the name of the new organization in the search field, and click the adjacent Find button.
3. Click the link labeled “Create user in this Organization”. This displays the Create User form, as shown in Figure 13.



Create User

General:	
First Name:	John
Last Name:	Doe
Billing ID:	[illegible]
Organization:	MegaCorp
Organization Unit:	None
Class of Service:	default
Contact Info:	
Mail ID:	
Login and Password:	
User Login:	johndo
Password:	*****
Confirm Password:	*****
E-Mail Addressing:	
Primary SMTP Address:	john.doe@software.com
Additional SMTP Addresses:	john.doe@pluto.software.com

Figure 13 Create User form

4. Fill in the fields to create the user you want to designate as the organization administrator.
5. Click the Create User button at the bottom of the form to create the user and associated account in the Integrated Services Directory.

Assigning the Organization Administrator

To assign the new user as the organization administrator:

1. On the Orgs tab, click the Edit button to display the Organization Info form.
2. Locate the Administrator’s section of the form, and click the Add link to display the Add Administrator form.
3. Click Find to generate a user list, select the check box next to the user’s name on this form, and click the Add Administrators button.

The new organization administrator can now manage the users and organizational units within his or her organization, and assign additional organization administrators.

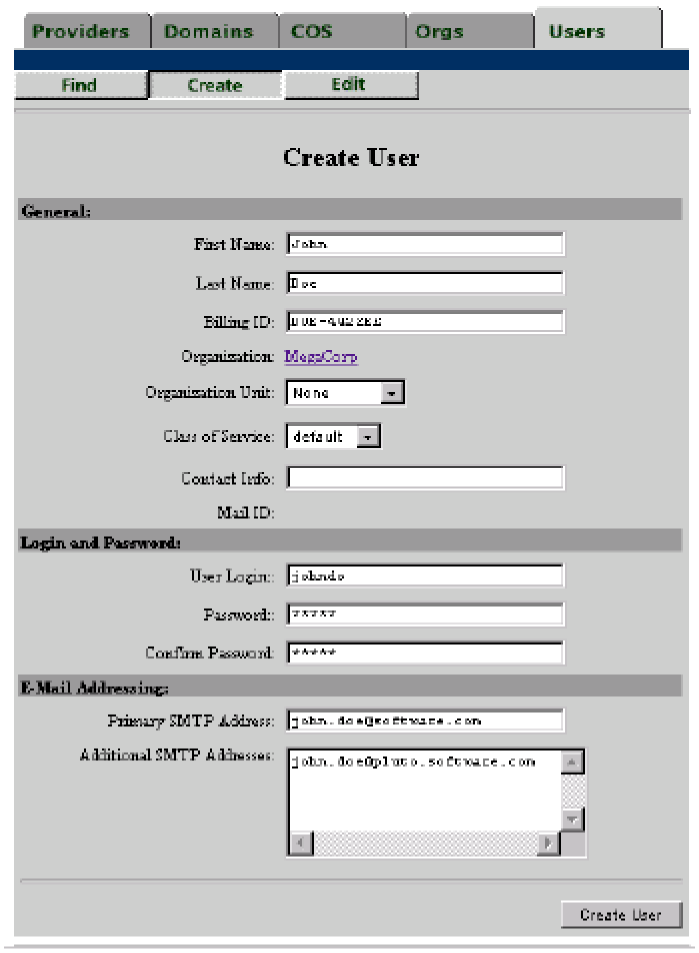
Adding Users

The final step in setting up your site is to add users. If you want to create a large number of accounts simultaneously, you can do so using the InterMail `imbatchload` utility. For more information on `imbatchload`, see Chapter 6.

More typically, users are created individually by administrators using InterManager at all levels of the hierarchy.

To add a user using InterManager:

1. On the Users tab, click the Create button.
2. Fill in the fields, and click the Create User button at the bottom of the form to create the user and associated account in the Integrated Services Directory (Figure 14).



Create User

General:

First Name:	John
Last Name:	Doe
Billing ID:	[illegible]
Organization:	MegaCorp
Organization Unit:	None
Class of Service:	default
Contact Info:	
Mail ID:	

Login and Password:

User Login:	johndo
Password:	*****
Confirm Password:	*****

E-Mail Addressing:

Primary SMTP Address:	john.doe@software.com
Additional SMTP Addresses:	john.doe@pluto.software.com

Figure 14 Creating a user

After adding a new user, you can edit his or her class of service. See “Setting Up Classes of Service” on page 62.

6

Account Management

When an InterMail site is configured, an initial set of accounts is created, which you will need to add, change, and view. These account management tasks can be handled by any level of group administrator—site administrator (SA), customer service administrator (CSA), organization administrator (OA), or organizational unit Administrator (OUA)—or by an administrator with system access but no group administrative privileges.

This chapter covers:

- Management interfaces
- Using InterManager
- Using command-line utilities
- Creating accounts with APIs
- Advanced account management

Management Interfaces

As discussed in Chapter 5, InterMail contains several user interfaces for account management: the InterManager Web interface, command-line utilities, the Perl API, and the C API. You can carry out many administration tasks using any of these interfaces. However, some tasks are most easily or efficiently performed with a particular interface:

For this task:	Use this interface:
Creating individual new users	InterManager `imdbcontrol`
Creating multiple new users	• C API • Perl API • `imdbcontrol` in batch mode
Comparing classes of service	InterManager

For this task:	Use this interface:
Modifying the class-of-service attributes of an account	`imdbcontrol`

Before deciding which interface to use, be aware of the relative authority and permissions that each requires:

- To use InterManager, you must be a site, customer service, organization, or organizational unit administrator. You must also have been assigned group privileges in InterManager.
- To use the command-line utilities, the Perl API, or the C API, you must be an administrator with access to a machine that runs InterMail and to the InterMail user/password that can execute command-line utilities.

Using InterManager

InterManager is an easy-to-use interface that allows you to create users in organizations and assign them to classes of service in one login session.

This section tells you how to use InterManager to:

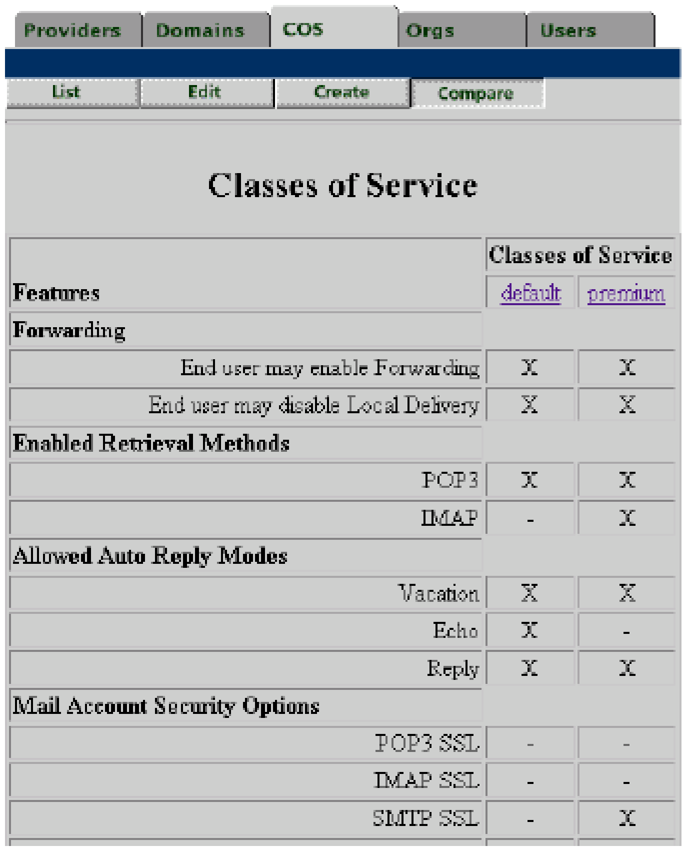
- Compare classes of service
- Create individual new users
- Modify individual users

Note: For instructions on logging into and out of and on navigating the InterManager interface, see Using the InterManager Interface in Chapter 5.

Comparing Classes of Service

There are times when you will want to compare the different attributes of multiple classes of service. Although it is possible to list class-of-service attributes with the command-line utilities, the InterManager interface is much more convenient.

To compare classes of service, simply go to the COS tab, select the classes of service you want to compare, and click the Compare button, as shown in Figure 15.



Classes of Service

Features	Classes of Service	
	default	premium
Forwarding		
End user may enable Forwarding	X	X
End user may disable Local Delivery	X	X
Enabled Retrieval Methods		
POP3	X	X
IMAP	-	X
Allowed Auto Reply Modes		
Vacation	X	X
Echo	X	-
Reply	X	X
Mail Account Security Options		
POP3 SSL	-	-
IMAP SSL	-	-
SMTP SSL	-	X

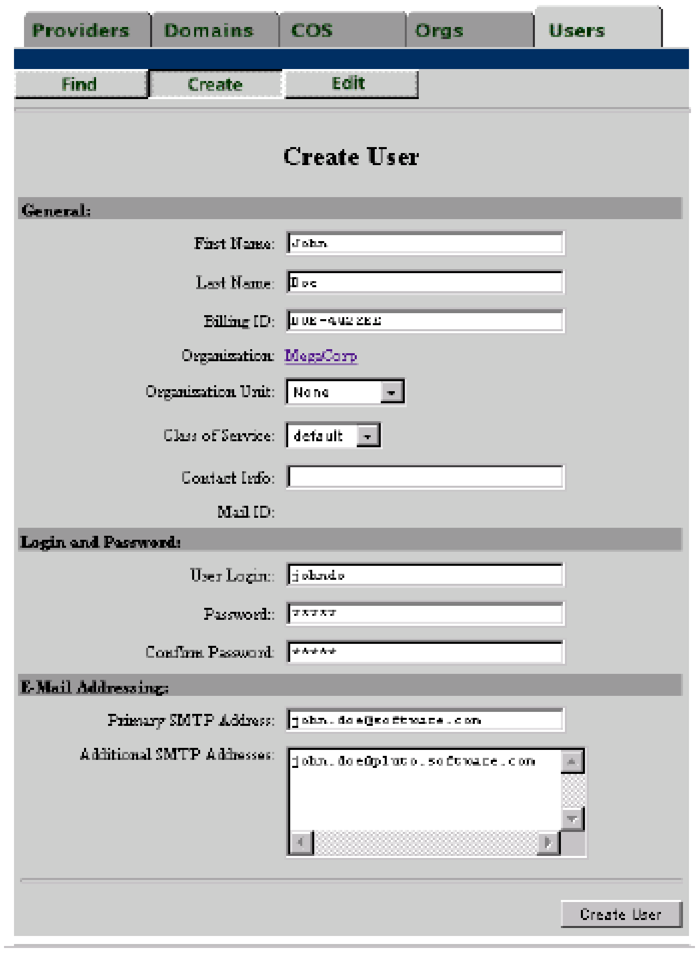
Figure 15 Viewing multiple classes of service in InterManager

Creating a User

Before creating a user in InterManager, make sure that you have sufficient permission. For example, if you log in as an organizational unit administrator, you will be able to create users only in the organizational unit that you manage.

To create a user:

1. Go to the Users tab and click the Create button.
2. Fill in the fields shown in Figure 16.



Create User

General:

First Name:	John
Last Name:	Doe
Billing ID:	008-442288
Organization:	MegaCorp
Organization Unit:	None
Class of Service:	default
Contact Info:	
Mail ID:	

Login and Password:

User Login:	johndo
Password:	*****
Confirm Password:	*****

E-Mail Addressing:

Primary SMTP Address:	john.doe@software.com
Additional SMTP Addresses:	john.doe@pluto.software.com

Figure 16 Using InterManager to create a new user

3. Click Create User at the bottom of the tab.

Modifying a User

User data can be edited for existing users. Typically, the fields that are edited for existing users are the Class of Service (when a user wants greater account privileges, for example) and Password fields.

Other user modifications may include adding or changing billing information, changing contact information when a new administrator is assigned to manage an account, or changing the organizational unit to which the user belongs.

Note: You cannot use InterManager to modify a user’s SMTP address. To modify an SMTP address, use `imdbcontrol ModifyAccountSmtp`.

To edit user data:

1. Go to the Users tab and click the Edit button
2. Enter the SMTP address of the user you want to edit.

3. Modify the fields on the Edit User tab.
4. Click the Save Changes button at the bottom of the tab.

Using Command-Line Utilities

InterMail also provides command-line utilities for managing accounts and account-related operations. Command-line utilities are advantageous in that they make scripting easy, can be automated, can be integrated with other UNIX tools, can read from files in batch mode, and can write reportable transactions that can be transferred to other applications.

This section tells you how to:

- Create, modify, and delete accounts using `imdbcontrol`
- Work with class-of-service attributes using `imdbcontrol`
- Create large numbers of accounts for use with InterManager using `imbatchload`
- List accounts and characteristic using `imdbcontrol`
- Set auto-reply messages on an account basis using `imreplyctrl`

Creating an Account

To create an InterMail account through the command-line interface, you use the `imdbcontrol` command with the `CreateAccount` argument. For example:

```
paris% imdbcontrol CreateAccount john.doe paris 12345 jdoe rosebud
clear software.com A S Basic
```

This example shows the creation of an account that has the following attributes:

- The SMTP address `john.doe`
- A mailbox on the host `paris`
- The internal ID number `12345`
- The POP/IMAP login name `jdoe`
- The password `rosebud` (clear, not hashed)
- The domain `software.com` (making the e-mail address `john.doe@software.com`)
- The status Active (`A`) and the account type Standard (`S`)
- The class of service `Basic`

Note: All `imdbcontrol` arguments can be abbreviated to the first letters of the words in the argument. In the previous example, for instance, the expression `CreateAccount` could have been abbreviated `ca`. For more information, see Chapter 3, Directory Management Utilities, of the *InterMail Kx Reference Guide*.

You can also use `imdbcontrol` to create a number of accounts in one batch process. This can be useful for quick creation of a number of temporary accounts for testing purposes. To use `imdbcontrol` in batch mode, create an input file with the `imdbcontrol` command lines you want to run, minus the word `imdbcontrol`. 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]
```

After this file has been created, run `imdbcontrol` in batch mode by typing:

```
imdbcontrol - < input
```

Note: It is not necessary that all commands in the file be `CreateAccount`; you can use different types of `imdbcontrol` commands in the same file.

Modifying an Account

You modify account information using the `imdbcontrol ModifyAccountSmtp` argument. In the following example, the SMTP address and domain information for `john.doe` are modified:

```
imdbcontrol ModifyAccountSmtp john.doe software.com jdoe majorcorp.org
```

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

Deleting an Account

You delete accounts using the `imdbcontrol DeleteAccount` argument. In the following example, the account for `john.doe` is deleted from `majorcorp.org`:

```
imdbcontrol DeleteAccount john.doe majorcorp.org
```

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

Working with Class-of-Service Attributes

You may occasionally want to examine class-of-service attributes and modify them 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 for some accounts. To do this, you use the `imdbcontrol`

GetAccountCos (or imdbcontrol gac) and imdbcontrol SetAccountCos (or imdbcontrol sac) commands.

You use the imdbcontrol GetAccountCos commands to list the class-of-service attributes and values for an account. For example:

```
paris% imdbcontrol GetAccountCos brock.lee minorcorp.com

Global COS Name: premium

    Attribute Name              Syntax Rule Global COS Account COS Result COS
    ==============              ====== ==== ========== =========== ==========

perm_autoreply                    N     L   1*         <undef>     1
perm_bypassauthentication         N     L   1*         0           0
perm_echo                         N     L   1*         <undef>     1
perm_forwarding                   N     L   1*         <undef>     1
perm_localdelivery                N     L   1*         <undef>     1
perm_mtafilter                    N     L   1*         <undef>     1
perm_quotabouncenotify            N     L   0          <undef>     0
perm_quotathreshold               N     L   1*         <undef>     1
perm_vacation                     N     L   1*         <undef>     1
pref_forwarding                   B     G   0          <undef>     0
pref_imap                         B     A   1*         <undef>     1
pref_intermanager                 B     A   0*         <undef>     0
pref_localdelivery                B     G   1*         <undef>     1
pref_mtafilter                    B     A   1*         <undef>     1
pref_numaliases                   N     A   5*         <undef>     5
pref_pop                          B     A   1*         <undef>     1
pref_popssl                       B     A   1*         <undef>     1
pref_quotabouncenotify            B     L   1*         <undef>     1
pref_quotamsgkb                   N     A   1000*      <undef>     1000
pref_quotathreshold               N     G   75*        <undef>     75
pref_quotatotkb                   N     A   10000*     <undef>     10000
pref_quotatotmsgs                 N     A   0          <undef>     0
pref_replymode                    N     A   0          <undef>     0
pref_selfcare                     B     A   1*         <undef>     1
pref_smtp                         B     A   0          <undef>     0
pref_smtpauth                     B     A   1*         <undef>     1
pref_smtpssl                      B     A   1*         <undef>     1
pref_webmail                      B     A   1*         <undef>     1
(* -- Assigned Value)
```

After determining the class-of-service attributes for an account, you can adjust individual attributes using imdbcontrol SetAccountCos. When you set a class-of-service attribute for an account, the account is modified accordingly.

Creating Accounts for Use with InterManager

After a site has been initialized, you may wish to add a number of accounts for use with InterManager. You do this by creating a properly formatted input file and using the imbatchload utility.

You typically use this type of operation when you have installed InterMail, configured your site, added some initial accounts, and now wish to add a number of new users without going through the steps required by InterManager interface. This can save a large amount of time when you have a large number of accounts to add.

Creating an Input File

The `imbatchload` utility requires an extended LDIF input file. The full allowable syntax for this file is described in Chapter 3, Directory Management Utilities, of the *InterMail Kx Reference Guide*.

In the following example, two accounts each are added to `minorcorp.com` (`Marge.Harding` and `Bill.Lee`) and to `majorcorp.org` (`Joe.Mocha` and `Penny.Lane`).

Note: The following example depicts the minimum entries required to create the desired accounts in the `minorcorp` and `majorcorp` organizations. This example assumes that these organizations already exist.

