O P E R A T I O N S G U I D E

InterMail®

Version 4.0

Software.com™
THE INTERNET INFRASTRUCTURE COMPANY™

Document number: O-980603

Table of Contents

Preface

Welcome to InterMail!

The *InterMail Operations Guide* includes instructions for the operation of InterMail. This document assumes you have a technical background as well as a working knowledge of the Internet and high-end system issues.

This manual is supplemented by the *InterMail Reference Guide* which provides information on system architecture, configuration keys, and administrative and database commands, as well as the *InterMail Installation Guide,* which offers detailed instructions for installing InterMail.

Overview of the Manual

The content of this manual is organized as follows:

- Chapter 1 provides an introduction to the InterMail system, including a summary of system components and architecture.
- Chapter 2 offers an outline of pre-production tasks. This is a list of brief discussions of what needs to be done after InterMail is installed, but before bringing the system into production mode.
- Chapter 3 discusses basic system management, including startup and shutdown of servers, as well as server configuration.
- Chapter 4 offers a detailed description of InterMail's security features, including sidelining potential junk e-mail and POP password defense.
- Chapter 5 covers mail routing—how mail flows through an InterMail system.
- Chapter 6 discusses how users' mail is stored, and offers methods for managing mailbox quotas and message aging, which determines how long users' mail can be stored on the system.
- Chapter 7 covers managing "mail in process"—messages that have not yet been delivered by InterMail.
- Chapter 8 offers a discussion of logging, including instructions for determining which types of events will be logged by InterMail.
- Chapter 9 provides information on system monitoring and maintenance, including a discussion of SNMP, and tips for expanding your system.
- Chapter 10 covers backup and recovery.
- Chapter 11 offers basic troubleshooting tips for high availability and failover capabilities.

Style and Conventions

To assist you in understanding the material presented in this manual, the following conventions have been observed:

- Commands and configuration options are referenced by their proper names.
- Environment variables (set at time of installation) are referenced with a preceding `"$"` (e.g., `$spoolDir)`.
- Commands (and other entries you might type) appear in `monospaced` type.
- Variable names for elements within a command either appear between <angle brackets>.
- Optional entries within a command appear in [square brackets] No option is required in such a list.
- Optional entries separated by a vertical bar (pipe) as in [option 1 | option 2] are exclusive—you can choose only one item from such a list.
- Optional entries followed by an ellipsis (…) can be repeated. When an ellipsis follows a bracketed set, the expression within the brackets can be repeated.
- {Curly braces} surround a list of options, one of which is required as an argument.
- **Boldface** indicates literal input used in an example

Questions and Comments

To suggest improvements or provide feedback on the content of this manual, send e-mail to `InterMail.Manual@Software.com`.

Legal Notices

Trademarks

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this documentation, and Software.com was aware of a trademark claim, the designations have been printed in initial caps or all caps.

InterMail and Software.com are trademarks of Software.com, Inc.

Licensing Agreement

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL SOFTWARE.COM BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The RSA Data Security, Inc. MD5 Message-Digest Algorithm

The MD5 Message-Digest algorithm used in InterMail is © 1991-92 RSA Data Security, Inc. Created 1991. All rights reserved.

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.

RSA Data Security, BSAFE

InterMail incorporates a derivative work of the BSAFE cryptographic toolkit, copyright 1992-1996, RSA Data Security, Inc. All rights reserved.

BSAFE is a trademark of RSA Data Security, Inc.

The RSA Public Key Cryptosystem is protected by U.S. Patent #4,405,829.

SSL Plus: SSL 3.0 Integration Suite Toolkit

InterMail incorporates a derivative work of the SSL Plus: SSL 3.0 Integration Suite Toolkit, copyright 1996, 1997 Consensus Development Corporation. SSL Plus: SSL 3.0 Integration Suite is a trademark of Consensus Development Corporation, which reserves all rights thereto.

Portions of the SSL Plus: SSL 3.0 Integration Suite Toolkit software are based on SSLRef(tm) 3.0, which is copyright (c)1996 by Netscape Communications Corporation. SSLRef(tm) was developed by Netscape Communications Corporation and Consensus Development Corporation.

The Regular Expression Routines

The Regular Expression Routines used in InterMail are © 1992-94 Henry Spencer. All rights reserved. This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

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:

1. 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.
2. 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.
3. 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.
4. This notice may not be removed or altered.

The Regents of the University of California Copyright

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.

GNU General Public License

Version 1, February 1989

675 Mass Ave., Cambridge, MA 02139, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The license agreements of most software companies try to keep users at the mercy of those companies. By contrast, our General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. The General Public License applies to the Free Software Foundation's software and to any other program whose authors commit to using it. You can use it for your programs, too.

When we speak of free software, we are referring to freedom, not price. Specifically, the General Public License is designed to make sure that you have the freedom to give away or sell copies of free software, that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of a such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING,

DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any work containing the Program or a portion of it, either verbatim or with modifications. Each licensee is addressed as "you".

1. You may copy and distribute verbatim copies of the Program's 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 General Public License and to the absence of any warranty; and give any other recipients of the Program a copy of this General Public License along with the Program. You may charge a fee for the physical act of transferring a copy.

2. You may modify your copy or copies of the Program or any portion of it, and copy and distribute such modifications under the terms of Paragraph 1 above, provided that you also do the following:

a) cause the modified files to carry prominent notices stating that you changed the files and the date of any change; and

b) cause the whole of any work that you distribute or publish, that in whole or in part contains the Program or any part thereof, either with or without modifications, to be licensed at no charge to all third parties under the terms of this General Public License (except that you may choose to grant warranty protection to some or all third parties, at your option).

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the simplest and most usual way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this General Public License.

d) 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.

Mere aggregation of another independent work with the Program (or its derivative) on a volume of a storage or distribution medium does not bring the other work under the scope of these terms.

3. You may copy and distribute the Program (or a portion or derivative of it, under Paragraph 2) in object code or executable form under the terms of Paragraphs 1 and 2 above provided that you also do one of the following:

a) accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Paragraphs 1 and 2 above; or,

b) accompany it with a written offer, valid for at least three years, to give any third party free (except for a nominal charge for the cost of distribution) a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Paragraphs 1 and 2 above; or,

c) accompany it with the information you received as to where the corresponding source code may be obtained. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form alone.)

Source code for a work means the preferred form of the work for making modifications to it. For an executable file, complete source code means all the source code for all modules it contains; but, as a special exception, it need not include source code for modules which are standard libraries that accompany the operating system on which the executable file runs, or for standard header files or definitions files that accompany that operating system.

4. You may not copy, modify, sublicense, distribute or transfer the Program except as expressly provided under this General Public License. Any attempt otherwise to copy, modify, sublicense, distribute or transfer the Program is void, and will automatically terminate your rights to use the Program under this License. However, parties who have received copies, or rights to use copies, from you under this General Public License will not have their licenses terminated so long as such parties remain in full compliance.

5. By copying, distributing or modifying the Program (or any work based on the Program) you indicate your acceptance of this license to do so, and all its terms and conditions.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

7. The Free Software Foundation may publish revised and/or new versions of the 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 Program specifies a version number of the 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 Program does not specify a version number of the license, you may choose any version ever published by the Free Software Foundation.

8. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, 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

9. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "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 PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

10. 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 PROGRAM 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 PROGRAM (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 PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to humanity, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) 19yy <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 1, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave., Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19xx name of author

Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.

This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (a program to direct compilers to make passes at assemblers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989

Ty Coon, President of Vice

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.

SmartHeap

Portions copyright 1991-1997 Compuware Corporation.

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.