```
option:
error: abort

context:
provider: Software.com, Inc

IM_Type: Org
ou: minorcorp

IM_Type: Person
uid: marge.harding@minorcorp.com
userlogin: marge
mailCos: basic
mailid: 33485958934
messagestorehost: minorcorp.com

IM_Type: Person
uid: bill.lee@minorcorp.com
userlogin: bill
mailCos: basic
mailid: 637658746584
messagestorehost: minorcorp.com

IM_Type: Org
ou: megacorp

IM_Type: Person
uid: joe.mocha@majorcorp.orgmajorcorp.org
userlogin: joe
mailCos: basic
mailid: 684568453
messagestorehost: majorcorp.org

IM_Type: Person
uid: penny.lane@majorcorp.org
userlogin: penny
mailCos: basic
mailid: 12394958
messagestorehost: majorcorp.org
```

Running imbatchload

In order to process the input file, you run `imbatchload` on the input file:

```
imbatchload <filename>
```

After you successfully run the `imbatchload` command, the accounts specified in the input file are active.

Listing Available Accounts

As an administrator, you can produce a list of account information through the `imdbcontrol ListAccounts` command.

Using imdbcontrol ListAccounts

The `imdbcontrol ListAccounts` command produces a list of accounts and account information. The `imdbcontrol ListAccounts` command also produces additional information, including:

- Local delivery information (P for POP, I for IMAP, W for Web, and N for None)
- Whether forwarding information is turned on
- Whether auto-reply is on, and if so, its mode
- 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
```

You can also use the `imdbcontrol ListAccounts` command with a typical UNIX search utility such as `grep`. For example, to list all accounts that have the name “bill” in the account record, including the SMTP address, login name, password, or even domain name, you might enter:

```
imdbcontrol ListAccounts | grep bill
```

Setting Auto-Reply for a User

Although auto-reply information is typically set in the InterManager interface, it can also be set through `imreplyctrl`.

To use `imreplyctrl` to set an auto-reply message for a user:

1. Create a plain text file that will serve as the auto-reply message—for example, `message.txt`.

2. Set this text file as the auto-reply message for the user:

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

3. Make sure that the user is associated with a class of service that has the necessary attributes. If necessary, add the class-of-service attribute as shown in "Working with Class-of-Service Attributes" on page 76.

Creating Accounts with APIs

InterMail allows you to create accounts using the Perl and C APIs.

Note: For more information on the Perl API, see Chapter 6, InterMail Perl API, in the *InterMail Kx Reference Guide*. For more information on the C API, see Chapter 5, InterMail C API, in the *InterMail Kx Reference Guide*.

Using the Perl API

The following example shows an input file used to create an account with the Perl API:

```
use SwCom::Mail::All;

im_init();

my $user = new Account(smtpAddress => "johndoe\@minorcorp.com",
                       popAddress => "johndoe",
                       mssHost => "paris",
                       emailIntID => 121212121,
                       plainPassword => "hello",
                       hash => IM_PWHASH_CLEAR,
                       status => IM_ACSTATUS_ACTIVE,
                       local => IM_DELIVERY_ENABLED);

my $err = $user->Create();
print "Create Failed\n" unless $err;

1;
```

The variable `$user` is instantiated as type Account and initialized with the appropriate values for the user. The subsequent call to `Create()` does the actual creation, and returns 1 on success, 0 on failure.

The characteristics shown are the minimum required for account creation; other characteristics are also available. For more information on account characteristics, see Chapter 6.

Using the C API

The following example shows a C source file used to create an account with the C API:

```
extern "C" {
#include "im_account.h"
}
#include <stdio.h>

main(int argc, char **argv)
{
   if (IM_InitLibrary()) {
       printf("Init failed\n");
   }

   IM_Account a;
   IM_Error err;

   IM_InitAccount (&a);
   IM_InitError (&err);

   a.smtpAddress = "janedoe@minorcorp.com";
   a.mssHost = "paris";
   a.emailIntID = "323232";
   a.popAddress = "jane";
   a.plainPassword = "doe";

   int ret = IM_CreateAccount(&a, &err);

   if (ret)
   {
     printf("IM_CreateAccounts returned an error\n");
     printf("Errorcode = %d ErrorString = %s\n", err.number,
err.string);
   }
}
```

After the C source file has been created, it must be compiled and then executed. For information on compiling and executing a C source file, see the *InterMail Kx Reference Guide*.

Advanced Account Management

The examples shown in this chapter provide some instruction on performing individual account management tasks using InterManager, command-line utilities, and APIs. However, in typical account administration, many tasks are performed together. For example, when you create a user, you will likely also want to assign the user to a class of service and place the user in an organization at the same time.

There are many ways to develop user applications using the various interfaces. The following is a list of suggestions for using the interfaces.

If there are administration tasks that you carry out repeatedly, you might consider building a front-end to the appropriate InterMail interface that can accept user input and that is generic enough to be reused. For example, if you prefer or require Web access, you might construct a Web form with text input areas, such that the form can then invoke a `.cgi` script to 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 uses this input when you create a Perl or C API source file or invoke 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:
imdbcontrol ca $smtpaddr $msshost $InternalID $popname $password clear
$domain A S premium
imdbcontrol sac $smtpaddr $domain pref_quotatotkb 10000000
imdbcontrol sac $smtpaddr $domain pref_quotamsgkb 1000000
```

The script in this example creates a new user, assigns the user to the `Premium` class of service, then modifies two class-of-service attributes, `pref_quotatotkb` and `pref_quotamsgkb`, to give this user greater account quotas than other members of the `Premium` class of service.

After you create this script and add execute permission for it, you invoke it as follows:

```
Superpremiumuser <smtpaddress> <internal id>
```

For example:

```
Superpremiumuser joebob 1284738
```

7

System Configuration

Although you are not likely to need to change system configuration settings for the initial operation of InterMail, it may become necessary later to modify some configuration options. This chapter discusses the tasks related to viewing and updating the Configuration database, and includes instructions for making configuration changes.

This chapter covers the following:

- The InterMail Configuration database
- Configuration key format
- Configuration key hierarchy
- Modifying the Configuration database

About the Configuration Database

All InterMail components are controlled by a Configuration database, which contains values for all configuration options, or keys, for each server. This configuration data is stored in a file named `config.db`. Although each InterMail host keeps a local copy of the Configuration database, there is only one master Configuration database, controlled by a Configuration server on a single host (the master configuration host). The Configuration server responds to requests by individual servers for updated configuration information. You can make changes to the Configuration database only on the master configuration host.

When configuration keys are modified, information about all potential changes is sent to each InterMail server. Each server then responds by indicating the impact of the changes. For example, a server may report that it it would need to be restarted to accommodate the changes. As the administrator, you are informed of the effects of the changes and have the option to commit or cancel the changes.

Once changes to the Configuration database are committed, each host is alerted to the presence of configuration changes and requests a copy of the new `config.db` file from the Configuration server. The new `config.db` file overwrites the previous `config.db` file, and all previous configuration data is discarded. Because configuration changes are automatically propagated throughout the system, manual intervention is not required.

Configuration Key Format

Configuration keys appear in the form:

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

Where:

`host`	Defines the host to which the key applies. If the key applies to all hosts, the wildcard (`*`) character is used.
`server`	Defines the specific InterMail server such as the POP server or MTA.) to which the key applies. The keyword `common` is used if the key applies to all servers.
`keyName`	Is the name of the configuration key. Some MTA configuration key names such as `/*/mta/Error-Actions/badReturn` are hierarchical with two levels.
`value`	Defines the value of the key. Some configuration keys take only a single value; others can have multiple values. All configuration key values appear between square brackets ([]).

For example, the following configuration key sets the MTA server option for automatically creating an account's mailbox the first time a message arrives for the account:

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

Because this key identifies the host (paris) and server type (`mta`) explicitly, this entry affects only the MTA on the host `paris`.

Configuration Key Hierarchy

The Configuration database can contain multiple entries for a single configuration key. In such a case, each InterMail server uses a hierarchy, from specific to general, to determine the appropriate value:

1. The server searches for an entry that defines the key explicitly for the host and server. For example:

   ```
   /paris/<server>/<key>
   ```

2. If there are no entries for the specific server on a particular host, the server searches the hierarchy for keys that applies to all servers on that host. For example:

   `/paris/common/<key>`

3. If there are no entries for the specific host, the server searches the database for an entry that applies to a particular server type on any host.For example:

   `/*/mta/<key>`

4. Finally, if the server finds no entries exist for the key anywhere in the Configuration database, the server uses the key's default value.

For example, when an MTA is started up on the host `paris`, the MTA reads the Configuration database to determine—among other things—the default domain name that it should use for address completion and other tasks (as defined by the configuration key `defaultDomain`). When searching the Configuration database, the MTA looks for the following Configuration database entries in this order:

```
1.  /paris/mta/defaultDomain
2.  /paris/common/defaultDomain
3.  /*/mta/defaultDomain
4.  /*/common/defaultDomain
```

To define its default domain, the MTA on `paris` uses the value of the first entry in this list that it finds in the Configuration database.

This hierarchy allows you to specify system-wide configuration values, but define specific exceptions to those policies on a per-server or per-host basis. The order of the configuration key entries in the `config.db` file has no effect on the hierarchy.

Modifying Configuration Data

Once the InterMail software is installed, you are ready to modify configuration options to suit your site requirements. You can do most of the essential configuration editing using the Web-based Configuration Editor. Under some circumstances, however, you will need to edit configuration keys directly using the `imconfedit` command-line utility.

Using the Configuration Editor

The Configuration Editor allows any authorized user to access and modify the Configuration database. All basic keys that are required to get the system up and running are available with this tool.

Modifying Configuration database information with the Configuration Editor involves four basic steps:

1. Accessing the Configuration Editor.
2. Accessing the configuration page on which you want to change keys.

3. Changing key values by entering changes in the appropriate fields.
4. Depending on change impact information reported by individual InterMail servers, committing or canceling your changes.

This section describes these steps in detail.

Note: For a list of all of the keys available through the Configuration Editor, see Appendix A. The keys are listed according to the functional groups in which they appear in the Editor.

Accessing the Configuration Editor

To access the Configuration Editor:

1. Enter the appropriate URL in your Web browser. The URL includes the fully qualified hostname of your InterMail host and the port number on which the Web server is running. For example, if the Web server were running on the standard httpd port (80), you might enter:

   ```
   paris.software.com/confedit
   ```

   If the Web server were running on a non-standard port, you would enter the port number along with the host name and `confedit` suffix. For example:

   ```
   paris.software.com:1080/confedit
   ```

2. In the login window, enter the Site Administrator’s username and password, created at the time of IntarMail installation. Once authentication is successful, the top-level page of the Configuration Editor (shown in Figure 17) is displayed.

Note: If you do not know the Site Administrator’s username and password, view the `imanLogin` and `imanPass` configuration keys in the `config.db` file.

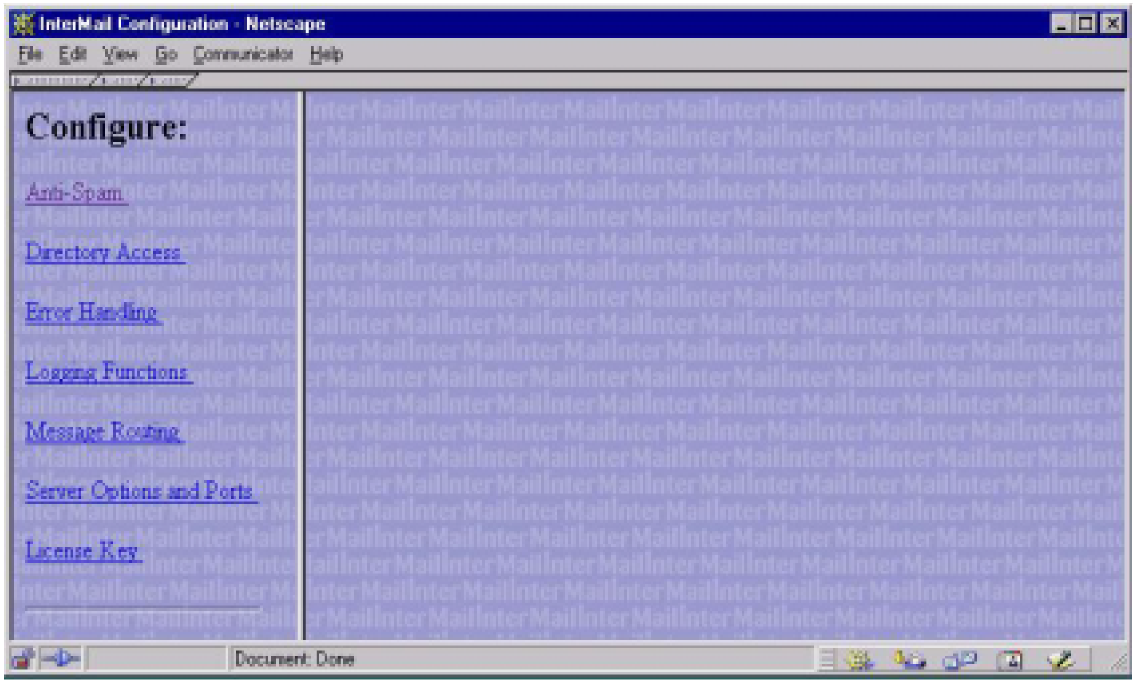


Figure 17 Top-level options in the Configuration Editor

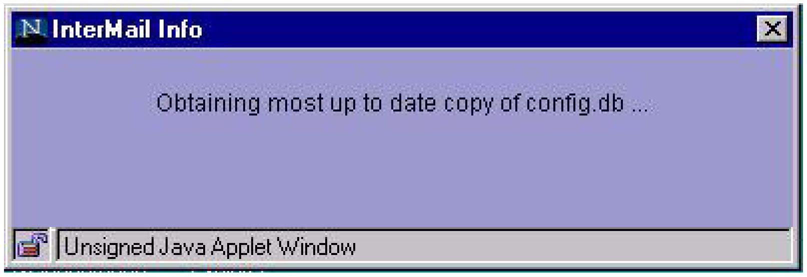
All configuration keys that are available through the editor are classified into functional categories:

- **Anti-Spam**—Keys related to mail blocking, connection dropping, relay prevention, mail sidelining, and other security features.
- **Directory Access**—Keys that control access to the Integrated Services Directory (ISD) and that specify the location of certain directories, such as the installation, log, and working directories for various InterMail servers.
- **Error Handling**—Keys that specify the actions to be taken by InterMail servers when certain error conditions occur. Most of these keys are MTA-related.
- **Logging Functions**—Keys used to configure logging operations.
- **Message Routing**—Keys that help control mail routing operations, such as setting the default domain and specifying rewrite domains.
- **License Key**—Key that determines the number of maximum number of users and groups allowed in the system.

If you are logged in to the Configuration Editor but are not actively using it, your session will expire automatically after a specified period of inactivity. You set this timeout interval with the `sessionExpireTimeoutSecs` configuration key.

Accessing a Configuration Page

To access one of the Configuration pages, click the appropriate link (for example, Logging Functions). The Configuration Editor displays a small status window that reports the operations being performed by the editor. Figure 18 shows an example of a status window.

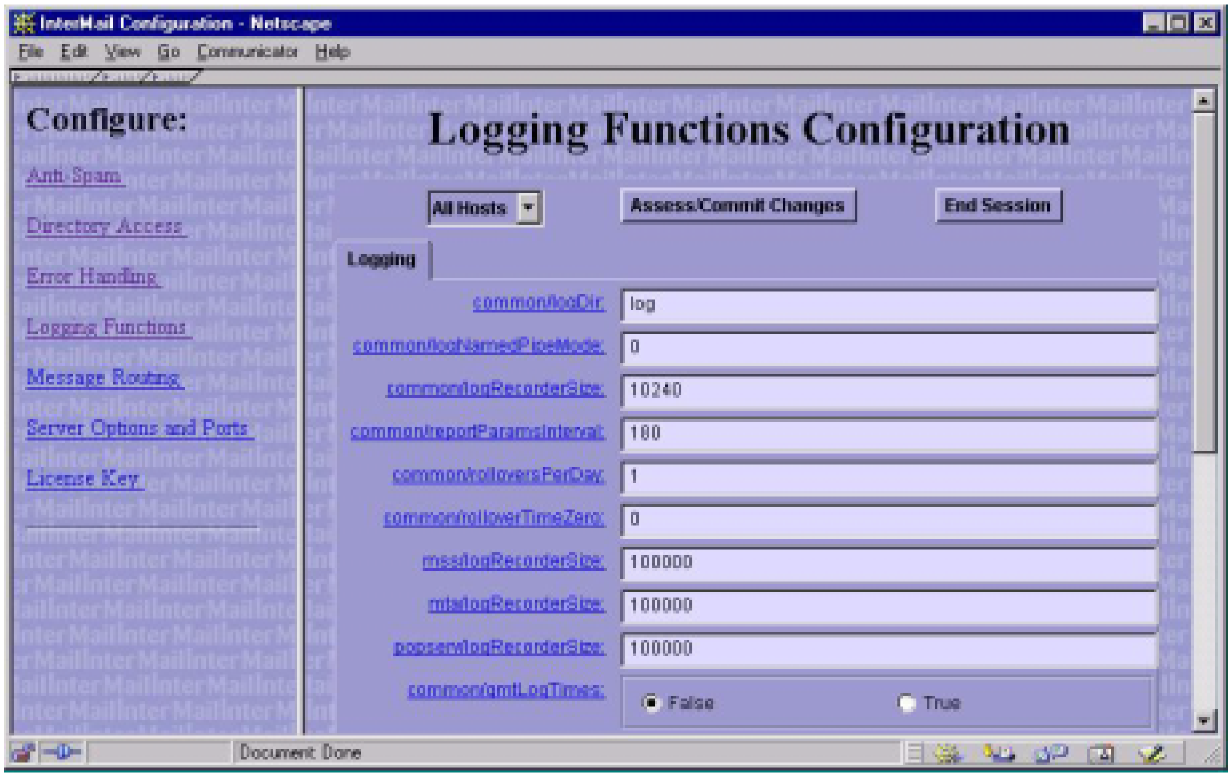


Obtaining most up to date copy of config.db ...

Figure 18 Status window

Three pop-up status windows occur in succession: **Current copy of config.db is fetched**, **Loading config.db values...**, and **config.db values loaded.**

After the status messages have been displayed, the desired page is loaded. For example, if you selected Logging Functions, the screen shown in Figure 19 would be displayed.



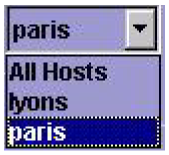
common/logDir:	log
common/logNamedPipeMode:	0
common/logRecorderSize:	10240
common/reportParamsInterval:	180
common/rolloversPerDay:	1
common/rolloverTimeZero:	0
mss/logRecorderSize:	100000
mta/logRecorderSize:	100000
popserv/logRecorderSize:	100000
common/gmtLogTimes:	[✓ False] True

Figure 19 Logging Functions Configuration Page

Making and Committing Configuration Changes

To edit the value of a configuration key:

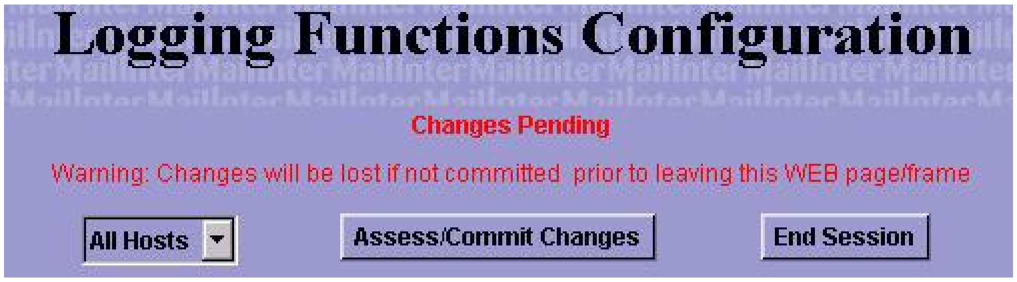
1. Determine if the changes you wish to make will affect all hosts or only one of the hosts in the system. By default, changes apply to all hosts in the system, but if you wish to make a host-specific change, select the host pull-down menu and choose a host. For example:



2. Modify the individual configuration keys listed on the page, making sure that you are editing the correct key on the correct server. If you enter an invalid value, an error message is displayed.

Note: For online help on the key that you are editing and valid values for it, click the keyname.

When you modify a value, the text in Figure 20 is displayed to warn you against accidentally clicking the Back button in your browser or navigating to another page in the Configuration editor. If you perform either of these actions during the editing process, you will lose your changes.


Logging Functions Configuration

Changes Pending

Warning: Changes will be lost if not committed prior to leaving this WEB page/frame

Figure 20 Warning text

3. When you have finished making on the page, click the Assess/Commit Changes button at the top of the page.
4. A report is displayed about the impact of the prospective changes on the affected servers. Depending on the contents of the report, commit the changes or cancel them.
5. When you have finished all edits, click the **End Session** button.

Propagating Configuration Changes

When you commit changes to the master Configuration database, each InterMail server is alerted and automatically retrieves a copy of this updated database from the Configuration server. No manual intervention is necessary to propagate the new Configuration database throughout the system.

However, this does not necessarily mean that all changes take effect immediately. If a configuration change requires restarting a server, you must shut down and restart the server before the change will take effect.

Note: After making configuration changes, you should monitor your system carefully to make sure that your changes result the behavior you expect.

Using the imconfedit Utility

The Configuration Editor gives you access to most common configuration keys; in contrast, the `imconfedit` utility allows you to edit keys that are not present in the Configuration Editor and to make configuration changes when you do not have access to a Web browser. The `imconfedit` also allows you to view the Configuration database in read-only format.

When you make changes to the Configuration database, `imconfedit` also displays a list of the prospective changes and reports the impact of the changes on InterMail servers.

Modifying Configuration database information with `imconfedit` involves three basic steps:

1. Executing `imconfedit` to display the current Configuration database.
2. Using a text editor to enter new key values and entries.
3. Committing or canceling your changes, depending on the change impact information reported by individual InterMail servers.

This section describes these steps in detail.

Running imconfedit

To run `imconfedit`, simply type the command name at the system prompt:

```
imconfedit
```

The utility begins by contacting 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 the system’s default text editor (or `vi`, if there is no default editor). The database appears as a long series of entries, each with a host, server, key name, and associated values:

```
/*/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.

Making Configuration Changes

Before making changes to the Configuration database, be sure you have at your disposal the comprehensive list of configuration keys presented in the *InterMail Kx Reference Guide*. This list includes the name, possible values, and other important characteristics of every configuration key. Access to this information is essential.

You can make configuration changes in two ways: by modifying the value of an existing keys (this is the more common method), and by creating new configuration key entries. Although the first of these is more common, you should understand the requirements and potential pitfalls of each.

Whichever kind of change you are making, start by searching for each instance of the key that already exists in the Configuration database (using the text editor’s search facility is typically the best method). A single key may appear multiple times in the Configuration database, at different levels of the hierarchy; to avoid modifying the wrong entry or causing unintended configuration errors, you should search for all occurrences of a particular key before modifying any single value.

Be particularly careful when adding a new Configuration database entry, since a new configuration key entry may well override a value set for the same key at a higher level in the configuration hierarchy, or be overridden by an entry lower in the hierarchy.

All configuration key values appear in `config.db` between square brackets ([]). To change the value of a configuration key, simply delete the value of the key inside the brackets, and enter a new value. All configuration keys have a set range of possible values, so be aware of the data requirements of a key before changing its value.

Editing Guidelines

When editing the Configuration database, keep the following guidelines in mind:

- If you want a key to assume its default value, it is recommended that you enter this value for the key under `/*/common`. Although you can simply delete the key from the Configuration database, deleted keys will not appear in subsequent `imconfedit` operations.

- Never edit configuration keys for the configuration path `/<host>/sysadmin` (that is, `*_opt` and `*_run` keys). These keys are set and used by the system and 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 and Propagating Configuration Changes

When you close the `config.db` file, `imconfedit` displays a review of the configuration changes that you have made, and prompts you 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] ?
```

If you are not satisfied with your changes, type `R` at this prompt to return to the text editor for further editing, or `Q` to cancel all changes and terminate `imconfedit`.

If you are satisfied and want to commit your changes to the Configuration database, type `P` at this prompt. This causes `imconfedit` to query all InterMail servers about the impact of these changes on their operations. The results of this query appear as follows:

```
"
*                              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] ?
```

As before, enter `R` at this prompt if you are not satisfied with your changes and want to re-edit them, enter `Q` to cancel all changes and terminate `imconfedit`, or enter `P` to commit the changes and propagate them to all InterMail hosts.

If you commit your configuration changes, `imconfedit` reports the outcome of the changes. If the changes require that one or more servers be restarted, `imconfedit` also prompts you to restart them:

```
---- 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 best performed 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, execute `imconfedit` with the `-viewonly` flag:

```
imconfedit -viewonly
```

When `imconfedit` runs with this flag, it retrieves a copy of the Configuration database and opens it in the system's default text editor (or `vi`, if there is no default editor). You can then use the editor's search functions to locate the values of individual configuration keys.

8
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 with the potential of abusing 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

Security Features Overview

The InterMail system has a number of features designed to restrict use of the system by those who attempt to appropriate its resources to transmit messages where the InterMail system itself is neither the source nor final destination of such messages. This type of transmission is called message relay. InterMail allows administrators 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 remainder of this chapter discusses these features in detail:

Feature	**Function**
Connection dropping	Used to immediately terminate connections from specific hosts. See “Connection Dropping” on page 97 for details.
Mail blocking	Used to prevent specific users or systems from sending any mail to an InterMail site. See “Mail Blocking” on page 99 for details.
SMTP authentication	Used to prevent individuals from forging addresses (a tactic often used by junk e-mailers to hide their true identity). See “SMTP Authentication” on page 105 for details.
Relay prevention	Used to control how senders may use InterMail as a relay point for messages. See “Relay Prevention” on page 106 for details.
Message sidelining	Used to place suspected “junk mail” into a holding directory for later evaluation. See “Message Sidelining” on page 114 for details.
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. See “Mail Filters” on page 116 for details.
Password protection	Used to prevent individuals from retrieving mail for accounts other than their own by discovering account passwords. See “Password Protection” on page 125 for details.
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. See “LDAP and RME Port Protection” on page 128 for details.
Secure socket layer (SSL) authentication	Used to perform a special type of encrypted authentication for message transactions. See “Secure Socket Layer (SSL) Authentication” on page 128 for details.
Blocking RCPT TO: harvesting	Provides a mechanism to dynamically detect and block `RCPT TO:` harvesters. See “Blocking RCPT TO: Harvesting” on page 131. for details.

Connection Dropping

Connection dropping is a tool for blocking unwanted mail. This feature allows the Message Transfer 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.

The InterMail connection dropping features also allow the MTA to terminate a connection if the client is sending a single message to more than a specified 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 114.

Configuration Options

The following configuration keys control connection dropping options:

`dropConnections`	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 to 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.
`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.`”

`dropTheseIPs`	A list of IP addresses of systems to drop. When the value of `dropConnections` is `true`, the system compares the IP address of each connecting client to 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.

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:

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 would not block legitimate mail.
2. Execute the `imconfedit` administrative command to edit the Configuration database (as described in Chapter 7).
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.

Carefully monitor the logs of each MTA to determine the effects of the new policy. 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” on page 99 are the best features for combating the distribution of junk e-mail. 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 it is assumed that the sender is 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]
   ```

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 any e-mail to your site. Unlike relay restrictions (described in “Security Features Overview” on page 95), 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

The InterMail system 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: There is an important difference between mail blocking and connection dropping. Because connection dropping occurs the moment 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 97), 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 messages from 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, before the `@` character). This allows you to block mail from users who send mail from multiple domains using the same username.

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

Blocking Modes

The InterMail mail-blocking feature can operate in either of 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. With 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 administrators 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. Per-account mail blocking is among the InterMail class-of-service options discussed in Chapter 4.

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 an account-by-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 block. These blocking methods are independent of one another, and you can use as many or as few as you like.

Blocking Mode

A single configuration key defines the mode of mail blocking:

`blockPerAccount`	Key that determines whether the blocking policy is system-wide (with blocking policies applying to all incoming messages), or per-account (with blocking policies applying 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`, the setting 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`, it 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`, it compares the IP address of a connected client to 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:

`blockTheseUsers`	A list of usernames to be blocked (for example, `make-money-fast`).
`blockUsers`	Key to enable (or disable) mail blocking by sender username. When this key is `true`, it 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.

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 return 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 use 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:

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 would not reject legitimate mail.
2. Execute the `imconfedit` administrative command to edit the Configuration database (as described in Chapter 4).
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 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.

Carefully monitor your MTA to determine the effects of the new blocking policy.

Warning! 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 problem for Internet service providers (ISPs) is the problem of users who send e-mail from non-existent addresses within the ISP's local domains. For example, an end 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, locate the key `/*/mta/blockAddresses`. If the value of 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:

```
/paris/mta/blockAddresses: [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 his 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. See Chapter 6 for a discussion of enabling SMTP authentication at the class-of-service level.

- **Authenticated SMTP**—This option specifies that the user must provide authentication information when sending mail using SMTP. With this option enabled, the MTA accepts a submitted 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 on the `From:` header line, to see if the SMTP authentication is set for the sender's account in the Integrated Services Directory (ISD). 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.

Mail Relaying Basics

Mail relay occurs whenever 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. When 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 relay; in fact, relay 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 function by using mail servers at other sites to relay huge volumes of junk e-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 e-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 e-mail sender who knows about its vulnerability. In contrast, preventing any relay means that users on an InterMail system can send mail only to other users on the same system.

These are extreme examples at both ends 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

A very 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, accepting 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 final destination of the messages that it receives. Technically, this MTA performs third-party relaying to the MTA inside the firewall.

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 can 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 mostly open relay, with restrictions that prevent relay only for specific hosts and users. This is the most commonly used relay policy.
- **Completely restricted**—This policy restricts all relay, causing the MTA to accept messages to addresses only in local mail domains, or in other specific domains.

- **Restricted with exceptions**—This policy allows you to maintain a mostly closed system, with relay allowed only under certain circumstances.

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

Restricted Relay

Restricting relay does not necessarily mean preventing relay. In InterMail terminology, a “restricted” message is a candidate for denial, perhaps because it is suspected to be junk e-mail. 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 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, the InterMail system 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 an InterMail system 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 addresses) 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 ISD 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 administrators 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 to recipients only 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 can have before relay checks are performed. This key allows you to exempt messages from your anti-relay policies if they are addressed to only a few recipients. For example, if this value is set to 3 and a user attempts to relay a message addressed to two 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`), or restrict all relay (`denyAll`). When defining relay restrictions, you should always start by setting the value of this key, or by verifying its value.

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 will relay 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 allowing relay. 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` applies 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 is 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 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 the address `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 restricted by relay source policies:

`relayDestAllowList`	A list of domains to which messages can be relayed, regardless of restrictions on the source of the messages. No unlisted domains 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. All unlisted 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 defined in `relayDestDenyList`.

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 return 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 used for 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, 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 restricted mail is addressed to accounts within your local mail domains, the system allows delivery of this mail. This allows normal delivery of messages addressed to your users from 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: After modifying these configuration keys, the MTA does not have to be restarted 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: [minorcorp.com]
            [*.majorcorp.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 `majorcorp.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 administrators 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 administrators 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, the message 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 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 then inspect sidelined messages 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 periodically review and handle the sidelined messages.

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.A value of zero (the default value) disables this feature.
`sidelineNumRcptsPerConnection`	Defines the maximum number of recipients allowed over a given connection before message sidelining begins. A value of zero (the default value) means that there is no limit.

Viewing Sidelined Mail

When sidelining is 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, see the discussion of message files in Chapter 10.

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 unavailable.

Use of the `Ok-To-Sideline` header is important for two reasons. First, it uses 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 resubmitted for processing with 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

The InterMail system 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 the mail’s content, the filters can specify that a message should be rejected, bounced, sidelined, forwarded, deleted, or delivered normally.

Filter actions can depend on the presence or absence of certain headers, the sender address, the recipient address, or any other text contained in the headers or body of the message. You can search message bodies 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 Kx 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 118 for more information.
`action`	Defines an action taken against the message by the MTA. See "Actions" on page 121 for more information on the values of this parameter.

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 false.
`TRUE`	Is always true.
`FALSE`	Is always false.

Note: Filter keywords are not case-sensitive; they appear in this section in capital letters for readability only.

For 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 according to 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`—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 measurement. Because the `SIZE` element is in bytes, specifying a unit of measure with a number value provides a convenient method to specify kilobytes, megabytes, or gigabytes. The possible values are: `Kkilobytes`—Multiplies the value by 1,024 (e.g., `1K`, `2K`,and so on) `Mmegabytes`—Multiplies the value by 1,048,576 (`1M`, `2M`, and so on) `Ggigabytes`—Multiplies the value by 1,073,741,824 (`1G`, `2G`, and so on)

For 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.DOMAINQ`: The domain of the sender address `SENDER.LOCAL-PAR`: The username of the sender address `RECIPIENTS`: All recipients of the message `BODY`: Message body, including attachments `BODY.TOP(#)`: The first # 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 indicates a case-insensitive query. For example:
	`CONTAINS-NOCASE`
	`IS-NOCASE`

`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`, this value is a character pattern that can include the wildcards `*` and `?` to find partial matches. In the case of `CONTAINS-RE`, this value evaluated as a regular expression.

For 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 is 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:

`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 just like `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 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>`

`DISCARD`	Deletes the message.
`KEEP`	Delivers the message normally.
`FORWARD "<address>"`	Forwards the message to a specific address. A filter can include multiple `FORWARD` actions. If it specifies `FORWARD` along with `KEEP`, it both forwards and delivers the mail normally.
`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.

`SIDELINE "<reason>"`	Sidelines the message. For information on sidelined mail, see “Message Sidelining” on page 114.
`STOP`	Stops execution immediately. If no action has run, the InterMail system delivers the message normally. Otherwise, the filter completes the previously executed actions.
`TOSS`	Deletes the message.

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 sample 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 would be 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 mail filters appear in the Configuration database, adding new ones 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> <bodyfile> |
                -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 a configuration key that contains the filter to be validated. The value of this parameter is typically `incomingMailFilter`.
`message-file`	Specifies a 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
  19970114235158898.AAA250@oslo.software.com-Header
  19970114235158898.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.

Password Protection

Another 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.

The InterMail system allows you to combat this type of improper mail access with a number of options on both the POP and IMAP servers that deter users from guessing passwords:

- **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. This allows the system to enforce increasing password delays on single users who attempt to guess the passwords of multiple accounts during one transaction.
- **Logging of password failures**—Several logging options allow for recording of 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, the InterMail system 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:

$$\left(\begin{array}{c}\text{1 failure on John Doe's account}\\ +\\ \text{1 failure from [29.82.172.24]}\end{array}\right) * \text{5 seconds} = \text{10 seconds}$$

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

$$\left(\begin{array}{c}\text{2 failures on John Doe's account}\\ +\\ \text{2 failures from }\texttt{[29.82.172.24]}\end{array}\right) * \text{5 seconds} = \text{20 seconds}$$

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:

$$\left(\begin{array}{c}\text{1 failure on Susie Queue's account}\\ +\\ \text{3 failures from }\texttt{[29.82.172.24]}\end{array}\right) * \text{5 seconds} = \text{20 seconds}$$

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 of 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 for notification of a failed authentication attempt. With each failed attempt, this number of seconds is added to the delay time, up to the limit on delay time 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 is entered. 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 communication is dropped.
`maxBadPasswordAddrs`	The maximum size of a list of IP addresses of clients that have issued incorrect login information. Each time an authentication fails, the IP address of the associated client is added to this 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 the subsequent delay for failed authorization attempts.
`maxPasswordFailures`	The maximum number of authentication failures allowed for an account in the time specified by the `badPasswordWindow` key, before an `AcctBadPswdMaxFailures` log event is generated in the POP or IMAP server logs.
`maxBadPasswordUsers`	The maximum size of a list of accounts that are associated with incorrect passwords. Each time an authentication attempt fails, 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.

LDAP and RME Port Protection

The InterMail system uses LDAP and RME for its implementation of the MSS and Directory server, InterMail contains LDAP and RME ports. In order to protect these ports from being denial-of-service attack points, InterMail allows you to specify that only certain IP addresses and netmasks are allowed to connect to the LDAP and RME ports. You use the `ldapAccessList` and `rmeAccessList` configuration keys to set up this firewall-like protection:

`ldapAccessList`	A list of IP addresses or IP masks that are allowed to connect to the LDAP 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 allowed only from the local machine.
`rmeAccessList`	A list of IP addresses or IP masks that are allowed to connect to RME ports. 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 will be the existing `NioConnNotAllowed` message, which specifies the IP address of the host attempting to connect, along with the port type.

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 Figure 21.

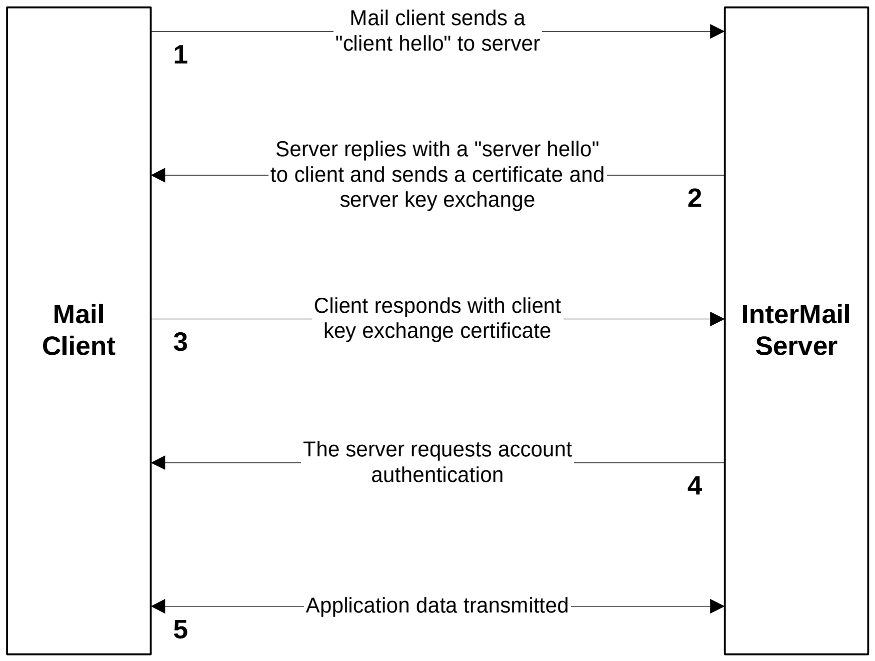

Mail Client
InterMail Server
Mail client sends a "client hello" to server
1
Server replies with a "server hello" to client and sends a certificate and server key exchange
2
Client responds with client key exchange certificate
3
The server requests account authentication
4
Application data transmitted
5

Figure 21 SSL data flow

1. An SSL-enabled e-mail client contacts the InterMail server on an alternate POP/IMAP port specifically used for SSL transactions.
2. The server 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:

1. Use the `sslPop3Port`, `sslIMAPPort`,and `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.
2. Use the `sslTrustedCertPathAndFile` configuration key to specify the file containing the certificate. As shown in Figure 21, the SSL process involves

certificate exchange by client and server. Therefore, a certificate must exist in the InterMail system.

3. 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.
4. 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 the InterMail server, 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.

Several conditions must be satisfied in order for enable TLS to be enabled in an SMTP session:

- The connected client must be SSL-compliant.
- The negotiation of data shown in SSL data flow must occur successfully.
- 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, then 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 this 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.

Blocking RCPT TO: Harvesting

One of the tools used by senders of junk e-mail is a program that guesses valid e-mail addresses. Using this tool, a spammer can do connect to your MTA and run `RCPT TO: <username>` commands. If the MTA accepts an address, the spammer can then save or “harvest” that address for later spamming.

InterMail provides a mechanism to allow you to track IP addresses that are the source of bad `RCPT TO:` commands, on a per-MTA basis. Using this feature, you can dynamically detect and block `RCPT TO:` harvesting.

To use this feature, set the `trackRcptHarvesters` and `/*/mta/verifyRCPTs` configuration keys to `true`.

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

- The number of bad `RCPT TO:` commands from that IP address exceeds the number specified by the `rcptHarvesterCount` configuration key.

- This number of bad commands occurs within the span of time specified by 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 by the `rcptHarvesterTTLMinutes` configuration key.

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

Note: Set the `rcptHarvesterCount` and `rcptPotentialHarvesterTTLMinutes` configuration keys high enough that you can identify true harvesting when it occurs and that you do not block legitimate mail.

Configuration Options

The following configuration keys define this feature:

`rcptHarvesterCount`	Indicates the number of bad `RCPT TO:` commands from an IP address within a specified time that will cause that IP address to be labeled as a `RCPT TO:` harvester.
`rcptHarvesterTTLMinutes`	Specifies the number of minutes for which connections should be blocked from an IP address 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 address is to be tracked as a potential harvester after one bad `RCPT TO:` is received from that IP address.
`trackRcptHarvesters`	Indicates whether `RCPT TO:` harvesters should be tracked and blocked. For this key to have an effect, the `/*/mta/verifyRCPTs` configuration key must also be `true`.

9

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 Kx 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 if `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 always tried 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 135.

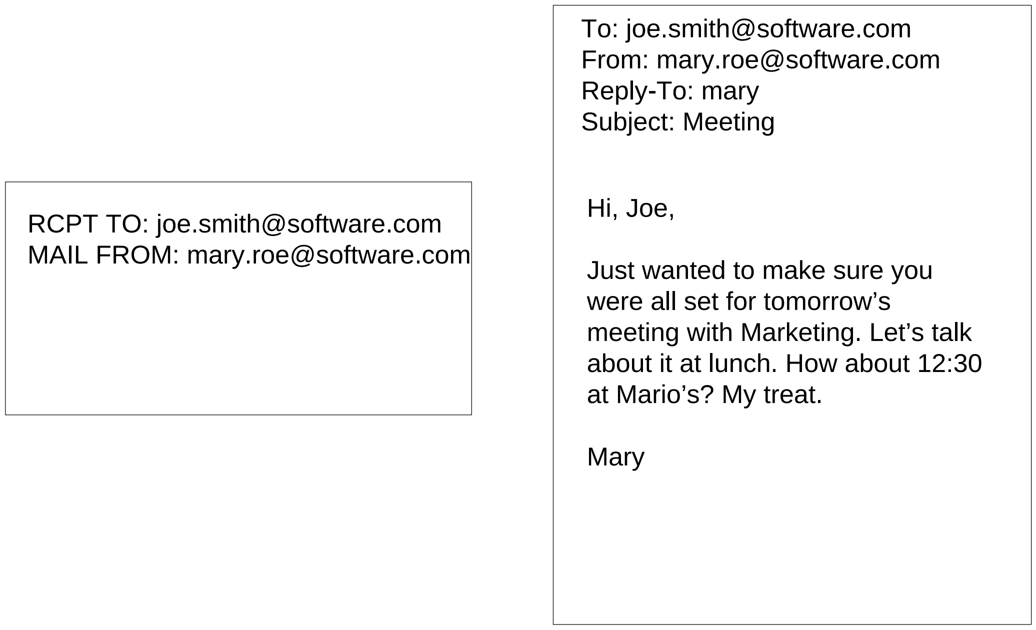
An InterMail message is similar to postal mail in that it consists of both an envelope and the actual message. Both envelopes and messages contain addresses, but their functions are very different.

In the envelope, there are only types of two "addresses": the `RCPT TO:` addresses, which are 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; however, the person reading a 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 22 shows how the addresses and headers on an e-mail envelope and message might look if they were postal mail.


RCPT TO: joe.smith@software.com
MAIL FROM: mary.roe@software.com

To: joe.smith@software.com
From: mary.roe@software.com
Reply-To: mary
Subject: Meeting

Hi, Joe,

Just wanted to make sure you were all set for tomorrow's meeting with Marketing. Let's talk about it at lunch. How about 12:30 at Mario's? My treat.

Mary

Figure 22 Envelope addresses and message headers

Address Completion

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

- Addresses that contain no domain 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.
- 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 username and hostname.

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.

If the value for `defaultDomain` is defined as `minorcorp.com`, 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 user name 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 `majorcorp.org`, respectively, InterMail completes John Doe’s address as follows:

`john.doe@london.majorcorp.org`.

Of course, there is no guarantee that this address exists or that mail for John Doe will arrive at this address. InterMail uses this as a backup method for address completion, just in case no `defaultDomain` is specified. In most cases, however, there will always be a valid setting for `defaultDomain`.

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 `majorcorp.org`, a message addressed to `mary.roe@rome` automatically changes to:

```
mary.roe@rome.majorcorp.org
```

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 4.

  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.

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 135. 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 a 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]
```

For 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.

  For example, suppose a message for Mary Roe in the `majorcorp.org` domain arrives addressed to her alias address:

  ```
  mary@majorcorp.org
  ```

  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@majorcorp.org
  ```

- Rules specified in a rewrite domain. This method can be used only if there is an applicable rewrite domain in the ISD.

  For 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 `majorcorp.org` 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@majorcorp.org
```

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 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.

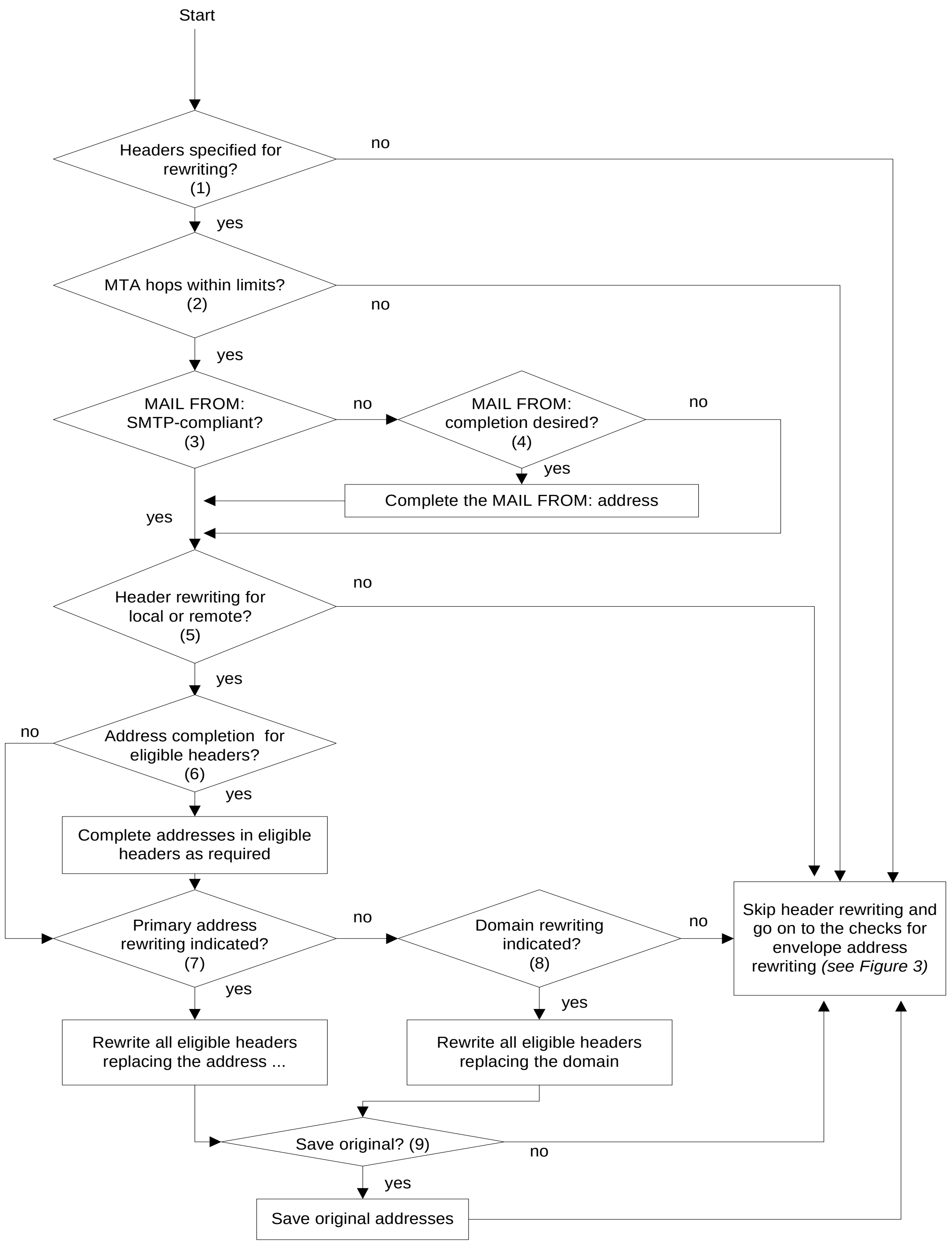

Start
Headers specified for rewriting? (1)
no
yes
MTA hops within limits? (2)
no
yes
MAIL FROM: SMTP-compliant? (3)
no
MAIL FROM: completion desired? (4)
no
yes
Complete the MAIL FROM: address
yes
Header rewriting for local or remote? (5)
no
yes
no
Address completion for eligible headers? (6)
yes
Complete addresses in eligible headers as required
Primary address rewriting indicated? (7)
no
Domain rewriting indicated? (8)
no
Skip header rewriting and go on to the checks for envelope address rewriting *(see Figure 3)*
yes
yes
Rewrite all eligible headers replacing the address ...
Rewrite all eligible headers replacing the domain
Save original? (9)
no
yes
Save original addresses

Figure 23 Header rewriting for incoming mail

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 24).
6. Checks whether addresses in eligible headers (those listed in the `rewriteHeaderList` key) require completion.

- If the addresses in eligible headers are complete, goes on 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 on 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 4.

Figure 24 shows the flow of the domain rewriting process, which starts with Step 10, after the checks for header rewriting.

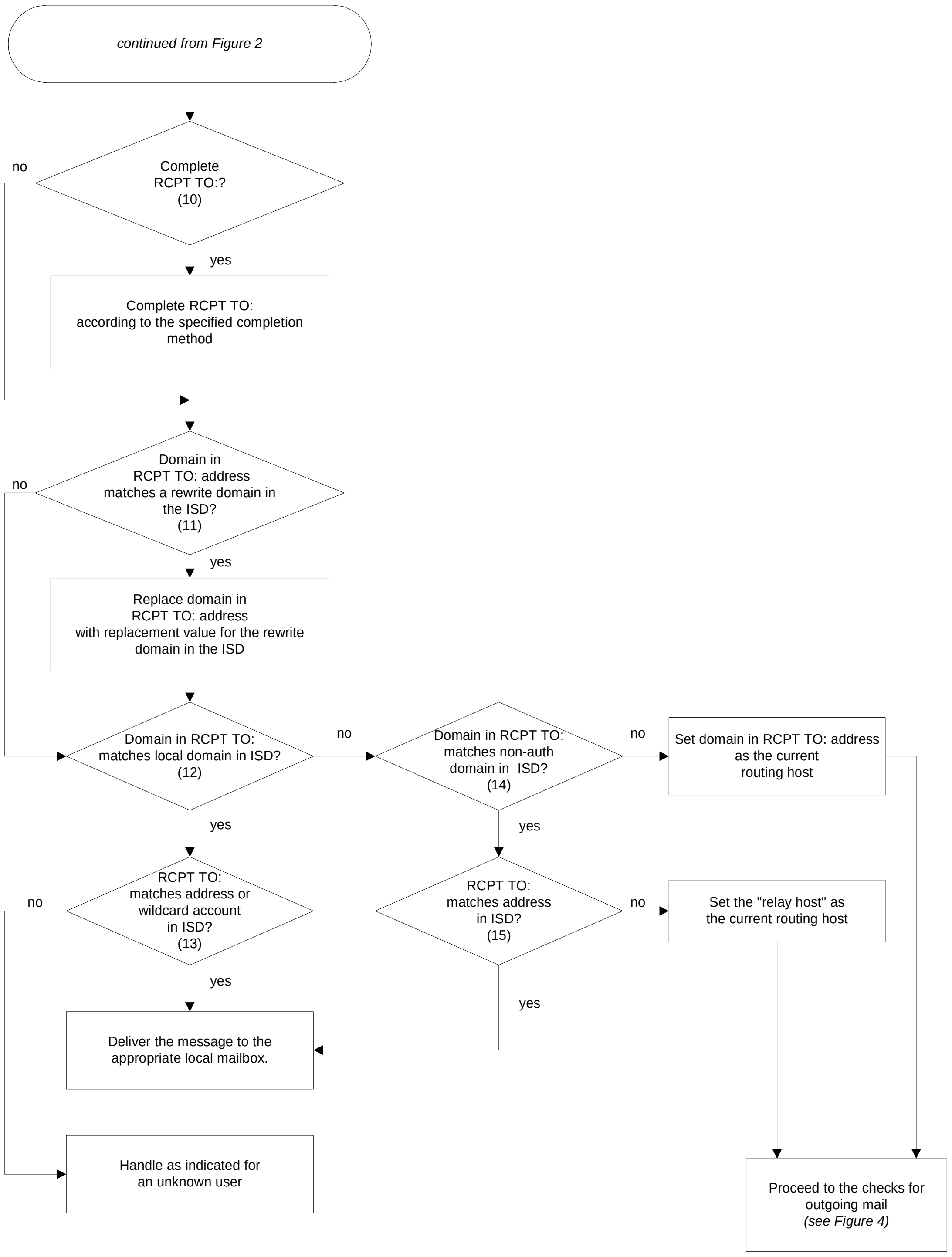

continued from Figure 2
Complete RCPT TO:? (10)
no
yes
Complete RCPT TO: according to the specified completion method
Domain in RCPT TO: address matches a rewrite domain in the ISD? (11)
no
yes
Replace domain in RCPT TO: address with replacement value for the rewrite domain in the ISD
Domain in RCPT TO: matches local domain in ISD? (12)
no
yes
Domain in RCPT TO: matches non-auth domain in ISD? (14)
no
yes
Set domain in RCPT TO: address as the current routing host
RCPT TO: matches address or wildcard account in ISD? (13)
no
yes
RCPT TO: matches address in ISD? (15)
no
yes
Set the "relay host" as the current routing host
Deliver the message to the appropriate local mailbox.
Handle as indicated for an unknown user
Proceed to the checks for outgoing mail *(see Figure 4)*

Figure 24 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 `RCPT TO:` 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 155.

  - 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 25 in “Rerouting and Rewriting Process Flow” on page 155).

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 25).

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 using the MX records in DNS or by delivering a message to the hostname specified in the envelope address. 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 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>` and `<new.rtg.host>` are hosts as defined for a `RCPT TO:` route in RFC 821. They may actually be mail domains such as `software.com`.

Note: If the `RCPT TO:` address of a message to be relayed contains routing information before the mailbox name, that routing information, not the routing host element in the mail routing table, determines the address to which the message is relayed.

For example, if the MTA received a message for relay with a `RCPT TO:` address of `@ONE,@TWO:JOE@THREE` (where `ONE`, `TWO`, and `THREE` are "hosts", or mail domains), 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 routine host—or mail domain—to which the outgoing message usually goes), or "default". If "default" is specified, all outgoing messages that have not matched an entry in `mailRoutineTable` prior to this entry will be routed to `<new.rtg.host>`. You should specify "Default" only if you want all outgoing mail to 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 the 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.

For example, the following entry specifies that any message that the system would normally deliver to the host `rome.majorcorp.org` will instead go to a machine called `gateway.com`:

```
/*/mta/mailRoutingTable: [rome.majorcorp.org: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.majorcorp.org:gateway.com]
[*.majorcorp.org: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.majorcorp.org`. 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>.majorcorp.org` to `internal_server.com`. Thus, for example, mail sent to `venice.majorcorp.org` or `paris.majorcorp.org` will go to `internal_server.com`.