Apache Server License

Copyright (c) 1995-1997 The Apache Group. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions 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 Apache Group for use in the Apache HTTP server project (http://www.apache.org/)."
4. The names "Apache Server" and "Apache Group" must not be used to endorse or promote products derived from this software without prior written permission.
5. Redistributions of any form whatsoever must retain the following acknowledgment:

   "This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/)."

THIS SOFTWARE IS PROVIDED BY THE APACHE GROUP ‘‘AS IS’’ AND ANY EXPRESSED 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 APACHE GROUP OR ITS 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.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Group and was originally based on public domain software written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign. For more information on the Apache Group and the Apache HTTP server project, please see <http://www.apache.org/>.

1

Introduction to InterMail

InterMail is a scalable, high performance, native Internet e-mail system for very large messaging environments. Each InterMail installation is a unique configuration created from a common set of software components. This chapter provides an overview of the InterMail system components, a discussion of the unique technologies supporting their operation, and detailed descriptions of their architecture and internal operation.

The following topics are outlined here and covered in detail throughout this manual, and in the *InterMail Reference Guide*:

- System components, including a brief description of each InterMail Server
- Information storage—how data is stored by InterMail
- Supporting technology, such as multi-threading and RME
- InterMail features and benefits

1.1 Server Distribution

The InterMail messaging system includes ten servers. Each server operates independently as a separate process with specific responsibilities.

The InterMail servers can be distributed into separate machines for flexible configuration and system scaling. A complete InterMail installation requires at least one server of each type specified, but typically includes multiple instances of all but the Configuration Server. Additional servers can be added as message traffic or the number of users supported increases.

Functionally, InterMail servers can be divided into four categories:

- Servers involved in delivering messages
- Servers involved in storing messages
- Servers involved in retrieving messages
- Servers that manage the system

Although most of the InterMail servers play a role in only one of these functions, there are two (the Directory Cache Server and the Queue Server) that have overlapping duties. Figure 1 illustrates the functional relationship between servers. The sections that follow describe these relationships in greater detail.

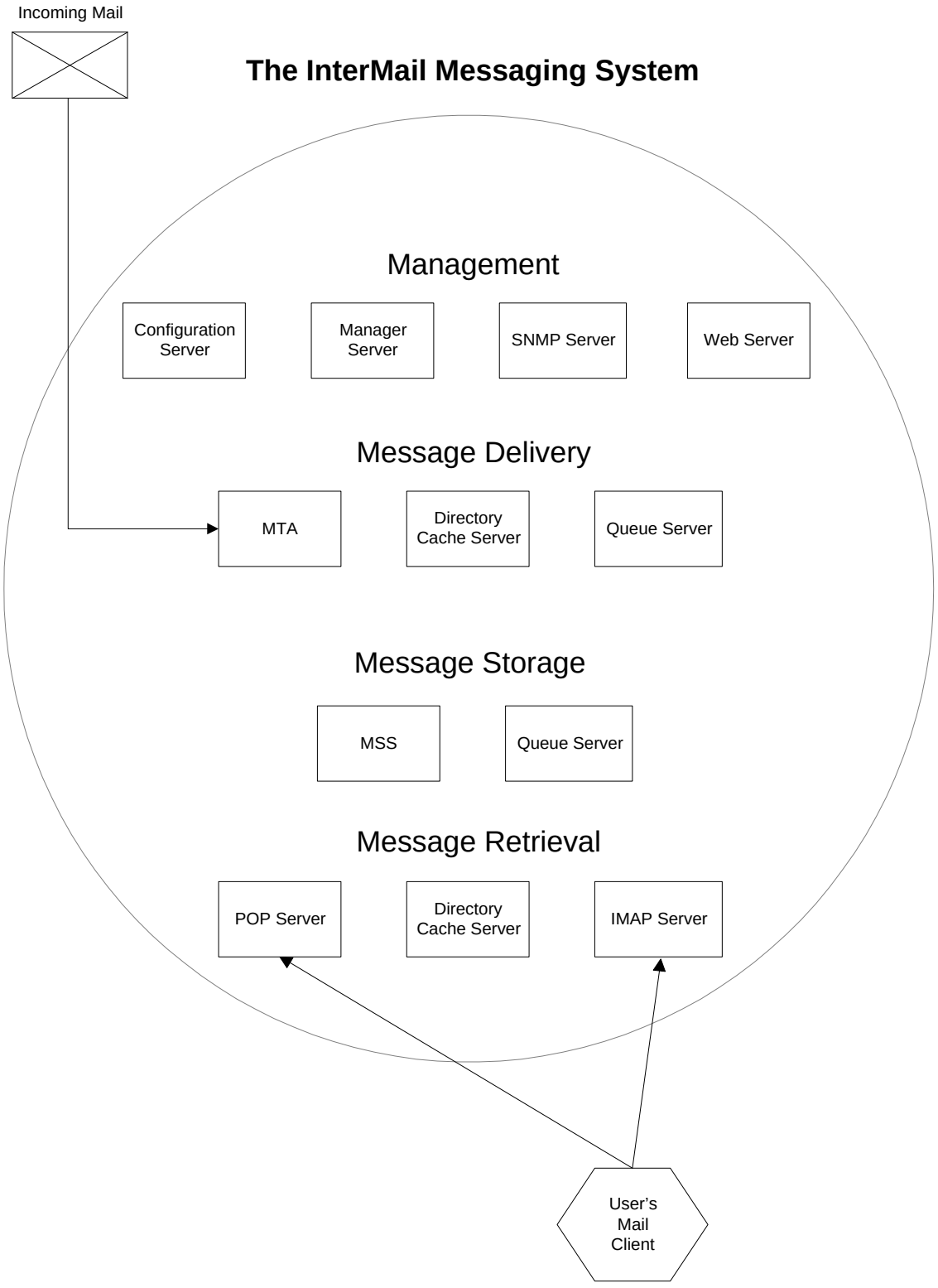

Figure 1. The Functional Role of InterMail Servers

Message Delivery

The MTA, Directory Cache Server and Queue Server work together to provide message delivery.

- The MTA receives and delivers messages. It also enforces relay restrictions and redirection instructions, and manages any errors that occur during the delivery process.
- The Directory Cache provides the MTA with the account information required for message delivery.
- The Queue Server also works with the MTA. Its job is to temporarily store (queue) messages that the MTA cannot deliver immediately.

Message Storage

Both the Message Store Server and the Queue Server play roles in message storage.

- The MSS is responsible for storing messages in users' mailboxes. It takes delivery of messages from the MTA, and services retrieval requests from the POP and IMAP Servers.
- The Queue Server also plays an role in message storage as it is responsible for storing any messages the MTA cannot deliver immediately. It is frequently installed on the same host as the MSS so that all messages can be backed up from a single machine. However, this is not a requirement.

Message Retrieval

Message retrieval by end users is handled by the POP, IMAP and Directory Cache servers.

- The POP Server handles requests for message retrieval using the POP3 protocol.
- The IMAP Server handles requests for message retrieval using the IMAP4 protocol.
- The Directory Cache Server, which provides account information for message delivery, has a similar function in message retrieval, providing the class of service, mailbox name and MSS host information that the POP and IMAP Servers require.

Management

The Manager, Configuration and SNMP servers are used to control and monitor the functioning of the entire InterMail system.

- The Manager Server, which runs on each host, is used to start and stop any server running on any host.
- The Configuration Server supplies configuration changes to all other InterMail servers as required.
- The SNMP Server gathers useful system information and, upon request, passes it to your SNMP monitoring station for real-time viewing.
- The Web Server supports access to InterMail account information via a web browser.

1.2 System Components

As illustrated in the previous section, the InterMail messaging system is a combination of components working together to provide message delivery, message storage, message retrieval, and system management. The following sections identify the InterMail components individually and describe the function performed by each one.