> ***Note:*** This entry would not reroute mail sent to just `majorcorp.org`; the “`*.`” portion of the entry means that some host name must appear before `majorcorp.org`.

- 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.majorcorp.org` to `gateway.com`.
2. Next, reroute messages destined for any other named `majorcorp.org` host (that is, `*.majorcorp.org`) 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@majorcorp.org` (the routing host portion being `majorcorp.org`). 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.majorcorp.org`, 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@majorcorp.org` to substitute for the wildcard (`*`) in `*.majorcorp.org`—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@majorcorp.org` to the `firewall.com` host machine.

In contrast, suppose that the order of the entries in the table were changed:

```
/*/mta/mailRoutingTable: [*.majorcorp.org:internal_server.com]
[rome.majorcorp.org:gateway.com][default:firewall.com]
```

In this case, messages destined for `rome.majorcorp.org` will never be rerouted to `gateway.com`. Because `*.majorcorp.org` is the first entry in the table, the system reads that entry first, and all messages with matches for `*.majorcorp.org`, including `rome.majorcorp.org`, 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.majorcorp.org` to `gateway.com`, you would enter the following line in the mail routing table:

```
/*/mta/mailRoutingTable: [rome.majorcorp.org: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]
```

For 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.majorcorp.org` to `gateway.com`, you would enter the following line in the mail routing table:

```
/*/mta/mailRoutingTable: [rome.majorcorp.org:gateway.com
rewrite-domain=majorcorp.org]
```

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=majorcorp.org]
```

Rerouting and Rewriting Example

Suppose that you want to reroute all outbound messages for the `minorcorp.com` host to the `majorcorp.org` 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:majorcorp.org header-rewrite
   rewrite-domain=majorcorp.org]
   ```

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 25 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.

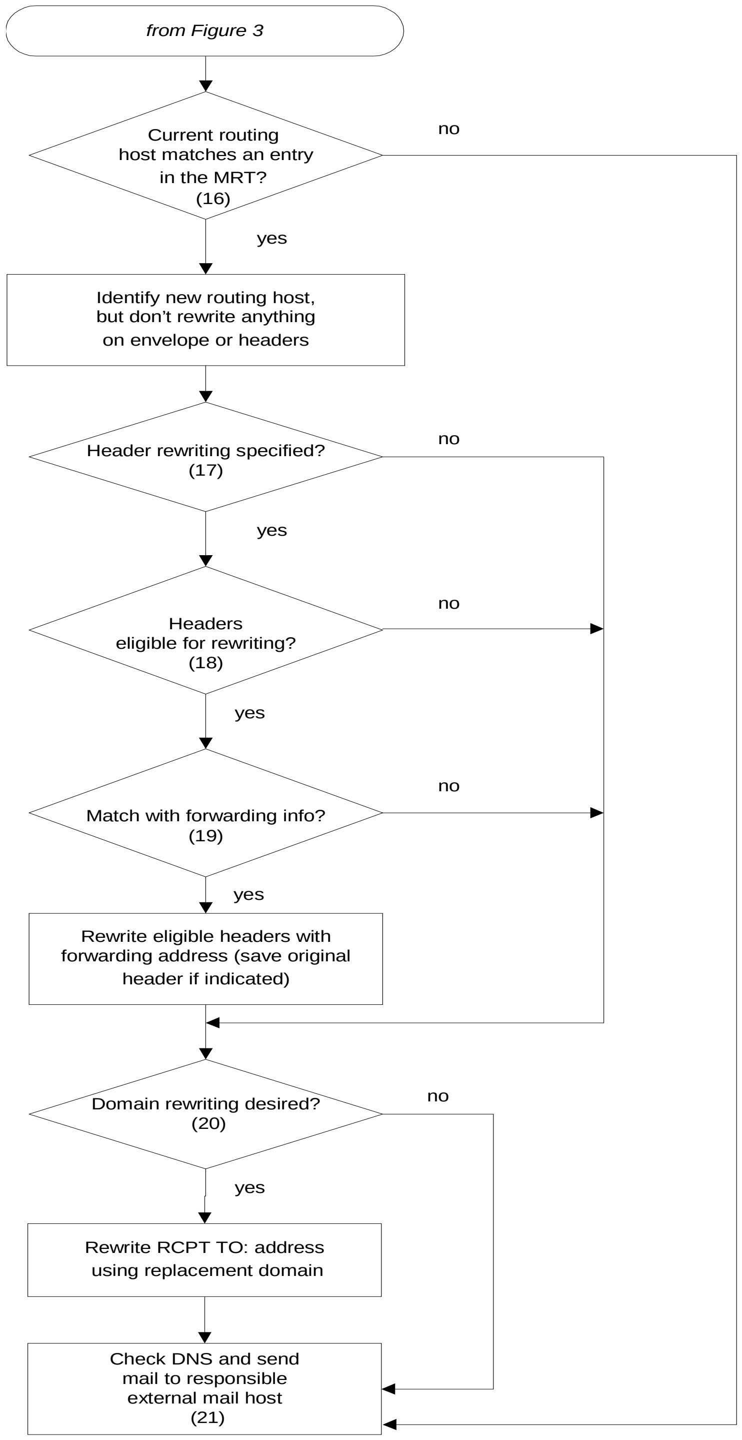

from Figure 3
Current routing host matches an entry in the MRT? (16)
no
yes
Identify new routing host, but don't rewrite anything on envelope or headers
Header rewriting specified? (17)
no
yes
Headers eligible for rewriting? (18)
no
yes
Match with forwarding info? (19)
no
yes
Rewrite eligible headers with forwarding address (save original header if indicated)
Domain rewriting desired? (20)
no
yes
Rewrite RCPT TO: address using replacement domain
Check DNS and send mail to responsible external mail host (21)

Figure 25 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 table 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, the system 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.

For 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/ mtaMesageDelivered`
Has been deferred and queued longer than the configured limit	`Error-Actions/ mtaMessageQueuedTooLong`
Reached its desintation 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`

10
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 vs. 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 11.

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, a significant percentage of incoming mail is processed entirely in memory. However, some circumstances require that mail be stored temporarily.

Mail in process may be categorized as follows:

- **Temporarily stored mail**, for messages that exceed configured limits on size, number of recipients, or the amount of time required for delivery.
- **Local deferred mail**, used when:
  - The Message Store Server (MSS) or the Integrated Services Directory (ISD) is temporarily unavailable

  - A message is received for an account whose mailbox is over quota and the configuration key `mta/bounceOnQuotaFull` is set to `false`.
  - Throttling of mail delivery to the message stores is in effect.
- **Remote deferred mail**, used when aremote mail host is temporarily unavailable
- **Sidelined mail**, used when a message is 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**, used when:
  - Something in a message causes an error that prevents delivery.
  - The `mta/Error-Action` configuration key is set to `hold` causing the message to be treated as having an error.

This section discusses each of these categories in detail.

Temporarily Stored Mail

If a message is large or addressed to a large number of recipients, delivery of the message may take longer than usual.

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 during lengthy delivery processes).

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 (`timeoutServerDelivery` configuration key). 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 (`maxDirectKB` configuration key).
- The maximum number of recipients a message can have without being written to disk (`maxDirectDelivery` configuration key)

If a message exceeds any one of these limits, the system will save 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 should wait for message delivery before writing it 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 cause either slow processing on the MTA, or slow response from the MSS.

Note: The maximum recommended value for the `timeoutServerDelivery` configuration key is 5 seconds.

Maximum Message Size (KB)

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 will write any message with more than this number of recipients to disk before attempting delivery.

For example, if the value of `maxDirectDelivery` key is set to 20, the system saves to disk any message addressed to more than 20 recipients, although the actual delivery process continues from memory.

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 when the configuration key `/<host>/mta/bounceOnQuotaFull` is set to `false` can cause a local deferral of mail. Delivery is re-attempted at regular intervals as defined by the `/<host>/mta/deferProcessInterval` configuration key.

When a message is deferred for this reason, an `MsLimitTotalSize` warning log message is written to the `mta.log` file.

Remote Deferred Mail

When the remote mail server is unavailable, the InterMail system must temporarily store mail for delivery to 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 delivery attempts 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 configuration key is `180`, the MTA re-attempts delivery every 3 minutes (180 seconds).

Note: A separate configuration key (`/*/mta/outboundDeferProcessInitialwait`) controls elapsed time between the restarting of the MTA and the initial attempt at reprocessing of 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 suspicious. Sidelining can occur according to system policies, or according to 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 8.

Mail Held Due to Errors

Sometimes, a problem with a 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 in the `errors` directory. If you do not, the disk can fill up, causing delivery problems.

Mail-in-Process Files

This section discusses the format and directory structure of mail-in-process files.

Filename Format

The system stores mail in process differently from mail delivered to users. (For a discussion of permanent mail storage, see Chapter 11).

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 contains information about why the message has been stored to disk rather than delivered from memory.
- The **Header file** contains the header information for the message, including `To:`, `Cc:`, `Bcc:`, `From:`, `Sender:`, and `Reply-To:` addresses.
- The **Body file** contains the body of the message—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.

For example:

```
19980601171253.AHA6542.mta1@rome.minorcorp.com
```

Each of the three files bears this prefix, and 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
```

Directory Structure

The MTA generally stores mail in process in the `$INTERMAIL/spool` directory. You can configure this directory using the `/*/mta/mtaSpool` configuration key.

Figure 26 shows the file structure of in-process mail.

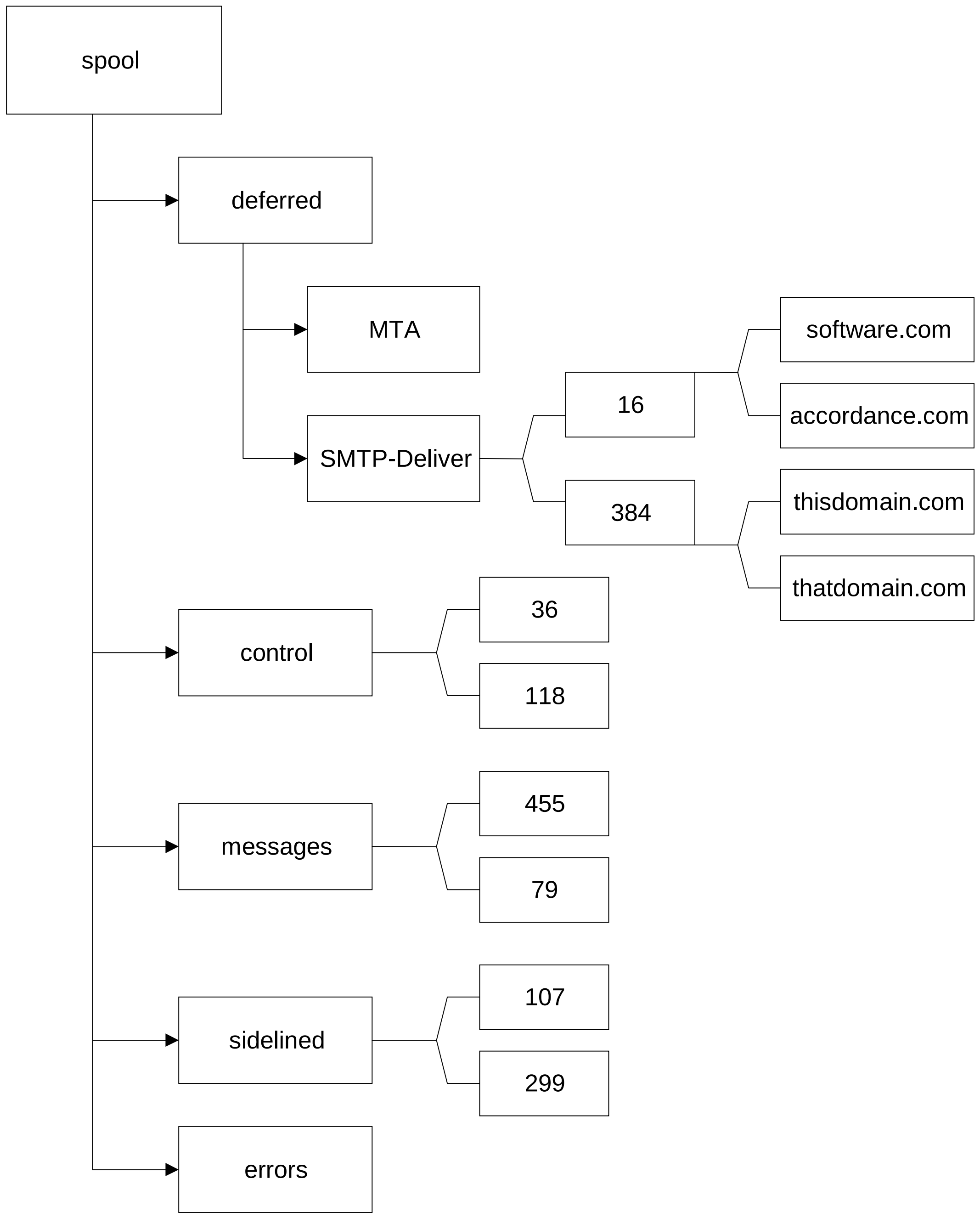


Figure 26 Structure of the MTA's spool directory

The numbered items in Figure 26 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 `spool` directory depends on the reason for their storage:

Reason for Deferral	**Control File Location**	**Header and Body File Location**
Temporary storage	In a bucket in the `spool/control` directory; for example, `spool/control/237`	In a bucket in the `spool/messages` directory; for example, `spool/messages/237`
Local deferred mail	`spool/deferred/mta`	`spool/messages`
Remote deferred mail	In a subdirectory, by domain, within a bucket in the `spool/deferred/SMTP-Deliver` directory; for example, `spool/deferred/SMTP-Deliver/345/minicorp.com`	In a bucket in the `spool/messages` directory
Sidelined mail	In a bucket in the `spool/sidelined` directory	In a bucket in the `spool/sidelined` directory (the same bucket as the Control file)
Mail held due to errors	`spool/errors`	`spool/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 8.

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 `spool/sidelined` directory, where the mail remains until you move or delete it.

Note: The system never deletes sidelined mail automatically; 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 decide what to do with it. If the message is actually junk mail, you will probably want to delete it. But if the message turns out to be legitimate, you will 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 can 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 files 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. For 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>
```

For example:

```
immsgprocess 19980114235158898.AAA250@paris.minorcorp.com-Control
```

The `immsgprocess` command moves the Control file of the message to the `spool/deferred` directory and the Header and Body files to the appropriate bucket in the `spool/messages` directory. The system then treats the message like ordinary deferred mail, and the MTA automatically attempts to deliver it to all of 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 `spool/errors` directory. To determine why a message was moved 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 you can fix the problem and want to have the message processed, you use the `immsgprocess` command. It is important to keep an eye on the `errors` directory because 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.

The InterMail system creates a separate queue directory for each remote domain to which it cannot deliver mail immediately. In very large 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 `thatdomain.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 `thatdomain.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 `thatdomain.com`. You can then create one new `thatdomain.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 thatdomain.com
```

Suppose this returns the following:

```
thatdomain.com preference = 10, mail exchanger = a.mx.thatdomain.com
thatdomain.com preference = 10, mail exchanger = b.mx.thatdomain.com
thatdomain.com preference = 10, mail exchanger = c.mx.thatdomain.com
thatdomain.com preference = 10, mail exchanger = d.mx.thatdomain.com
```

This tells you that there are four MX records for `thatdomain.com`, which means that you could have four queues—and four simultaneous connections—to `thatdomain.com`.

However, before you decide to split your existing `thatdomain.com` queue into four queues, consider the following:

- How much mail you have to send. If the amount is not large, four connections may be a waste of resources for both you and `thatdomain.com`.
- Whether `thatdomain.com` will permit you to make four simultaneous connections to their site.
- Whether `thatdomain.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 `thatdomain.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 thatdomain.com a.mx.thatdomain.com b.mx.thatdomain.com
c.mx.thatdomain.com
```

The `imqueuesplit` utility first creates the specified new queue directories. It then moves approximately 75% of the messages from the original `thatdomain.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 `thatdomain.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.

For 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

This section describes other options that affect mail queuing and reprocessing, including a variety of configuration options and the SMTP ETRN commands.

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 `spool` 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 end of 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 `thatdomain.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 `spool` 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 attempts 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. For example, if your MTA sends one message to `thatdomain.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 that affect mail deferred due to unavailable local or remote domains. For a more complete description, see the *InterMail Kx 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)

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.

For example:

```
220 london.minorcorp.com ESMTP server (InterMail v4.0 212) ready Tue,
12 May
1998 09:20:30 -0700
HELO
250 london.thatdomain.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 the limit for mail that can be in the process of delivery; 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.

11
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 store all messages addressed to local users until the users retrieve or delete them.

This chapter covers:

- Physical and logical message storage
- Creating and deleting mailboxes

Note: Temporary storage of mail in process is discussed in Chapter 10.

Message Storage

A typical InterMail installation 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. The 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 1 illustrates storage of a 5 Mb message addressed to local users John Doe and Susie Queue.

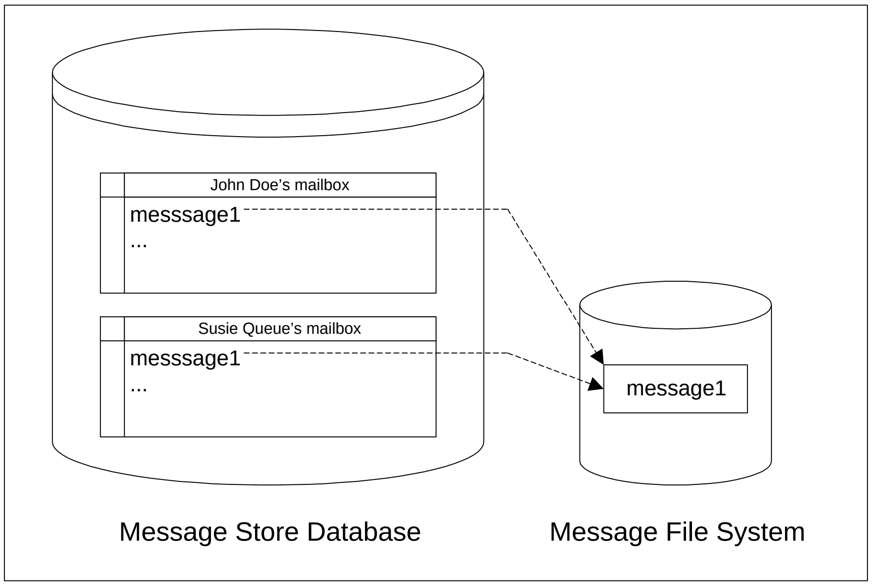


Figure 27 A single message stored for multiple users on the MSS

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 nested "bucket" directories. The lowest-level directories 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 bucket directories that contain the greatest message volume.

Note: Bucket directories are assessed according to the number of messages contained within them. The top third of these directories containing the most number of messages are excluded from the `buckets` listing.

This Message File system directory tree is created automatically during installation. If you need to create another directory structure later to accommodate greater message activity, use the `imbucketscreate` administrative command, which is described in the *InterMail Kx Reference Guide*.

Warning! 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 information about routine tasks needed to maintain the Message File system, such as expanding, balancing, and defragmenting, see Chapter 12.

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, as do files in the Message File system.

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 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 the 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 these 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).

Warning! 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 Kx 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 the 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 those 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 should be stored in its 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, and IMAP 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 Kx Reference Guide*.

The `imboxcreate` command creates a single mailbox in the Message Store database. The syntax of this command is:

```
imboxcreate [-help] <host> <internalID> [TotalBytes]
```

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 associated with the mailbox.

`TotalBytes`	Is the maximum number of bytes for this message store.

Note: To obtain the internal ID number of an account, use the `imdbcontrol` or `imaccountquery` administrative commands. For descriptions of these two commands, see the *InterMail Kx Reference Guide*.

For example, to create a mailbox on the host `paris` for the account whose internal ID number is `123456`, enter:

```
imboxcreate paris 123456 1000000
```

When it completes its operation, `imboxcreate` displays the results:

```
Message store 123456 created!
```

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> <mailboxID>
```

Where:

`-v`	Indicates verbose execution.
`-a <file>`	Is the input file containing e-mail addresses.
`-m <file>`	Is the 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.

For 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!
```

12
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 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. It also discusses monitoring tools that you can use to analyze and maintain an optimally performing InterMail system.

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 the following topics:

- System monitoring
- Performance monitoring and tuning
- InterMail performance tuning
- System maintenance
- Monitoring tools

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 will 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 within 3 seconds. If the banner does not appear quickly, 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 Message Transport Agent (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 Integrated Services Directory (ISD) 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 `log` 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 files 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.

For these files:	In this directory:	Do this:
Log files	`log`	Set log rollover to occur more frequently; physically move old log files to a separate file system or media source.
Sidelined mail	`spool/sidelined`	Manually inspect messages and reprocess them using the `immsgprocess` command.
Deferred mail	`spool/deferred`	Fix any undeliverable mail and reprocess this mail using the `immsgprocess` command.
Messages held in error	`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.
Orphaned message files	`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, 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 to 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 “Monitoring Tools” on page 199.

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`.

For information on configuring the logging levels for the `syslogd` daemon, see the `syslogd` man page.

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 contains `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. The areas that typically require monitoring 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 this example, 60% of swap space is used (`swap` column / (`swap` + `free` columns)) which is within reasonable parameters. If the total used swap space were at 75%, it would indicate a possible problem. If total used swap space were at 95%, this would indicate a definite problem.

Note: Analyze the swap partition to determine if it is large enough to handle the system's usual 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.

It is also important to monitor the memory scan rate since a high scan rate will negatively impact system performance. To bring the scan rate down, it may be necessary to add memory to the system or, if the system requires backing store for all stack-type data, to add swap space. For more information, see your system tuning guide.

Disk Performance

InterMail (particularly the MSS) 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
```

This output shows the number of reads per second (`r/s` column), writes per second (`w/s` column), and average service time. 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 time. 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 -o /var/tmp/disk.sar 10 100000

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 determine CPU performance based on the average number of requests that are outstanding (`avque`) and the average time to process a request (`avwait`). In general, a high `avque` time indicates a problem 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 percentage, 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. For example, you might:

- 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 whether the device has become fragmented and, if it has, run a utility to reduce fragmentation.
- Analyze the data that is assigned to the device, and determine whether some of the data can be moved to a less busy device.
- Add spindles to the device (assuming a virtual device, such as a Veritas file system). InterMail disk access tends to be random, and input and output occur in

relatively small blocks, so the more spindles with which the device has to work, the better.

- Check whether the device houses Oracle indexes. If it does, consider reorganizing the indexes to compress free space.

CPU Performance

To analyze the CPU load, issue the following command:

```
paris% sar -u -f /var/tmp/disk.sar 10 100000

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 way to determine CPU capacity is to run queue statistics using the `uptime` command. A high run queue level indicates that the system is experiencing a bottleneck somewhere and that 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 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 indicator of network health. In general:

- The `errs` value should be close to 0. Any substantial errors would point to a hardware problem with the associated interface.
- The collision rate ((`#coll`/`#outputPackets`)*100) should be under 5%. 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.

Performance Tuning

In InterMail, performance tuning is typically an interactive 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, 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 to 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`), try the following:

- Increase the value on 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 regularly back up critical components of the InterMail system. Backup efforts should focus primarily on message information stored in the Message File system and Message Store database and on 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 13 in this manual.

Expanding the Message File System

When the Message File system is running low on space, you can expand it by adding additional file systems.

To expand the Message File system:

1. Either mount an additional external storage device or symbolically link an existing partition, or add a subdirectory 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 better 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 Kx 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.

For example:

```
paris% cd msgfiles
paris% ls
messages        buckets      buckets.dir
paris% mkdir 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 Kx 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 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 up these files 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 the log files are to roll over, use the configuration key `rolloversPerDay`. For example:

```
/paris/common/rolloversPerDay: [2]
```

In this example, log files will 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), which defines offset from GMT. For example:

```
/paris/common/rolloverTimeZero: [18000]
```

In this example, the rollover period will begin at 00:05:00 hours GMT (18,000 seconds, or 5 hours, after 00:00:00 GMT).

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 the InterMail 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 elect to 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 Chapters 8 and 10.

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 variable `/*/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 this 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 `.ERRORS` 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`.

Monitoring Tools

InterMail provides several powerful monitoring tools that are used to track system health and performance:

- Simple Network Management Protocol (SNMP)
- `imsysmon` utility
- Event logging

The following sections describe the `imsysmon` utility and the logging operations in InterMail.

SNMP

SNMP (Simple Network Management Protocol) 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 on the MTA, and the total number of messages on the 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 Kx 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 to 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 by pinging. However, the `imsysmon` utility can provide time-sensitive monitoring of servers. The `imsysmon` utility 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 sends 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 are to be monitored, where log files are to be written, and who is to receive e-mail and/or pager alerts. You modify the `imsysmon.ini` file (located in `$INTERMAIL/config`) to configure the `imsysmon` utility.

The `imsysmon.ini` file has three sections:

- IMSYSMON contains variables controlling the running environment of `imsysmon`
- INTERMAIL contains variables setting the location of InterMail
- DEFAULT contains variables setting the threshold limits for parameters that affect the operation of `imsysmon`

This section describes the variables and meaning of values 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`	E-mail addresses that will receive a message when a Fatal or Urgent event occurs.

`MailFilters`	Urgent error log codes for which you do not want e-mail notification from `imsysmon`.
`EmailFromAddr`	The address `imsysmon` uses to send its e-mail messages from. This account is not created automatically. It must exist and be active for `imsysmon` e-mail notification to work.
`MaxConnAttemptNote`	Maximum number of attempts allowed to connect to a port.
`MinErrDiskUsage`	Minimum amount of disk usage before an Error event is reported.
`MinFatlDiskUsage`	Minimum amount of disk usage before a Fatal event is reported.
`MinUrgtDiskUsage`	Minimum amount of disk usage before an Urgent event is reported.
`MaxErroFreeExtents`	Minimum percentage of free extents available before an Error event is reported.
`MaxFatlFreeExtents`	Minimum percentage of free extents available before a Fatal event is reported.
`MaxUrgtFreeExtents`	Minimum percentage of free extents available before an Urgent event is reported.
`PageRcptTo`	E-mail addresses 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 variable in the INTERMAIL section and its meaning is as follows:

`UserName`	Name of the InterMail user, typically the same value as for the `common/commonUser` configuration key.

DEFAULT: Setting the Threshold Limits for imsysmon

The DEFAULT section of the `imsysmon.ini` file determines which events are to be monitored. 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, 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 an event 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 the severity level to be reported. If you do not want `imsysmon` to monitor and report a particular event, you 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`.
`ProcNotFound`	An InterMail server is not responding to a ping.
`FatalMsgsFoundInLog`	Fatal messages were found in server logs.
`AlarmTimeout`	A timeout has occurred while the server was waiting for a task.
`ArchivePathNotFound`	The Oracle archive path for a specific instance could not be found.
`DiskSpaceCheckFail`	An error occurred during a disk space check.
`LogFileOpenFail`	The `imsysmon` log file could not be opened.
`UrgtDiskSpace`	The percentage of disk usage is equal to or greater than `MinUrgtDiskUsage`.
`PortBannerFail`	Connection to a server port expecting a banner has failed.
`UrgtFreeExtents`	The number of free Oracle extents is less than or equal to `MaxUrgtFreeExtents`.
`UrgtMsgsFoundInLog`	Urgent messages were found in server logs.

`CheckFreeExtentsFail`	A check for free Oracle extents failed.
`ErroFreeExtents`	The segment for an Oracle instance is low.
`IMAPQuitError`	The IMAP server rejected a “QUIT” command.
`IMDbSpaceChecker`	`imdbspacechecker` failed.
`IMDbLogParsingFailure`	Parsing an InterMail log file failed.
`IMServPingFail`	`imservping` failed.
`LogSegmentCreationFailure`	Creation of a log segment failed.
`LogSegmentReadFailure`	Reading of a previously saved log segment failed.
`POPQuitError`	The POP server rejected a “QUIT” command.
`SendMailAttemptFail`	Notification of an event through sendmail failed.
`SMTPDataError`	The SMTP server rejected the “DATA” address.
`SMTPFromError`	The SMTP server rejected a “MAIL FROM” address.
`SMTPRcptToError`	The SMTP server rejected a “RCPT TO” address.
`SMTPMessageBodyError`	The SMTP server rejected the body of the message.
`SMTPNotifyFail`	Notification of an event through e-mail failed.
`SMTPQuitError`	The SMTP server rejected a “QUIT” command.
`SocketSMTPMailFail`	Notification of an event through e-mail failed.
`SockOpenFail`	A socket to communicate with one of the servers could not be created.
`UnhdlSignal`	An unaccounted-for signal occurred.
`WarnDiskSpace`	The percentage of disk usage for a file system is at least `MinErrorDiskUsage`.
`BadPortBanner`	A bad port banner was received.
`NoteDiskSpace`	A notification message on disk space.
`NoteFreeExtents`	A notification message on free extents.

`NoteOraInstResponded`	A notification message that an Oracle instance has started.
`ProcFound`	A process was found.
`PortBanner`	A banner was 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

# MailFilters - List any Urgent error log codes that you do *not*
#               want to be notified about via email from imsysmon.
MailFilters = <<EOT
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

[DEFAULT]

# Percent of disk usage is at least MinFatlDiskUsage.
FatlDiskSpace = 4
......
```

Running imsysmon

After configuring the `imsysmon.ini` file, you are ready to run the `imsysmon` command as follows:

1. Set user to `root`.
2. Enter:

   ```
   imsysmon
   ```

Note: For a complete description of available syntax for `imsysmon`, see the *InterMail Kx Reference Guide*.

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
        InterMailDir           /vol1/imail
        VerboseMode            0
        LogMode                1
        LogFile                /vol1/imail/log/
imsysmon.paris.19990108144615.log
        MailDeliveryHosts      earth:7131
        WaitTime               2
        Environment: imail
imsysmon:  monitoring InterMail modules: mss mta popserv
--
Checking module mss...
--
imsysmon:  FATAL!  19980508144624:  mss does NOT respond to connection
requests.
--
Checking module mta...
--
…
```

Based on the output shown in the example above, `imsysmon` creates a log file (`imsysmon.venus.19990108144615.log`) and sends an e-mail to the addresses specified in `MailRcptTo`.

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 the amount of information 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 Kx 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 you can roll over after a configurable period of time. It 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 printed 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 of the 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 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, set the value of the configuration key `traceToStderr` to `true`, or 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 28 shows a typical event log in an event log file.

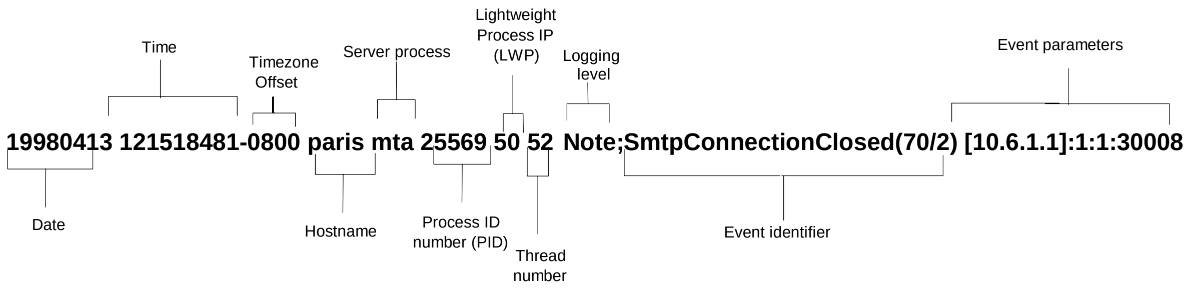

Time
Timezone Offset
Server process
Lightweight Process IP (LWP)
Logging level
Event parameters
19980413 121518481-0800 paris mta 25569 50 52 Note;SmtpConnectionClosed(70/2) [10.6.1.1]:1:1:30008
Date
Hostname
Process ID number (PID)
Thread number
Event identifier

Figure 28 Event log file format

This event log 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.

Hostname—The machine name where the server resides.

Server Process—The name of the process that logged an event (such as `mta` or `popserv`).

Process ID number (PID)—The standard PID of the operating system.

Light Weight Process ID (LWP)—The light-weight PID; not supported by all UNIX platforms.

Thread Number—The thread number for multi-threaded processes,

Logging Level—The logging level for each log event. This level indicate the level of effect that an event will have on a server or process. Logging levels are Fatal (Fatl), Urgent (Urgt), Error (Erro), Warning (Warn), and Notification (Note). For additional information, see “Log Files” on page 188.

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 contain the network address of the connection.
- Message events include a message ID as their first parameter.
- Events associated with a specific client identity provide the username when it is available.
- Process IDs 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 29 shows a typical status file entry:

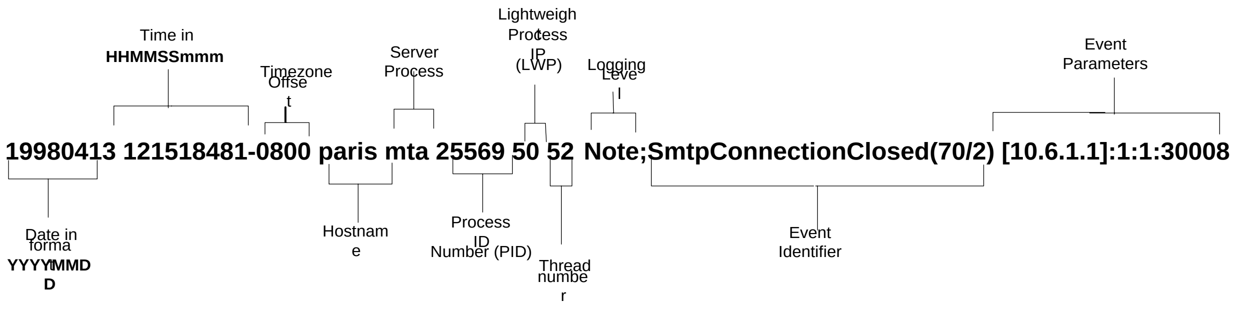


Figure 29 Statistics log file format

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 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 program resides.

Program—The name of the program that logged the statistic (such as `mta` or `popserv`).

Process ID (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

If the message tracing capability is turned on for a server, an entry is added to the server's log file whenever a message is processed. These message tracing entries allow you to trace messages through the system, and help to track down problems with messages.

The format for entries to 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 Kx 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 pre-filtering 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[localtime]` to have the time stamps converted to local time.

Although `imlogprint` typically accepts a log entry from standard input, you may 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 example below shows a standard `imlogprint` command and resulting output.

```
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
```

In this next example, the command also includes –x `1-2` to exclude the first two fields of the log entry (the date and time).