1.2.1 Message Transport Agent (MTA)

The MTA handles the receipt of all incoming messages. It listens on the SMTP port, accepts messages from clients, then determines whether the recipients are in a local or remote domain. For recipients in a local domain, it obtains account information from the Directory Cache Server then delivers messages to the appropriate mailbox. For recipients in other domains, it sends mail to the remote location indicated.

The MTA is responsible for enforcing relay restrictions and executing redirection instructions. It is also responsible for managing any error conditions that arise in the delivery process, producing and sending bounce messages as required.

One MTA may be run per host. Additional hosts can be added independently as needed, with message traffic distributed via a load-leveling mechanism such as round-robin DNS.

Please see Chapter 3 of this manual and Chapter 2 of the *InterMail Reference Guide* for further information on the MTA.

1.2.2 Queue Server

If the MTA cannot deliver a message immediately, the MTA passes the message to a Queue Server which stores the message in its queue directory (see Chapter 7 of this manual). Each Queue Server can be shared by multiple MTAs. This means that a message sent to a Queue Server by one MTA can be reprocessed later by any other MTA.

By running a Queue Server on the same host as the MSS, you keep both your temporarily queued messages and your users' permanently stored messages on the same machine, thus creating a central source for message storage. This allows you to concentrate your backup resources efficiently.

Note: *Although running a Queue Server on the same host as the MSS can be a useful strategy for backing up messages, there is no requirement to do so.*

Please see Chapter 3 of this manual and Chapter 3 of the *InterMail Reference Guide* for further information on the Queue Server.

1.2.3 Directory Cache Server

The Directory Cache Server facilitates the retrieval of account. For fast access, the Directory Cache Server maintains a local copy of account information. It is also capable of reading through to the Integrated Services Directory (the source of all InterMail account information) if it cannot find the desired information in its own local cache.

The Directory Cache Server also interacts with the MTA, and the IMAP and POP Servers, providing the information required for message delivery and message retrieval.

The Directory Cache Server can be run on multiple machines (in fact, a minimum of two such servers is recommended). Configuring a system with multiple Directory Cache Servers provides high performance ISD (Integrated Services Directory) look-ups.

Please see Chapter 3 of this manual and Chapter 4 of the *InterMail Reference Guide* for further information on the Directory Cache Server.

1.2.4 Message Store Server (MSS)

The MSS is the InterMail component responsible for persistent storage of messages in a user's mailbox. It takes delivery of messages from the MTA and services requests for message retrieval from the POP and IMAP Servers. The MSS interacts with both an Oracle database and the Message File System. Information about mailboxes, folders, message headers and message structure is stored in the database. The messages themselves are stored as individual files in the file system.

Note: *Several MSS processes can be run on the same host simultaneously, but they will be supported by a single Message Store database and a single Message File System.*

Please see Chapter 3 of this manual and Chapter 5 of the *InterMail Reference Guide* for further information on the MSS.

1.2.5 POP Server

The POP (Post Office Protocol) Server communicates with the Directory Cache Server to validate the user's login name and password, and to obtain information required to service message retrieval requests (e.g., mailbox location, class of service, etc.).

One POP Server may be run per host. If needed, several hosts can be configured to run POP Servers with the load distributed via a load-leveling mechanism such as round-robin DNS.

Please see Chapter 3 of this manual and Chapter 6 of the *InterMail Reference Guide* for further information on the POP Server.

1.2.6 IMAP Server

The IMAP (Internet Message Access Protocol) Server allows most IMAP-enabled clients to send and receive messages in on-line, off-line and disconnected modes. Unlike POP users whose messages are automatically downloaded to a single local machine each time they connect, IMAP users have the option of working with their messages directly on the server.

The IMAP Server communicates with the Directory Cache Server to validate a user's login name and password, and to obtain information required to service message retrieval requests (e.g., mailbox location, class of service, etc.).

One IMAP Server may be run per host. If needed, several hosts can be configured to run IMAP Servers, with the load distributed via a load-leveling mechanism such as round-robin DNS.

Note: *Since the IMAP protocol gives users the ability to work with messages directly on the server, it is very likely that average connection times will be longer than with POP connections.*

Please see Chapter 3 of this manual and Chapter 7 of the *InterMail Reference Guide* for further information on the IMAP Server.

1.2.7 Configuration Server

Each host on an InterMail system has its own Configuration Database that contains a complete set of system settings.

The Configuration Server runs on a single host and holds a "master" Configuration Database. The purpose of the Configuration Server is to distribute the latest version of the master Configuration Database to all the other hosts of an InterMail system. When you make changes to the master Configuration Database and elect to propagate them, all the InterMail hosts check to see if the master Configuration Database is newer than the one they are currently using. If it is, they pull the latest version from the Configuration Server.

Please see Chapter 3 of this manual and Chapter 8 of the *InterMail Reference Guide* for further information on the Configuration Server.

1.2.8 Manager Server

A Manager Server is installed on each InterMail host. The Manager Server allows you to log on to an InterMail host and start or shut down any of the servers running on any other host in your InterMail system. When you issue a startup or shutdown command from one host, the instructions are passed to the Manager Server on each affected host, and the Manager Servers then carry out the operation(s) you requested.

Please see Chapter 3 of this manual and Chapter 9 of *The InterMail Reference Guide* for further information on the Manager Server.

1.2.9 SNMP Server

The SNMP (Simple Network Management Protocol) Server communicates with the other InterMail servers, gathering useful system information, such as present server status, number of connections since the server started, the number of messages stored in the MSS, etc. A monitoring station can request information from the SNMP server for viewing and processing. Currently, all servers except the Manager and Configuration Servers can be monitored by InterMail's SNMP Server.

Note: *You must supply your own monitoring system.*

Please see Chapter 3 of this manual and Chapter 10 of the *InterMail Reference Guide* for further information on the SNMP Server.

1.2.10 Web Server

The Web Server provides access to account information via a web interface. It supports individual account management in InterMail, and provides a means for delegated administration when used in combination with the InterManager product.

1.3 Information Storage

InterMail servers are complemented by a series of databases that maintain information about InterMail users, their messages, and even the servers themselves. These databases are described in the sections that follow.

1.3.1 The Integrated Services Directory

The Integrated Services Directory (ISD) is the definitive source of InterMail account information. There is a single ISD for the entire InterMail installation. It is the foundation from which all Directory Caches are created.

1.3.2 The Directory Cache Database

A Directory Cache Database, containing a complete copy of all user account and domain information, is maintained on each host running a Directory Cache Server. These caches provide quick, local access to account information.

Each Directory Cache Server automatically updates its local cache with the most recent information in the Integrated Services Directory. This update occurs at a configurable interval (by default, every 60 seconds), guaranteeing that the local cache is up-to-date within this time interval. A Directory Cache Server can also read through to the Integrated Services Directory at any time, requesting account information that is new and not yet contained in its cache. When such a read-through occurs, the Directory Cache Server automatically updates its cache with the account information for the new record, without waiting for the regularly scheduled update interval to pass.

1.3.3 The Message Store Database and Message File System

The Message Store Database and Message File System are responsible for persistent storage of messages. Dynamic data (e.g., information about which users have read and deleted messages) is housed in the Message Store Database. Information about mailboxes, folders, message headers, and message status is maintained here. The message data itself (static information not subject to change) is maintained in the Message File System. Each message is stored in a single file which includes all headers, body text, and attachments.

1.3.4 The Configuration Database

The Configuration Database stores configuration options that control the behavior of the InterMail servers and indicate the location of required system resources. A copy of the Configuration Database is maintained on each InterMail host and is read by every InterMail process at startup.

The “master” Configuration Database is stored on the host on which the Configuration Server resides. Any time you make changes in the Configuration Database, you should always make them to the “master” and then propagate them throughout your system.