In this example, `imlogprint` prints multiline reformatted output that indicates the date, time, host, server process, and other information in an easy-to-read format. It also uses tags (`-t`) to identify each field in the output.

```
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 Kx Reference Guide*.

Accessing Log Data Through Named Pipes

Logs are archived records of 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 can use the following configuration keys to control this behavior:

- `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>` (where `<filetype>` can be `log`, `stat`, or `trace`).

13

Backup and Recovery

A comprehensive backup strategy combined with properly tested recovery technique enables your data to be recovered in the event of catastrophic failure. Each site will have its own issues concerning its provisioning system, the size of its database, available hardware, and available person-hours that must be addressed on a case-by-case basis.

This chapter presents the concepts supporting backup and recovery in InterMail, not a prescriptive, step-by-step method. You can apply these concepts to your own backup and recovery environment. This chapter provides:

- An overview of InterMail's backup and recovery strategies
- Instructions to guide you in backing up and recovering account data, message data, and configuration information

Backing Up the Mail System

InterMail backup and recovery procedures address three types of disaster scenarios:

- **Hardware failures**—Memory or disk errors; or CPU, disk controller, device driver, or total system failure
- **Software failures**—Operating system errors, or third-party software application failure
- **Operator errors**—Errors due to unexpected user behavior such as the erroneous deletion of files or logs, or accidental shutdown of the system

Note: No backup and recovery strategy can be considered effective until it has been fully tested under field conditions. You should conduct regular tests in a lab environment to ensure that the recovery strategy works.

There are many considerations involved in backing up your InterMail system. What should you back up, and how often should backups occur? What strategies should you employ to make the most efficient use of your backup and recovery resources? Precise answers to these questions will depend on a variety of factors including available hardware, message traffic, and any number of variables that are unique to your InterMail configuration.

The frequency with which you should back up your system depends on the characteristics of your installation, including the amount of mail processed per day, the potential importance of a single message, the time required to execute a backup, and so on. You should review the practices established at your site for guidance in determining how often to back up the mail system.

When backing up the InterMail files, it is recommended that you do the following:

- Observe the original naming conventions for directories and files. This is not required, but it will make things simpler if you need to use the recovery procedure.
- Store all backup files on another host.
- Store all files and directories relating to a single backup in the same parent directory.

Types of Backup

There are two types of backup in InterMail. These backups provide protection for messages stored on the Message Store Server (MSS), account information contained in the Integrated Services Directory (ISD), and system configuration information in the Configuration database:

- Occasional complete image backups of the host system software, InterMail software, configuration files, and other resident software
- Regularly scheduled backups of the MSS and Directory server

Complete Image Backup

A complete image backup involves copying the entire file system, or at least the InterMail portion. It can be carried out only when the service is down and should be performed during regularly scheduled maintenance windows or as opportunity allows when servers are down for other reasons.

You should also make a complete image backup under the following circumstances:

- Whenever you change the software configuration of the InterMail host, including after the first installation of InterMail software. This is a fairly infrequent operation.
- Whenever you add another host to your system. A new host can cause configuration information to change, messages to be stored on a second host, and mail to be deferred on a second host. In such a case, you may want to be able to restore the system to the state it was in before the additional host was added.

Complete image backup is essential for recovering the current working system in an emergency. It should take place whenever the system is in a quiescent state: the more recent the complete backup, the better. Immediately after an InterMail installation, no services have started. If you have started them manually or if this is an existing InterMail system, stop the services before making the backup.

Note: Whenever installing new software, you should back it up immediately. This type of backup should be part of your environment system backup strategy. The backup procedures presented here do not incorporate this type of file protection. If you have a highly customized installation, note your customized directories and be sure to back them up accordingly.

The specific backup method you should use to perform this operation depends on your system and configuration. You should copy the entire installation, all directories and files created as a part of InterMail, and any other files that 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. The system can begin running at this point. If this is a first-time installation, you can begin provisioning new accounts.

Mail System Backup and Recovery

The mail system must be backed up regularly. You can make these backups online, without shutting down the system. However, feature is available only if it is enabled in your license key.

It is recommended that you do this kind of backup daily in order to minimize the damage caused in the case of a system failure.

Regular backup in InterMail should focus on:

- Permanent mail storage
- Temporary mail storage
- Account information
- System configuration information

Permanent Mail Storage

Data pertaining to mailboxes and messages is stored in the Message File system and the Message Store database. The Message Store database maintains tables of information that use pointers to allow designated mailboxes to access specific files in the Message File system. In addition, these tables keep track of the status of message files (which users have read and deleted them, for example). The message files themselves (which contain the content of the messages) are stored separately in the Message File system, in the `$INTERMAIL/msgfiles` directory.

Message Store database files are located in the `$INTERMAIL/db/mbox` and `$INTERMAIL/db/msgid` directories.

Temporary Mail Storage

Messages in transit may be stored temporarily for a variety of reasons: the MSS or Directory server may be unavailable, or a remote SMTP host may be down. In such situations, instead of handling messages in memory, the Message Transport Agent (MTA) stores the messages in its local `spool` directory. If you so choose, you can periodically back up the `spool` directory on the MTA. However, since this is transient data, you do not need to do this unless you are concerned that something is wrong with your system.

Account Information

It is essential that you back up the ISD on a regular basis, since it is the sole source of directory information for InterMail.

The frequency of backups depends on how fast the data in the ISD changes. If your site creates or modifies a large number of accounts, you should back up this data relatively often.

Data in the Directory database is contained in the following subdirectories under `$INTERMAIL/db:`

- `entry`
- `index`
- `dircache`

System Configuration Information

The Configuration database contains values for all configuration options for each server and controls all InterMail components. This configuration data resides in the `config.db` file in the `$INTERMAIL/config` directory. The Configuration server manages the content of this database. It is extremely important that you back up the configuration information in the Configuration database.

Backup Instructions

This section tells you how to back up the:

- Directory database
- Message Store database and Message File system
- MTA `spool` directory
- Configuration database

The backup instructions in this section assume that you have installed InterMail in the default locations. If you selected other locations, you must adjust the instructions accordingly.

Directory Database

The procedure for backing up the Directory database, contained in the `entry`, `index`, and `dircache` subdirectories, is as follows:

1. Use the `imconfcontrol` command to turn backup mode on:

   ```
   imconfcontrol –install –key /hostname/imdirserv/backupMode=true
   ```

   where `hostname` is the logical host name, from `$INTERMAIL/config/hostname`.

2. Back up the following files in the order shown using a separate UNIX `tar` command for each:

   ```
   $INTERMAIL/db/entry/*/db
   $INTERMAIL/db/index/*/db
   $INTERMAIL/db/dircache/*/db
   $INTERMAIL/db/entry/*/log.*
   $INTERMAIL/db/index/*/log.*
   $INTERMAIL/db/dircache/*/log.*
   ```

   For example:

   ```
   tar cvf backup.entry.tar $INTERMAIL/db/entry/*/db
   tar cvf backup.index.tar $INTERMAIL/db/index/*/db
   tar cvf backup.dircache.tar $INTERMAIL/db/dircache/*/db

   .
   .
   .
   ```

Note: You should build the list of `log.*` files after all the `db` files have been backed up. Otherwise, you will miss any new `log.*` files that have been created in the interim. Although the recovery will still work, some transactions will be lost.

3. Use the `imconfcontrol` command to turn backup mode off:

   ```
   imconfcontrol –install –key /hostname/imdirserv/backupMode=false
   ```

Warning! Do not forget this step, even if the backup is interrupted. The server will gradually use up more and more disk space if backup mode is not turned off.

Message Store Database and Message Files Directory

The procedures for backing up the Message Store database and the Message Files directory is as follows:

1. Use the `imconfcontrol` command to turn backup mode on:

   ```
   imconfcontrol –install –key /hostname/mss/backupMode=true
   ```

   where `hostname` is the logical host name, from `$INTERMAIL/config/hostname.`

2. Back up the following files in the order shown using a separate UNIX `tar` command for each:

```
$INTERMAIL/db/mbox/*/db
$INTERMAIL/db/msgid/*/db
$INTERMAIL/db/mbox/*/log.*
$INTERMAIL/db/msgid/*/log.*
$INTERMAIL/msgfiles
```

For example:

```
tar cvf backup.mbox.tar $INTERMAIL/db/mbox/*/db
tar cvf backup.msgid.tar $INTERMAIL/db/msgid/*/db

.
.
.
```

Note: You should build the list of `log.*` files after all the `db` files have been backed up. Otherwise, you will miss any new `log.*` files that have been created in the interim. Although the recovery will still work, some transactions will be lost.

3. Use the `imconfcontrol` command to turn backup mode off:

```
imconfcontrol –install –key /hostname/mss/backupMode=false
```

Warning! Do not forget this step, even if the backup is interrupted. The server will gradually use up more and more disk space if backup mode is not turned off.

MTA Spool Directory

To back up the local `spool` directory, use the UNIX `tar` command:

```
$INTERMAIL/spool
```

For example:

```
tar cvf backup.spool.tar $INTERMAIL/spool
```

Configuration Database

To back up the system Configuration database file, use the UNIX `tar` command:

```
$INTERMAIL/config/config.db
```

For example:

```
tar cvf backup.config.tar $INTERMAIL/config/config.db
```

Recovery Instructions

Restoring your mail system directories is a relatively easy process, provided that you took the precaution of making regular and frequent backups. The recovery strategy described in this section will restore your system to its state as of the last backup. Any messages received and not read by users since the last backup will be lost.

InterMail generates database log files that help ensure graceful recovery from machine crashes, server crashes, or ungraceful server shutdowns. These log files are called `log.NNNNNN`, where `NNNNNN` is some number and are stored under subdirectories in the `$INTERMAIL/db` directory. InterMail automatically uses these files when the system is restarted after a crash. These files are deleted automatically when they are no longer required for recovery.

Note: The software may recover on its own under certain conditions. If the database host was powered off unexpectedly, or if the process was killed, the InterMail database may be able to fix itself using the log files it keeps. However, UNIX may have corrupted the files that were being written. You should save these corrupted files if you have the space, in case they contain valuable information that was written since the last backup was made.

If you have backed up the mail system files as instructed in the previous section, you should have copies of the following:

- Directory account information
- Mailbox information
- Configuration information

Once you make sure that you have the required backup files, restore backups of the following directories/files on top of the current versions:

- `msgfiles`
- `db/mbox`
- `db/msgid`
- `db/entry`
- `db/index`
- `db/dircache`
- `spool`
- `config/config.db`

The following sections describe how to restore backups of individual components.

Directory Database

To restore the Directory database, which is contained in the `entry`, `index`, and `dircache` subdirectories:

1. Make sure `imdirserv` is shut down.
2. Delete the directories, as follows:

```
rm –rf $INTERMAIL/db/entry
rm –rf $INTERMAIL/db/index
rm –rf $INTERMAIL/db/dircache
```

3. Restore the backed up files using the UNIX `tar` command. For example:

```
cd $INTERMAIL
tar xvf backup.entry.tar
tar xvf backup.index.tar
.
.
.
```

4. Run `imdbrecover` on the restored files:

```
imdbrecover -d
```

Message Store Database and Message File System

To restore the Message Store database and the Message File system:

1. Make sure the MSS is shut down.
2. Delete the directories, as follows:

```
rm –rf $INTERMAIL/db/mbox
rm –rf $INTERMAIL/db/msgid
rm –rf $INTERMAIL/msgfiles
```

3. Restore the backed up files using the UNIX `tar` command. For example:

```
cd $INTERMAIL
tar xvf backup.mbox.tar
tar xvf backup.msgid.tar

.
.
.
```

4. Run `imdbrecover` on the restored files:

```
imdbrecover –m
```

MTA Spool Directory

To restore the `spool` directory:

1. Make sure the MTA is shut down.
2. Restore the backed up file using the UNIX `tar` command:

```
tar xvf backup.spool.tar
```

System Configuration Database

To restore the Configuration database:

1. Make sure the Configuration server is shut down.
2. Restore the backed up file using the UNIX `tar` command:

   ```
   tar xvf backup.config.tar
   ```

14

LDAP Directory Synchronization

You can synchronize LDAP directory data between an InterMail directory and a foreign directory in a master and slave configuration. After the contents of the Directory database are copied from the master to the slave, you can configure and enable continuous data synchronization between the two directories.

This chapter covers:

- Operation of the synchronization utility
- An overview of replication and synchronization
- Preparing for LDAP replication
- Preparing for LDAP synchronization
- Importing data from a foreign directory

Operation of the Synchronization Utility

The LDAP directory synchronization utility operates under the following rules:

- Only one synchronization direction is supported at a time for a dual-directory setup. Directory data cannot be sent simultaneously from the InterMail directory to the foreign directory and from the foreign directory to the InterMail directory.
- The master directory is always considered authoritative.
- Only the master directory in a dual-directory setup requires backup.
- If the foreign directory is the master LDAP database, it must contain all objects required in the InterMail LDAP schema. For information on the LDAP schema, see the *InterMail Kx Reference Guide*.
- Before attempting to do any schema translation, you should make sure that the foreign directory schema matches the InterMail schema as much as possible in order to minimize the changes that need to be made to the `xlatreplog` tool. This tool is described in “Configuring the xlatreplog Utility” on page 233.

Note: You should be familiar with LDAP in order to do schema translation.

- InterManager is supported only in configurations in which the InterMail directory is the master. This is because the InterManager application structure is not normally supported by third-party directories.
- It must be possible to generate a change log (replication log file) with the master directory. The slave directory is synchronized using this file.
- Consistency between the InterMail directory and the foreign directory cannot be guaranteed at any given time; however, consistency is enforced between the two directories at the time of each update.

Replication and Synchronization Overview

This section provides an overview of the replication and synchronization processes. For instructions on preparing for replication and synchronization, see “Preparing for LDAP Replication” on page 231 and “Preparing for LDAP Synchronization” on page 232.

LDAP replication in InterMail can occur in either of two ways:

- A foreign directory can be imported into the InterMail system. In this scenario, the foreign directory is the master directory, and the InterMail directory is the slave.

Note: After the initial import of data from the foreign directory, all directory changes are made in the master (foreign) directory in order to synchronize the two directories.

- InterMail data can be exported to a foreign directory. In this scenario, the InterMail directory is the master directory, and the foreign directory is the slave.

The replication process described in this section assumes the data export scenario.

LDAP Replication

Figure 30 illustrates the LDAP replication process using the export method.

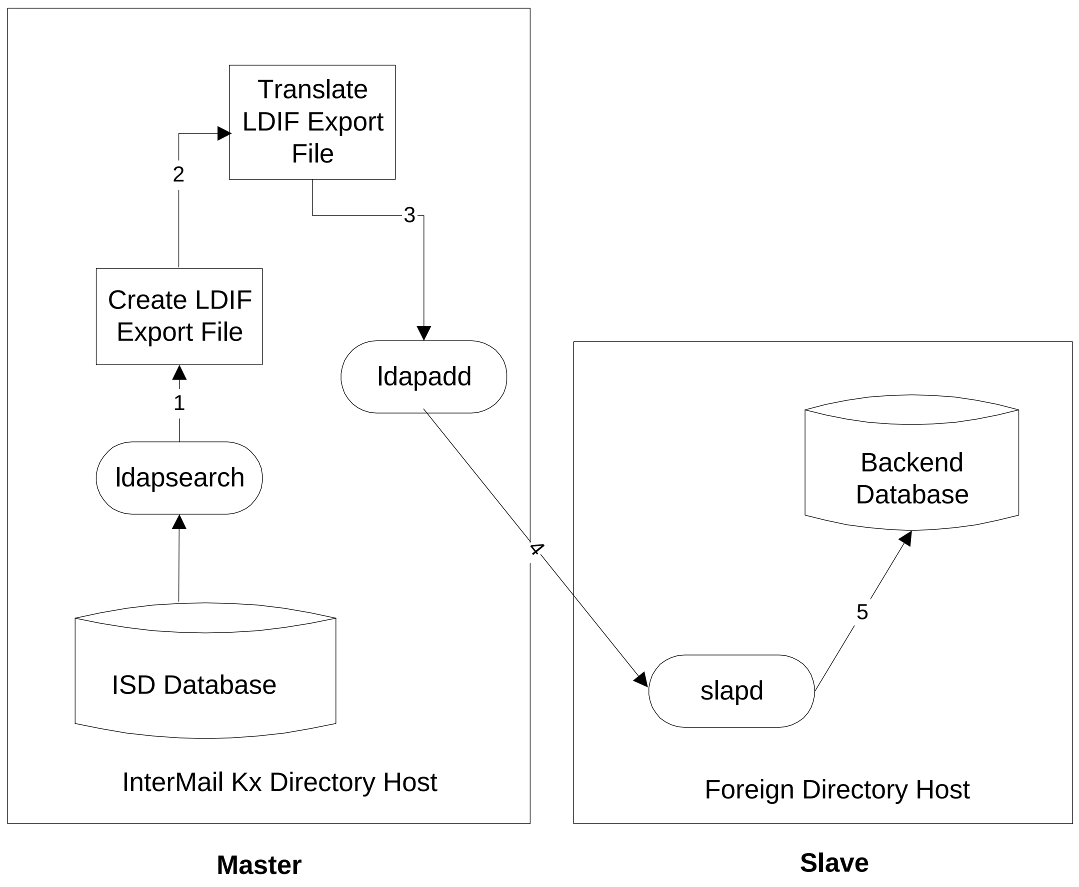


Figure 30 LDAP replication

LDAP Replication

Steps 1 through 4 show how LDAP directory data is replicated from InterMail to a foreign directory:

1. The `ldapsearch` utility produces an LDIF export file representing a complete or partial copy of the contents of the Integrated Services Directory (ISD) database.
2. If there are schema differences between the master and slave directories, the LDIF export file must be translated. You can translate it by manually editing the LDIF export file or by customizing the `xlatreplog` tool to modify the file.
3. The `ldapadd` utility reads the contents of the LDIF export file.
4. The contents of the LDIF export file are sent to the `slapd` process on the foreign host for processing.
5. The `slapd` process on the foreign host writes these changes to the backend database.

LDAP Synchronization

After the LDAP replication process is complete, LDAP synchronization can begin.

Figure 31 shows the LDAP synchronization process.

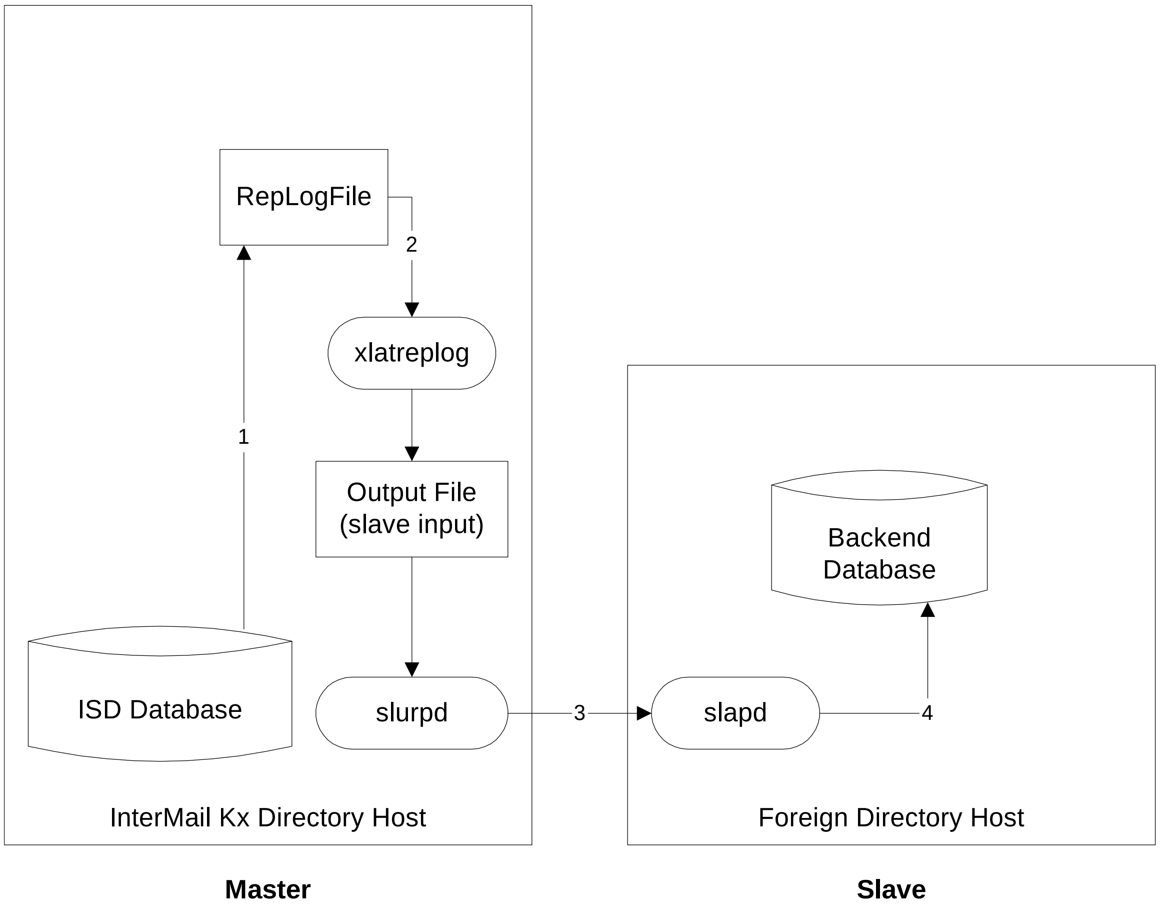


Figure 31 LDAP synchronization

Steps 1 through 4 describe the process of LDAP synchronization:

1. When changes are made to the ISD database, the Directory server sends these changes to `RepLogFile`, which contains an LDIF record of all changes that have been made to the Directory database since the last time either of the following occurred:
   - The Directory server was rebooted or started
   - The `slurpd` utility processed `RepLogFile`
2. Whenever it detects that changes have been made to `RepLogFile`, the `xlatreplog` utility translates all entries in `RepLogFile` to ensure that the foreign directory will understand all schema elements. The `xlatreplog` utility

produces a slave input file consisting of directory information that has been added or modified.

3. The `slurpd` process waits for and processes the slave input file and sends this data to the `slapd` process on the foreign directory host. `slurpd` uses the data as a stream of modifications to be made to its internal database.
4. The `slapd` process on the foreign directory host writes the LDAP directory data to the backend database.

Preparing for LDAP Replication

The process of replicating directory data from InterMail to the foreign directory using data export involves the following preparatory steps:

1. Stopping provisioning on the Directory server
2. Creating an LDIF export file
3. Copying directory data

This section describes these steps in detail.

Stopping Provisioning on the Directory Server

To ensure that no modifications are made to the master directory during replication, thereby causing the master and slave databases to become unsynchronized, you must stop the provisioning of the Directory server. If you do not, modifications might be made to the master database but not propagated to the slave directory.

Creating an LDIF Export File

Once you have stopped provisioning on the master Directory server, you are ready to use the `ldapsearch` utility to process InterMail directory data and produce an LDIF export file. Depending on the amount of information in the ISD, you can use this utility to process the entire Directory database at once, or you can sort portions of the database and import the data in increments.

Note: Before running `ldapsearch`, it is important to familiarize yourself with the schema definitions contained in both the slave and master directories. If the InterMail schema contains elements that are not present in the foreign directory, either the elements must be added to the `slapd.oc` configuration file on the foreign host, or the elements must be added to the `XlatRecord` subroutine in the `xlatreplog` utility. For more information, see “Configuring the xlatreplog Utility” on page 233.

For example, to produce an export file of the entire contents of the ISD, you might enter:

```
ldapsearch –L –D cn=root –w <ldapRootPwd> -b “mail=*software.com” >
export
```

Warning! You must use the `–L` argument in order to produce an LDIF export file. Failure to do so will prohibit you from using the file for export purposes.

For more information on using `ldapsearch` and producing a partial export file based on query parameters, see the `ldapsearch` man page located in the `man` directory or see the *InterMail Kx Reference Guide*.

Copying Directory Data

After you have created an LDIF export file using the `ldapsearch` utility, you are ready to add it using the `ldapadd` utility. In order to use the `ldapadd` utility, you must have IP access to the LDAP server you want to search; to have this access, you must be within the firewall.

When you invoke the utility, specify the filename of the LDIF export file and the hostname and port number of the foreign directory host. For example:

```
ldapadd -h <foreignHost> -p <port> -f <file>
```

The `slapd` process on the foreign directory then services the request.

Preparing for LDAP Synchronization

LDAP synchronization is an ongoing process that needs to be initially configured before it can run. In order to make sure that all new data is written to the master directory (in this example, InterMail), you must stop all provisioning activities and perform the following steps:

1. Check for the foreign directory host.
2. Set the `ldapRepLogFile` and `ldapRepLogRollHours` configuration keys.
3. Configure the `xlatreplog` utility.
4. Configure `slurpd`.
5. Restart provisioning on the master directory.

When you have completed these steps, LDAP synchronization begins automatically.

This section describes these steps in detail.

Checking for the Foreign Directory Host

For synchronization to proceed, the backend database must be available and running. Check to make sure that this host is available and that it is operating normally.

Setting the Configuration Keys

To have a replication log file (`RepLogFile`) created, set the `ldapRepLogFile` configuration key to the directory path and filename to which you would like `RepLogFile` to be written. For example:

```
/*/imdirserv/ldapRepLogFile: [RepLog/RepLogFile]
```

Also, set the `ldapRepLogRollHours` configuration key to the interval at which you want `RepLogFile` to roll over into an archived version of the replication log file. For example:

```
/*/imdirserv/ldapRepLogRollHours: [1]
```

Configuring the xlatreplog Utility

In order for `slurpd` to process `RepLogFile`, the file must be processed through the `xlatreplog` utility into the file format of the slave input file. Setting up the `xlatreplog` utility in the LDAP synchronization process involves two steps:

1. Modifying the `XlatRecord` subroutine in `xlatreplog`
2. Running the `xlatreplog` utility as a continuously looping process

Modifying the XlatRecord Subroutine in xlatreplog

The `XlatRecord` subroutine in the `xlatreplog` utility defines how the LDAP schema elements are translated. You may need to modify this subroutine using a text editor. In the following example, the attribute `emailaddress` is remapped to the attribute `mail`:

```
sub XlatRecord( @ )
############################################################
# Input:  A complete LDIF record as seen in a replication log file.
#
# Return: A rewritten LDIF record with any schema translations done.
#
# Comments:
#     This is a simple translation example.  It only remaps the attribute
#     named "emailaddress" to the attribute named "mail".
#
#     Be careful about remapping attributes that occur in keywords.  For
#     example, remapping the attribute "lace" will corrupt "replace"
#     instructions in the replication log file.
############################################################

{
    my @rec = @_;
    my @result;

    # Add lines to the start of every record here
    push @result, "replica: lex-devsun4.software.com:389";
    foreach (@rec)
    {

        # Translate single attribute names... but be careful about
        # rewriting more than you intend!
        s/emailaddress/mail/;
        push @result, $_;
    }

    # You may need to check the type of record (modify, add, delete) and
    # take special action here.

    return @result;
}
```

You may need to remap other attribute names using this subroutine. For example, to remap the attribute `facsimilenumber` to the attribute `fax`, you would add the following just below the line that remaps `emailaddress` to `mail`:

```
s/facsimilenumber/fax/;
```

Warning! Do not remap more attributes than necessary. In addition, since this script performs a regular-expression string replacement, you should check the uniqueness of attribute names to be added, as well as of any string expressions that already exist in the file.

Running the xlatreplog Utility

Set up the `xlatreplog` utility so that it produces an output file that can be read by `slurpd`:

```
xlatreplog –i <RepLogFile> –o <SlaveInputFile>
```

Where:

`RepLogFile`	Is the name of the replication log file.
`SlaveInputFile`	Is the name of the slave input file.

By default, `xlatreplog` will run in a continuously looping mode so that any new changes to the `RepLogFile` are translated and ready to be input to the slave directory through `slurpd`.

Configuring slurpd

The `slurpd` process takes the output file created by the `xlatreplog` utility and sends it to the `slapd` process on the foreign directory host.

You configure `slurpd` with two entries:

- The first entry is a line starting with `replogfile`, specifying the name of the output file created by the `xlatreplog` utility.
- The second entry is a line starting with `replica`, specifying the slave database information. Although this second line is normally called `slapd.conf`, this is not required. The only requirement is that this `replica` line specify the same host and port as the `replica` lines of the output file created by the `xlatreplog` utility.

Because `slurpd` checks at regular intervals for new entries to be sent to the slave directory, you should invoke `slurpd` so that it reads the slave input file and runs as a daemon process instead of as a single invocation. For more information, see the `slurpd` man page.

Restarting Provisioning on the Master Directory

Since all directory changes will now go to the replication log file and be transferred automatically from master to slave, you can now restart provisioning on the master directory.

Importing Data from a Foreign Directory

To import directory data from a foreign directory (master directory) to an InterMail directory (slave directory):

1. Make sure that the `slapd.conf` file (the master configuration file for `slapd` and `slurpd`) contains these settings:

   ```
   include ./slapd.at.conf
   include ./slapd.oc.conf
   schemacheck off
   # ldbm database definitions
   database ldbm
   directory <directoryname>
   suffix ““
   rootdn “cn=root”
   rootpw secret
   index cn,sn,uid pres,eq,approx
   index default none
   lastmod on
   # replication
   replogfile <replogfilename> # points to location of xlatreplog tool
   output
   replica host=sbs=qusun3:389 # slave information (Kx imdirserv)
   binddn=”cn=root” # Kx root cn and pwd
   bindmethod=simple
   credentials=”MonMay1718:17:12PDT1999” # From /*/imdirserv/
   ldapRootPwd
   ```

2. Bring up the master `slapd` server. For example:

   ```
   cd <your directory>
   ./slapd -e slapd.conf -p <uniqueport> -d 65535
   ```

3. Make some changes to the master directory. For example, to add a new user, you would enter:

   ```
   ldapadd -h <masterhost> -p <masterport> -D “cn=root” -w secret
   -f <your directory>
   ```

4. Bring up the LDAP directory synchronization tools:

   ```
   xlatreplog.test -i <location of replog file found in slapd.conf>
   -o <outputfile>
   ```

5. Bring up `slurpd`:

   ```
   etc/slurpd -f slapd.conf -d 65535 -r <output of xlatreplog tool>
   [-t <temporarydirectory>]
   ```

6. Verify that the changes have propagated to the InterMail (slave) directory:

   ```
   imdbcontrol ga <username> software.com
   ```

Note: `username` in the above command should match your `newuser.ldif` file.

A

Configuration Editor Keys

This appendix lists all of the configuration keys that are available via the web-based Configuration Editor. The keys are listed functionally as they appear in the editor. For detailed information about the keys listed here, see the *InterMail Kx Reference Guide*. For operational information about the keys, see the related chapters in this manual. For example, for operational information about the anti-spam keys, see Chapter 8.

Anti-Spam

The following configuration keys are related to security.

```
mta/blockAddresses
mta/blockConnections
mta/blockDomains
mta/blockLocalNoAcct
mta/blockPerAccount
mta/blockUsers
mta/checkAuthentication
mta/dropConnections
mta/rejectSenderBadDomain
mta/rejectSenderIPDomain
mta/rejectSenderNoDomain
mta/relayLocalDomainsOk
mta/relayLocalMustExist
mta/relayNullRestricted
mta/sidelineMessages
mta/sidelineNullToMany
mta/verifyRcpts
```

```
mta/blockReplyCode
mta/dropMaxMessageRcpts
mta/maxNullSenderRcpts
mta/relayMaxRcpts
mta/relayReplyCode
mta/sidelineNumRcpts
mta/sidelineNumRcptsPerConnection
popserv/lockTimeout
mta/blockReplyText
mta/blockTheseAddresses
mta/blockTheseDomains
mta/blockTheseIPs
mta/blockTheseUsers
mta/dropRcptsReplyText
mta/dropTheseIPs
mta/relayDestAllowList
mta/relayDestDenyList
mta/relayReplyText
mta/relaySourceDomainList
mta/replaySourceLocalIPList
mta/replaySourceRemoteIPList
mta/incomingMailFilter
mta/rejectDnsServer
mta/relayHost
```

Directory Access

The following keys are related to accessing data in the Integrated Services Directory database.

```
imdirserv/cacheAuthoritativeForMTA
imdirserv/cacheAuthoritativeOnDbFail
imdirserv/cacheDisableReadThrus
imdirserv/cacheDisableWriteThrus
imldapupdate/ldapUpdatePeriod
imldapupdate/logAgeHours
imldapupdate/logExpireHours
```

```
common/binDir
common/installDir
common/logDir
common/nlsDir
common/pidDir
common/queueDir
common/sharedDir
common/tmpDir
imapserv/runDir
imdirserv/DBFilePath
imdirserv/loginFilter
imdirserv/runDir
imldapupdate/ldapHost
imldapupdate/ldapPassword
immgrserv/runDir
imqueueserv/runDir
mss/runDir
mta/MTASpool
popserv/runDir
web/sessionDBFilePath
```

Error Handling

The following keys are related to error handling.

```
popserv/moveRetrieveErrors
mta/Error-Actions/AcctInactive
mta/Error-Actions/AcctInvalidUser
mta/Error-Actions/BadReturn
mta/Error-Actions/FiltActionBounce
mta/Error-Actions/MTAMsgNoRecipients
mta/Error-Actions/MsLimitMsgSize
mta/Error-Actions/MsLimitNumMsgs
mta/Error-Actions/MsLimitTotalSize
mta/Error-Actions/MsNoAllowDeliver
mta/Error-Actions/MtaHostInvalid
mta/Error-Actions/MtaMaxMtaHopCountExceeded
```

```
mta/Error-Actions/MtaMessageQueuedTooLong
mta/Error-Actions/MtaMessageTooLarge
mta/Error-Actions/MtaRecipientsRejected
mta/Error-Actions/MtaSenderRejected
mta/Error-Actions/SmtpClientMailLoopDetected
mta/Error-Actions/SmtpDnsBadConfig
mta/Error-Actions/SmtpProtocolNotSupported
mta/Error-Actions/MtaMessageDelivered
mta/Error-Actions/MtaMessageExpanded
mta/Error-Actions/MtaMessageRejected
mta/Error-Actions/MtaMessageRelayed
mta/Error-Code/AcctInactive
mta/Error-Code/AcctInvalidUser
mta/Error-Code/FiltActionBounce
mta/Error-Code/MsLimitMsgSize
mta/Error-Code/MsLimitNumMsgs
mta/Error-Code/MsLimitTotalSize
mta/Error-Code/MsNoAllowDeliver
mta/Error-Code/MtaHostInvalid
mta/Error-Code/MtaMessageDelivered
mta/Error-Code/MtaMessageExpanded
mta/Error-Code/MtaMessageQueuedTooLong
mta/Error-Code/MtaMessageRejected
mta/Error-Code/MtaMessageRelayed
mta/Error-Code/MtaMessageTooLarge
mta/Error-Code/MtaRecipientsRejected
mta/Error-Code/MtaSenderRejected
mta/Error-Code/SmtpClientMailLoopDetected
mta/Error-Code/SmtpDnsBadConfig
mta/Error-Code/SmtpProtocolNotSupported
```

Logging

The following configuration keys are related to logging operations.

```
common/gmtLogTimes
common/servWarnToStderr
```

```
imapserv/logMailboxCreation
mss/logLogRedirects
popserv/logMailboxCreation
common/logRecorderSize
common/reportParamsInterval
common/rolloverTimeZero
common/rolloversPerDay
mss/logRecorderSize
mta/logRecorderSize
popserv/logRecorderSize
common/logDir
common/logNamedPipeMode
```

Routing

The following configuration keys are related to mail routing options.

```
mta/canonicalize
mta/convertDomainLiterals
mta/rewriteDomains
mta/rewriteOnlyLocal
mta/rewritePrimary
mta/rewriteSaveOrig
mta/rewriteMaxMtaHops
mta/subDomains
popserv/popProxyHost
mta/autoReplySuppressList
common/defaultDomainTable
common/domainName
common/loginDefaultDomain
common/loginNameConvertFrom
common/loginNameConvertTo
mta/completionMethod
mta/defaultDomain
mta/mailRoutingHost
mta/mailRoutingTable
mta/rewriteGatewayHeaderList
mta/rewriteHeaderList
```

B

License Information

This appendix contains license information related to InterMail Kx.

InterMail Licensing Agreement

Apache Server License

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

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 ARISING 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, 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

An attribute of an entry that governs how the other attributes of that entry may be modified. Attribute constraints define which administrators and users (if any) may create and modify attributes, and which legal values they may assign to the attributes.

attribute value

The data associated with an attribute. For example, `555-1212` is an attribute value for the attribute `telephoneNumber`. Attributes may be single-valued or multiple-valued.

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 on 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. A DIT takes the form of an inverted tree, with the root at the top and the branches and leaves below it. Each entry in the DIT has a distinguished name (DN) that uniquely identifies it within the tree, and a relative distinguished name (RDN) that uniquely identifies it among its peers at any node in the tree.

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 by providing a complete pathname through the Directory Information Tree (DIT), analagous to a filename composed of a path of directory names in an operating system.

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 outputs 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 outputs 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, 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 across system, corporate, and international boundaries for information such as names, telephone numbers, and addresses.

LDAP export

The stage of synchronization in which the LDAP directory data is copied from an InterMail directory (the master directory) to a foreign directory (the slave directory).

See also *LDAP import*.

LDAP import

The stage of synchronization in which the LDAP directory data is copied from a foreign directory (the master directory) to an InterMail directory (the slave directory).

See also *LDAP export*.

LDAP replication

The process of initial duplication of directory data through LDAP import or export.

LDAP synchronization

The process by which updated LDAP directory data is continuously copied from a master directory to a slave directory through a replication log file.

See also *LDAP export*, *LDAP import*.

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 descendents.

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 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 `policy` is the string of numbers assigned to Software.com by the Internet Engineering Task Force (IETF), `1.3.6.1.4.1.2415`, followed by a string of numbers designated by Software.com, `1.2.2.1`. This creates an OID of `1.3.6.1.4.1.2415.1.2.2.1` for the object class `policy`.

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.

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 Directory Information Tree (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 the database, 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.

slapd (standalone LDAP daemon)

A process that listens for LDAP connections and services those requests through a backend database.

slurpd (standalone LDAP update replication daemon)

A process that takes slave LDAP input files and passes them to the master `slapd` process.

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.

xlatreplog

A Perl script tool that translates data from the format of the replication log file onteh master directory into the format of the slave input file. This tool must be customized to perform any necessary schema translation.

Index