1.4 Supporting Technology

The sections that follow describe technology shared among all components of the InterMail messaging system.

1.4.1 Multi-Threading

All InterMail servers are multi-threaded to enhance performance. Multi-threading allows individual InterMail processes to handle simultaneous message delivery and retrieval requests. The result of multi-threading is high message throughput.

Multi-threading also allows a server to take advantage of multiple CPUs on a single machine. Performance will increase as system CPU resources grow. In addition, the number of threads each process may use is configurable, allowing you to fine tune your system for the most efficient use of system resources.

1.4.2 Remote Method Execution (RME)

InterMail servers communicate using a distributed object protocol called Remote Method Execution (RME). This communication method is designed to facilitate the exchange of data at the object level. For example, if a server asks the MSS to retrieve a message, it replies by sending an object representing the message.

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

1.4.3 Event Logging and Statistics

All InterMail server processes share a common logging mechanism. This mechanism can be used to produce standard logs which record events as they occur. It can also produce statistics logs which indicate activity over time. These logs can be used to monitor the overall performance of the system for capacity planning and fine tuning.

1.4.4 Administration Commands

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

1.5 InterMail Features and Benefits

InterMail encompasses a rich feature set, whose highlights are described in the following sections.

1.5.1 Native Internet Standards Support

InterMail is a native Internet messaging server. It supports Internet messaging protocols at its core, without the use of gateways or connectors to convert to proprietary protocols or message formats.

1.5.2 Scalability via Distributed Architecture

All mail servers perform the basic functions of message delivery, message storage, and message access. What sets InterMail apart is its unique architecture, with separate servers independently dedicated to each of these three functions.

This design provides flexibility as it allows any InterMail installation to scale independently along three vertical axes (as depicted in Figure 2). As a result, InterMail supports customized scaling to suit the unique needs of a particular user base, with hardware and software added to the system as the user base expands.

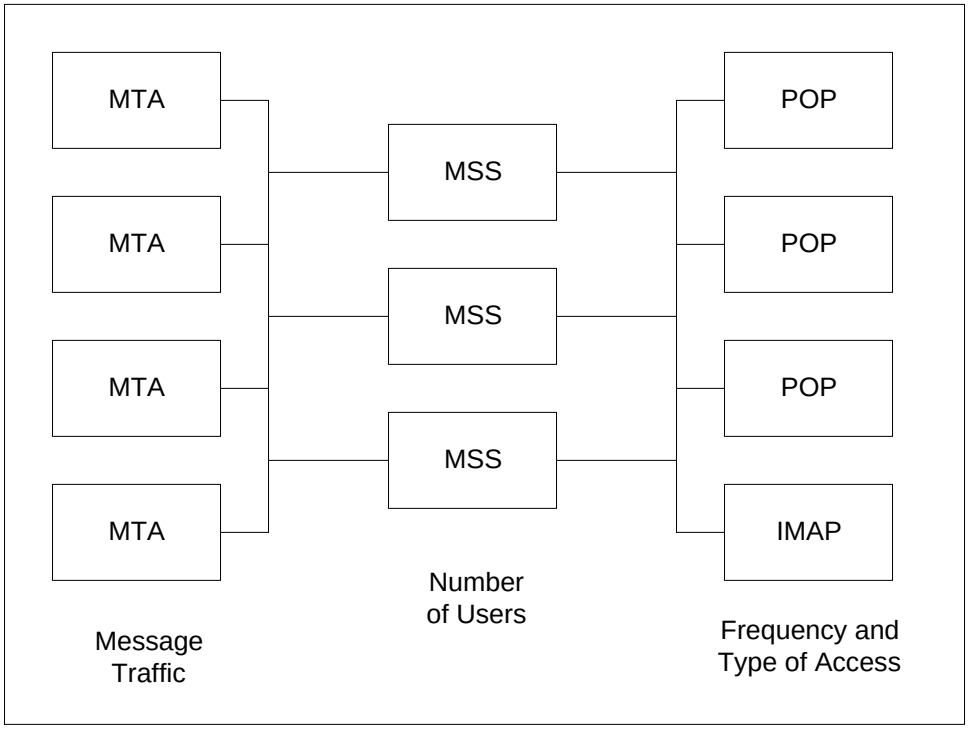

Figure 2. Example of a Distributed InterMail Installation

The variables for scaling are based on server type and correspond to the three axes in the preceding illustration:

- **Message Traffic:** the volume of mail going in and out of the system
- **Number of Users**: the number of accounts in the Integrated Services Directory
- **Frequency and Type of Access**: how (and how often) users retrieve their messages

MTAs are responsible for the set of tasks governing incoming and outgoing message delivery. They allow the system to scale with respect to the number of messages delivered to and sent from the mail system. In other words, the number of MTAs you need depends on the volume of message traffic on your system.

The Message Store Servers are responsible for persistent storage of messages in users' mailboxes. The number of Message Store Servers required is determined by the number of users in the system. In typical configurations, a single machine will easily support hundreds of thousands of users.

The POP and IMAP servers are responsible for providing users with access to Message Store Servers via the POP3 and/or IMAP4 protocols. The number of POP and IMAP servers can be increased as the number of users expands. The frequency and type of access will determine how many machines you need to run these servers.

1.5.3 High Availability

High availability, the need to have a system up and running at all times, is also referred to as "24x7" access. InterMail supports multiple strategies to enhance high availability, as described in the sections that follow.

Server Redundancy

Server redundancy simply means that more than one server is available to fulfill a function. The first and third axes of Figure 2 (MTAs and POP/IMAP Servers) are designed to take advantage of server redundancy. The system is also configured so that there are more machines than necessary to handle expected traffic. In the event that one of the machines goes down, the other machines can pick up the load until the first machine is brought back on line. When a machine goes down, it is taken out of the DNS rotation; it is then reintroduced when it comes back up.

Failover

If an MSS and Queue Server machine fails, the alternate assumes the network identity of the failed machine and accesses its disk array through a second port. Such "failover" capability is recommended for the MSS and Queue Server machines because they host actual message data, information that must be accessible at all times. In order to insure continuous access, another active MSS and Queue Server machine should be designated as an alternate.

Failover is provided by third party software and is not part of InterMail itself. When you design your failover strategy, there are two aspects to consider. The first is monitoring your hardware so that failures can be detected. The second is creating a set of scripts that are executed if a failure occurs. These scripts should cause the switch-over to the alternate machine, which will connect to the failed machine's disk array (through a second port), and begin to handle the activity of the failed machine.

Note: *While Software.com recommends the use of a failover mechanism, it is not required by InterMail.*

Disk Mirroring

Disk mirroring is used to protect data integrity. All critical disks should be mirrored to insure that data will always be accessible. In the event of a disk failure, the mirror will immediately be used to give access to the data. 'Hot spare' disks will be available in the storage array and will be brought up to date with their mirrors. By using this mechanism, single points of disk failure are eliminated.

The MSS and, Queue Server machines, and the machine that hosts the ISD should always be backed up, as should the master Configuration Database on the Configuration Server. All other servers require only redundancy to provide continuous operation.

Online Operations

InterMail enables you to carry out crucial operations, such as account migration, system backups and configuration changes while the system remains online. The helps minimize maintenance downtime, during which service is unavailable to customers.

1.5.4 High Reliability

InterMail's high reliability features minimize the danger that any messages will be lost before delivery is completed.

Journaling

InterMail provides message-level journaling. All transactions to the Message File System are saved in a journal; every insertion, change and deletion to the Message File System is recorded. The journal can be played back to re-create a full image of the Message File System. InterMail's use of relational databases and message-level journaling provides for reliability and rapid system recovery. Relational databases provide transaction logging which enhances data integrity, robustness, and recoverability. Journaling provides for recovery in the event of a Message File System failure. When used in conjunction with full backups, these mechanisms provide full disaster recovery, insuring that no messages are lost.

Transaction-Based Communications

Communications between servers is transaction-based, meaning that an operation is not considered complete until the initiating server receives confirmation from the server to which it communicated. For example, when the MTA passes a message to the MSS, the operation is not considered complete until the MSS signals the MTA that it has accepted the message; this happens only after the MSS has successfully written the message information to the Message Store database and the Message File System.

1.5.5 Class of Service Support

. Classes of service can be used to bundle features in distinct packages for consumers. For example, a service provider may charge customers one monthly rate for e-mail accounts with a limited range of services (simple POP3 access, low mailbox quotas, etc.), and a second rate for e-mail accounts with a larger set of allowed services (IMAP access, higher mailbox quotas, security features, etc.). Classes of service allow you to manage these sets of features and associate them with e-mail accounts. See the *InterMail Integrated Services Directory Guide* for a complete description of class of service features.

2

Outline of Pre-Production Tasks

Installing the InterMail software is the first step in establishing your complete messaging system. 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 chapter discusses a variety of pre-production tasks. The tasks are grouped into five categories:

- Tasks that should be completed before installing InterMail.
- Basic integration tasks. These tasks are very broad in nature and involve your operation as a whole, rather than your messaging system specifically.
- Tasks that pertain to InterMail directly. Completion of these tasks results in a custom configuration specifically suited to the needs of your organization.
- Tasks required to add account and domain information to the system.
- Tasks to test the procedures that you have established before going on line.

2.1 Pre-Installation Planning

This chapter assumes prior installation of the InterMail software, and therefore prior consideration of any pre-installation issues. The list below provides examples of issues you should have addressed in planning your installation.

Note: *For further information on planning your installation, refer to the* InterMail Installation Guide.

Determine server configuration

InterMail's distributed design allows for a variety of server configurations. As part of the installation process you should have established the number of servers required (both in total and by type) and the distribution of those servers on available hardware. In making those determinations, issues such as your specific requirements for security and high availability should have been addressed.

Define storage requirements

InterMail is intended for use in very large installations, sites that handle millions of messages for millions of users. Mail processing on this scale involves a substantial number of files for which a significant amount of disk space may be required.

Before beginning production, you should make certain that you have sufficient free disk space to meet the needs of InterMail's various servers and file systems. Chapter 9 of this manual offers an in-depth discussion of managing disk space requirements for InterMail.

2.2 Integration with Existing Systems

Certain tasks need to be considered when integrating InterMail into your existing operations environment. Those tasks are listed in the sections that follow. Because these tasks are necessarily site-specific, detailed descriptions cannot be provided; instead, these issues are addressed in general terms that can be applied to all installations.

Create a log monitoring program

The content of the InterMail log files enables 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 own log file, and log rollover periods are configurable.

Log files are a comprehensive dump of data, designed for flexible reporting. While 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.

Refer to Chapter 8 of this manual for an overview of InterMail logging, and Chapter 13 of the *InterMail Reference Guide* for a complete listing of InterMail events.

Link InterMail to your SNMP monitoring station

InterMail supports additional system monitoring through inclusion of an SNMP Server that gathers useful system information and passes it to an SNMP monitoring station for real-time viewing.

If you wish to take advantage of InterMail's SNMP monitoring capabilities, you will need to link the SNMP server with your SNMP monitoring station. To assist you in this task, please refer to Chapter 9 of this manual for a discussion of configuring your SNMP monitoring station, and Chapter 10 of the *InterMail Reference Guide* for a description of server architecture.

Establish a connection to your provisioning system

To link InterMail accounts with related billing information, you'll need to integrate the content of the Integrated Services Directory (the central repository for InterMail account information) with your existing provisioning system.

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

Precise answers to these questions will depend on factors that are unique to each InterMail installation—issues such as available hardware and message traffic. But whatever the particulars, the surest way to minimize any potential disaster is by establishing a suitable backup and recovery policy, modifying it as needed, and making certain that it is enforced.

Chapter 10 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.

2.3 Custom Configuration

When your InterMail system was installed, a value was established for each available configuration option. You are probably unaware of how most values were set, as very few of them required direct selection on your part. Instead, most options were set to their initial values (the "starter" entry that is set for a particular configuration key when InterMail is installed). This method simplifies the installation process, but does not necessarily result in the ideal configuration for your site.

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

For instructions on editing configuration keys, please refer to Chapter 3 of this manual. For details on specific configuration keys, see Chapter 11 of the *InterMail Reference Guide.*

Determining a security policy

InterMail provides a number of security-related features you should thoroughly understand before going into production. It is not possible to offer a definitive set of rules on how to define your security settings since this would depend chiefly on your particular environment and security needs. However, Chapter 4 of this manual explains each security option in detail, and discusses the potential impact of different security settings on your mail service.

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

Set mail routing rules

Mail routing rules allow you proxy mail from an InterMail host to some other mail system. This is typically useful during the migration process, when accounts are being transferred from a legacy mail system to InterMail. (see the *InterMail Migration Guide).*

There are also a number of other mail routing policies that can be set, including the use non-authoritative and rewrite domains and header rewriting. You can add to these rules or change them at any time after going into production, but it may be useful to establish an initial set of rules in pre-production. For a full discussion of routing rules, see Chapter 5 of this manual.

Establish message quotas

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

There are three types of message quotas:

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

Setting per-account quotas is an ongoing administrative task, rather than a pre-production issue. On the other hand, the system-wide settings are important considerations for pre-production, since the choices you make will affect your entire system right from the start. See Chapter 6 of this manual and the *Integrated Services Directory Guide* for additional information about setting message quotas.

Examine the message aging policies

Your message aging policy determines how long messages are allowed to remain in a user's mailbox. This is important whether you enforce mailbox quotas or not. If quotas are enforced, and an aging policy allows messages to remain in users' mail boxes for too long, then busy mailboxes may fill up quickly, causing further messages for these users to bounce. If quotas are not enforced, then a strict message aging policy is still crucial, since a large number of messages on the Message Store Server may fill up disk space rapidly.

Separate controls are provided to enforce message aging policies for retrieved and unretrieved mail. During installation, you will be asked to specify the length of time unread messages can remain in a user's mailbox. Unread messages "older" than this will be removed from mailboxes.

When determining your message aging policy, you will need to consider both the needs of your users, as well as the resources you are able to allocate to message storage. For more detailed discussions of message aging, see Chapter 6 of this manual.

Set queuing policy

When InterMail cannot deliver a message immediately, it queues it for future delivery. Delivery of queued mail is re-attempted at regular intervals. InterMail allows you to determine how often it will re-attempt delivery of such deferred mail. You can also decide how long deferred mail can remain in the queue before it is considered undeliverable and returned to sender.

Before going into production, you will need to decide on a policy for how queued mail is handled. This is important because deferred mail that is allowed to remain on your system for too long can consume large amounts of disk space. Also, delivery attempts that are too frequent may vie for system resources required by other processes. On the other hand, if you bounce deferred mail too quickly, or allow too much time between delivery attempts, you may diminish quality of service to your users.

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

Review the welcome message

If specified, a welcome message can be inserted automatically in every new mailbox. Before going into production, you may want to create the welcome message for new users. A well-written welcome message can create a positive first impression and convey important information, such as: contact numbers and/or e-mail addresses for technical support, information on quota policies, and the URL for your organization's Web site.

You can use the `immsinit` command to create an administrative mailbox containing the welcome message. It is important to keep this message in a special administrative mailbox, since this will prevent its being deleted by your message aging policy. Use of the `immsinit` command is described in Chapter 12 of the *InterMail Reference Guide*.

Set up cron jobs

Certain InterMail processes should be run periodically to perform routine system monitoring or maintenance tasks. One example is garbage collection, which does the actual removal of messages that have been flagged for deletion.

Most recommended cron jobs are established automatically at time of installation, however you may wish to review the schedules set for those utilities, and supplement them with additional cron jobs that perform tasks such as the archiving of mail-related log files.

Back up your newly configured system

Once InterMail is fully installed and configured, you should make a complete backup of your customized system. This will provide you with a "template" that contains a clean installation along with all your initial configuration settings. While this initial backup is not an absolute requirement, the safety margin it provides is well worth the time. Making a second copy of this backup and storing it in a secure, remote location is a useful added precaution. For a discussion of backup and recovery procedures, please see Chapter 10 of this manual. For a description of all configuration settings, see Chapter 11 of the *InterMail Reference Guide.*

2.4 Creating Account and Domain Information

Account and domain information is defined in the Integrated Services Directory (ISD). The following sections offer a brief description of the data contained in the ISD. For detailed discussions on how to create account and domain information, please see the *InterMail Integrated Services Directory Guide*.

Establish Domains

In addition to the default domain defined during the installation process, InterMail allows you to specify other domains over which your server can claim authority. If you want to provide mail service to users whose e-mail addresses are not within your default domain, you must identify those other domains to the system before mail processing begins. For example, if your 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 will need to define the `accordance.com` domain in the Integrated Services Directory.

If you fail to identify one of your desired mail domains, InterMail will be unable to deliver messages to users in that domain. In fact, you will be unable to create accounts with addresses that in are that domain. See Chapter 6 of this manual for a full discussion of domains.

Create Accounts

Like domain information, account information is created in the Integrated Services Directory. Accounts can be created using standard InterMail administrative commands, but you will probably want to use a provisioning system of some kind to simplify this task. There are four ways in which a provisioning system can interface with InterMail: CAPI, Perl, API, or via command line interface.

Create Class of Service Options

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

Note: *The names “class of Service A” and “class of Service B” are used only as examples. They are not necessarily real class of service names.*

An important pre-production task is to determine the various feature sets to use in each class of service you wish to offer.

Migrating existing accounts

If you have a database of account information from a legacy mail system, one of your most important pre-production tasks will be to “migrate” existing account information to the Integrated Services Directory, which stores all account information for InterMail and related Software.com products.)

Migration also addresses the issue of moving messages stored by your legacy mail server to mailboxes in the InterMail messaging system.

Migration is performed after InterMail is installed, but before you bring the system online. You will find a discussion of migration in the *InterMail Migration Guide.*

Note*: Migration is an intricate procedure which should never be performed without the support of Software.com’s Professional Services staff.*

2.5 Testing your Procedures and Policies

Having reviewed and refined all mail processing polices and performed the work necessary to integrate InterMail with the rest of your operating environment, you are now ready to test the system.

It is recommended that you create some “dummy” accounts and start sending them mail to check the validity of your configuration settings. Exactly what you should test depends on what settings you feel you must verify.

Example

To test your quotas, create a dummy account with a mailbox quota of 5MB, then flood the account with mail until it reaches its limit. The next message sent to that account should be bounced.

Note: *If a feature of the system is not behaving as expected, the first thing to do is double check the setting for that feature to make sure it’s right. If the setting appears correct and the system’s behavior is still not as expected, consult the InterMail documentation again to make sure you haven’t missed an important detail.*

Once testing is complete and you are satisfied that the system is operating as desired, full scale mail processing can begin.

3

Basic System Management

This chapter discusses the basic operations associated with administering InterMail. Anyone who administers InterMail in any capacity should be familiar with the concepts in this chapter, which include:

- Logging in to the InterMail system
- Starting up and shutting down servers
- Viewing and editing the configuration database

3.1 Logging In

The initial task required to administer InterMail is to log in to one of the UNIX systems on which InterMail is installed.

The InterMail User

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

Note: *All hosts in the system include a complete set of utilities for administering InterMail, so you can accomplish administrative tasks from any InterMail host.*

Executing Commands

The assorted utilities used to administer InterMail from the UNIX command-line are stored in the `bin` and `lib` directories in the InterMail user's home directory. These directories are added to the user's path during installation, so you can execute all InterMail commands 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 (venus)

login: imail
Password:

venus% imservdisplay
......
```

Figure 3. Executing an InterMail administrative command.

Throughout this manual, examples are given for executing specific InterMail commands. These examples assume that you have already logged in as the InterMail user; if you have not, then you must do so before executing the command(s).

3.2 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.

3.2.1 Overview

All InterMail servers can be started up, shut down, or restarted from any host in the system. This centralized control of servers is facilitated by a Manager Server, which runs on each host in the InterMail system and handles startup and shutdown instructions for all other servers on that host.

Manager Server

All hosts which run at least one InterMail server also run a Manager Server. This server is responsible for receiving startup and shutdown instructions for the InterMail servers on that host. By accepting these instructions from any host in the system, the Manager Server allows the administrator to control all InterMail servers from a single location.

When startup or shutdown instructions are issued by the administrator, these instructions are passed to the Manager Servers on the affected hosts. Each Manager Server then carries out whatever operation was requested, starting up and/or shutting down the specified servers.

Note: *The Manager Server is described in detail in Chapter 9 of the InterMail Reference Guide.*

Types of Startup and Shutdown Operations

There are several different methods for starting, stopping, and restarting InterMail servers:

- **Starting**. Starting up a server means executing the process associated with the server. There is only one type of startup for InterMail servers. When servers are started up, they refer to their local copy of the Configuration Database for relevant configuration data, and then go “online” and can begin receiving client connections and processing operations.
- **Stopping**. Stopping a server means causing the process to exit as soon as possible, but in an orderly fashion. Client sessions will be interrupted by the server shutdown, but meaningful error or status messages will be given to clients that are disconnected.

- **Draining**. Draining a server means causing the process to refuse to accept any new client connections, while completing those still in process. When all transactions are finished, the server then exits gracefully (that is, without terminating any client connections). For example, when a POP Server is drained, no new connections from POP3 clients are accepted; the server waits for all existing client connections to be closed, and when no more clients are present, the POP Server closes its connection to the MSS and the process exits.
- **Killing**. Killing a server means using the utmost force to cause the process to exit. This is equivalent to using the UNIX `kill -9` command, and does not allow the server to gracefully close its connections. This method of shutdown is not recommended and should be used only when absolutely necessary.
- **Restarting**. In addition to startup and shutdown options, there are commands which both shutdown and immediately restart InterMail servers. These restart options are simply variations of the three shutdown methods which also execute a start of the appropriate servers when shutdown is complete. The available types of restart are: drain and start, stop and start, kill and start.

imctrl Syntax and Options

The administration command responsible for starting up and shutting down InterMail servers is `imctrl`. 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 Servers then execute the appropriate startup or shutdown operations for the affected servers.

The basic syntax for using `imctrl` is:

```
imctrl [-verbose] [-dryrun] <task> ...
```

Where:

`-verbose`	Causes all program output to be printed to standard output and/or standard error.
`-dryrun`	Causes the program to show you the results of your `imctrl` commands, but without executing the commands.
`task`	Defines a specific startup or shutdown command to be executed by `imctrl`.

The `<task>` parameter of `imctrl` includes one or more host names, an operation, and one or more server types:

```
<host>[:<host>:...] <operation> <server>[:<server>:..]
```

The following table defines these `imctrl` task parameters:

`host`	The host name of the target system. Multiple host names can be specified, each separated by a colon (:). To specify all InterMail hosts, use the value `allhosts` for this parameter. To specify the host one which imctrl is being executed, use the value `localhost`.

`operation`	The type of startup/shutdown operation performed. The option for starting up a server is:
	`start`
	The values used to shut down servers are:
	`stop`
	`drain`
	`kill`
	`exit`
	The available restart options are similar to the shutdown options, but restart the server(s) immediately after shutdown:
	`stopStart`
	`drainStart`
	`killStart`
	`exitStart`
	`restart`

`server`	The server component being started/stopped. The available values are: `mta` — MTA `mss[.<number>]` — MSS `popserv` — POP Server `imapserv` — IMAP Server `imdircacheserv` — Directory Cache Server `imqueueserv` — Queue Server `imconfserv` — Configuration Server `immgrserv` — Manager Server snmpdm — SNMP Server httpd — InterManager web server `allservers` — all servers but Manager Servers mailservers — all InterMail servers (MTA. MSS, POP Srerver, IMAP Server, Directory Cache Server, Queue Server).

Note: *Multiple task arguments can be included in a single execution of imctrl. For example, you can use a single invocation of this command to shut down a server on one host while starting up a server on another host.*

The following sections demonstrate using `imctrl` to startup, shutdown, and restart InterMail servers.

3.2.2 Starting Up

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

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

For example, to start all InterMail servers on all hosts, enter the following:

```
imctrl allhosts start allservers
```

To start all servers on a particular host (in this case, `pluto`), include the host name before the `start` parameter:

```
imctrl pluto start allservers
```

Finally, to start one or more specific servers on a particular host, include both a host name and the server type(s). For example, to start the IMAP and POP servers on a particular host (again, `pluto` in this case), you would enter:

```
imctrl pluto start popserv:imapserv
```

3.2.3 Shutting Down

To shut down one or more InterMail servers, execute `imctrl` with any of the available shutdown parameters (`drain`, `stop`, `kill`, `exit`).

Stopping

The stop method causes a server to immediately shut down regardless of the presence of client connections, but terminates client connections with meaningful error or status messages. This is the most typical method of shut down.

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

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

For example, to stop all InterMail servers on all hosts, enter the following:

```
imctrl allhosts stop allservers
```

To stop all the servers on a particular host (in this case, `pluto`), include the host name before the stop parameter:

```
imctrl pluto stop allservers
```

Finally, to stop a particular server on a particular host, include both a host name and a server type:

```
imctrl pluto stop imapserv
```

Draining

As described in Section 3.2.3, the drain method of shut down allows the servers to be shut down without interrupting any current client connections. This is particularly useful for the POP Server and MTA components, which are the most visible to end users.

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

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

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

For example, to drain the POP Server on a particular host (in this case, `pluto`), enter the following:

```
imctrl pluto drain popserv
```

When this command is executed, the POP Server on `pluto` will continue to service connected clients, but will refuse to accept any new connections. Because there is no interruption to current connections, the end users who are connected to this POP Server when the command is executed will be unaware of the imminent shutdown of the server.

Killing

As described in Section 3.2.3, the kill method of shutdown is equivalent to using the UNIX `kill -9` command to kill a process. It causes server processes to exit immediately, and does not report error or notification messages to connected clients. Shutting down servers with the kill method is generally not recommended.

Note: *Servers can also be shutdown by executing* `imctrl` *with the* `exit` *option, which behaves similarly to* `kill`.

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

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

For example, to kill the MTA on a particular host (in this case, `venus`), enter the following:

```
imctrl venus kill mta
```

3.2.4 Restarting

Several configuration changes require InterMail servers to be shut down and restarted before the change can take effect. To restart an InterMail server that is currently running, you could execute two separate `imctrl` operations: one to shut down the server and a second to start it. However, `imctrl` also includes variations of the shutdown options that allow you to accomplish both of these in a single `imctrl` execution.

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

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

For example, to stop and restart all InterMail servers on all hosts, enter the following:

```
imctrl allhosts stopStart allservers
```

To drain and then restart the POP Server on a particular host (in this case, `pluto`), enter the following:

```
imctrl pluto drainStart popserv
```

To kill and restart the MTA on a particular host (in this case, `venus`), enter the following:

```
imctrl venus killStart mta
```

Again, the use of the restart options is identical to the use of the shutdown options described in the previous section.

3.3 System Configuration

One of the most important tasks involved with setting up InterMail is editing the configuration database. This section discusses the tasks related to viewing and updating the configuration database, and includes “how to” instructions for making configuration changes.

3.3.1 Overview

All InterMail components are controlled by a configuration database, which contains values for any and all configuration options 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, a single host maintains the “master” configuration database. On this host—known as the *master configuration host*—the Configuration Server responds to requests by individual servers for updated configuration information. Changes to the configuration database can be made only on the master configuration host.

Centralized Configuration

When configuration data is modified by the administrator, information about all potential changes is sent to each InterMail server. Each server then responds by indicating the impact of the new changes. For example, a server may report that it must be restarted to accommodate the changes. The administrator is informed of the effects of his/her changes, and has 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. These hosts then request a copy of the new `config.db` database from the Configuration Server. The new configuration database 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 Keys

The configuration database contains a series of individual data objects, known as *configuration keys*. Each key represents a specific aspect of InterMail server functionality or behavior.

Every entry in the configuration database is made up of four elements: a host, a server, the key name, and one or more values. When viewing or editing configuration database entries, configuration keys are displayed in the form

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

The `host` portion of a key defines the host to which the key applies. This allows you to define configuration options on a per-host basis. The wildcard (`*`) character is used to define entries in the configuration database that apply to all hosts.

The `server` portion of a key defines the InterMail server (POP Server, MTA, Directory Cache Server, etc.) to which the key applies. This means that keys can be defined to apply only to a particular server. The keyword `common` is used to define entries in the configuration database that apply to all servers.

The `keyName` is the literal name of the configuration key; this is the configuration option being defined for the specified host(s) and server(s). Some MTA configuration key names are themselves split into more than one hierarchical level. For example, the configuration entry

```
/*/mta/Error-Actions/smtpDnsBadConfig: [return]
```

defines a value for the configuration key `Error-Actions/smtpDnsBadConfig`.

The `value` portion of a configuration database entry defines the value assigned to the specified configuration key. Some configuration keys take only a single value, while others can include multiple values. All configuration key values are given between [square brackets].

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

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

In this example, the wildcard (`*`) character specifies that the entry affects all hosts. Because the server type is defined as `mta`, this entry affects only MTAs.

Be aware that configuration keys can be defined multiple times in the configuration database, and can be combined with any host or server type. This means that if your system consists of three hosts (in this example, `saturn`, `venus`, and `pluto`), you can define a configuration key three times – once for each host – instead of using the wildcard:

```
/saturn/mta/createsMboxes: [true]
/venus/mta/createsMboxes: [true]
/pluto/mta/createsMboxes: [true]
```

Because all InterMail hosts are accounted for here, this series of keys has the same effect as the previous example: they set "on-the-fly" mailbox creation for every MTA in InterMail. In fact, because this particular key (`createsMboxes`) affects only the MTA, you can define the server of these entries as `common` and achieve the same results as the previous examples:

```
/*/common/createsMboxes: [true]
```

or

```
/saturn/common/createsMboxes: [true]
/venus/common/createsMboxes: [true]
/pluto/common/createsMboxes: [true]
```

It is certainly less intuitive to define an MTA-specific key as `common` instead of `mta`, but it is important to understand that you can define the same key in several different ways. The way that you specify a configuration database entry – the host and the server for which it is defined – determines the precedence of that entry in the configuration hierarchy.

Configuration Key Hierarchy

As described in the previous section, you can define a single configuration key several times in the InterMail configuration database. This can result in configuration entries that overlap. For example, there may be an entry that defines a key value for all hosts, while a second entry defines a different value for the same key for only a particular host. In these cases, InterMail servers use a hierarchy to determine the appropriate value for a configuration option.

The InterMail configuration hierarchy goes from most specific to most general. Keys defined in the configuration database for a specific host and server are used first, while keys defined for all servers and hosts are used last. This allows you to specify system-wide configuration options, but define specific exceptions to those policies on a per-host and/or per-server basis.

InterMail servers go through the following steps to determine the appropriate value of a configuration key:

1. The server looks for a configuration database entry that defines the key explicitly for the server and the host on which it runs. These keys are defined as:

   ```
   /<host>/<server>/<key>
   ```

2. In the absence of a host- *and* server-specific key, the server next looks for the key to be defined for all servers on the host on which it runs. These keys are defined as:

   ```
   /<host>/common/<key>
   ```

3. If there are no host-specific entries for the configuration key, the next level in the hierarchy are server-specific entries that are defined for all hosts:

   ```
   /*/<server>/<key>
   ```

4. If an entry has not been found in any of the previous levels, the server looks for keys that are specified for all servers on all hosts. These keys are defined as:

   `/*/common/<key>`

5. Finally, if no entries exist for the key anywhere in the configuration database, the server uses the key's default value (if one exists).

For example, when an MTA on the host `pluto` is started up, this server 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 in the configuration key `defaultDomain`). When searching the configuration database, the MTA looks for the following configuration database entries:

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

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

Accessing the Configuration Database

The administration command used to access the configuration database is `imconfedit`. This command is used both to edit the configuration database and to view it in read-only format. When changes are made to the configuration database, `imconfedit` shows the administrator a review of his or her changes, and reports the impacts of the new changes on InterMail servers.

The following sections contain information on using `imconfedit` to edit or view the configuration database.

3.3.2 Modifying Configuration Data

This section deals with the tasks required to modify configuration database information. There are three basic steps to making such changes:

1. Execute `imconfedit` to display the current configuration database.
2. Use a text editor to manually enter new key values and/or entries.
3. Commit or cancel your changes, based on change impact information reported by individual InterMail servers.

Executing imconfedit

To execute `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 pluto
```

Figure 4. Confirmation that configuration server is active.

If the configuration server is running, `imconfedit` retrieves a copy of the master configuration database and open it in the system's default text editor (or `vi` if no editor is defined). The database appears as a long series of entries, each with a host, server, key name, and associated value(s):

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

Figure 5. The configuration database as seen in the system text editor. Note that only the beginning of the file is shown here; there are hundreds of configuration keys.

Once the configuration database is displayed in the text editor, edit its contents to change current configuration settings.

Making Configuration Changes

There are two basic types of configuration changes: modifying the value of an existing key, and creating a new configuration key entry. Although the first of these is by far the more common, you should understand the requirements and potential pitfalls of each.

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

To edit the value of an existing entry, you must first find the entry in the configuration database (using the text editor's search facility is typically the best method). Be aware that a single key may appear multiple times in the configuration database; to avoid mistakenly modifying the wrong entry or causing unintended configuration errors, you should search for all occurrences of a particular key before modifying its value anywhere in the configuration hierarchy.

When adding new configuration database entries, extra caution should be taken. In most cases, the addition of a new configuration key entry overrides a value set for the same key at a higher level in the configuration hierarchy, or is itself overridden by an entry lower in the hierarchy. Be sure to identify every instance of a key in the configuration database, and consider the impact of each of these entries, before adding another instance of the key.

All configuration key values are shown 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 an associated range of possible values, so be aware of the data requirements of a key before changing its value.

Additional Editing Tips

Aside from the configuration key and value syntax requirements, there are few restrictions on editing the contents of the configuration database. However, the tips that follow can simplify the task of editing the configuration database:

- Keys which signify boolean options (that is, keys that enable/disable a configuration option) can have any of four values: `yes`, `no`, `true`, and `false`. For the sake of simplicity, it is recommended that you consistently use only one boolean value pair (`yes`/`no`, `true`/`false`) for these keys.
- If you want a key to assume its default value, it is generally preferable to simply enter this value for the key under `/*/common`, rather than delete the key from the configuration database. Deleted keys will not be displayed in subsequent `imconfedit` operations, so the key (and its value) will be hidden from you.
- You should never edit configuration keys which are defined under the configuration path `/<host>/sysadmin`. These keys are set and used by InterMail, and should never be modified manually.
- When adding a second value to a key that can accommodate multiple values, you should place the new value—enclosed in [square brackets]—on its own line beneath the current value. For example:

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

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

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

Committing Configuration Modifications

When `config.db` is closed from the system's text editor, `imconfedit` displays a review of the configuration changes that you have made, and prompts you for how to proceed:

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

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

Figure 6. Reviewing configuration changes before committing them.

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 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 operation. The results of this query are reported to the terminal:

```
"
*                              Impacts of Changes
:
-------------------------------------------------------------------------
* ************************************************************************
*                          Servers Needing Re-starting
* ************************************************************************
* imapserv on venus (dirCacheConnections): requires a re-start
* imconfserv on venus (domainName): requires a re-start
* mss.1 on venus (dirCacheConnections): requires a re-start
* mta on venus (dirCacheConnections): requires a re-start
* popserv on venus (dirCacheConnections): requires a re-start
* ************************************************************************
*                        Parms Not Yet (if ever) Fetched
* ************************************************************************
* imconfserv on venus (journalDir): has not yet been fetched
-------------------------------------------------------------------------

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

Figure 7. Displaying the effects of configuration changes on InterMail servers.

As with the previous prompt, 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 decide to commit your configuration changes, `imconfedit` reports the outcome of the changes. If the configuration changes require that one or more servers be restarted, `imconfedit` also prompts you for whether the affected servers should be immediately restarted:

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

Figure 8. Prompt to automatically restart servers after committing configuration changes.

Although `imconfedit` can restart servers, this is typically a manual operation to be performed at an off-peak time or during a maintenance window. Refer to Section 3.2.4 for instructions on manually restarting servers.

Propagation of Configuration Changes

When changes are committed to the master Configuration Database, each InterMail system is alerted to the presence of configuration changes, and automatically retrieves a copy of this updated database from the Configuration Server. No manual intervention is required 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 that a server be restarted, then you must shut down and restart the server before the change will be incorporated. Until this happens, the server will continue to use the old value for the affected configuration options.

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

3.3.3 Viewing Configuration Information

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

4

Security

This chapter addresses security-related InterMail administration tasks. Before putting your mail system into full production, you should understand the features discussed in this chapter, which include:

- Mail relay prevention
- Mail blocking
- Connection dropping
- Message sidelining
- Mail filtering
- Authenticated SMTP
- Password protection
- SSL (Secure Socket Layer) authentication

4.1 Relay Prevention

Relay prevention allows you to protect the MTAs at your site from third-party relay, a tactic commonly used by distributors of "junk" e-mail. Third-party relay can disrupt mail activity at your site, so defining policies that prevent this type of relaying is an important step in preparing your site for full production.

InterMail includes several options for restricting the source and destination of relay mail. However, before attempting to set anti-relay policies, you should have a good understanding of the principles of mail relay. The following section provides an introduction to the basics of mail relay; if you are already familiar with the topic, skip ahead to Section 4.1.2.

4.1.1 Mail Relaying Basics

Relaying is often difficult for novice administrators to understand, because there is both "good" relaying and "bad" relaying. The desirability of relay is generally measured by the types of messages being relayed, and whether or not the administrator of the mail server that is used for the relay deems it acceptable.

Mail relay occurs every time that a message given to an MTA is destined for some other MTA. Users do this with their mail clients each time they send a message to a user whose e-mail account is stored 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 relaying; in fact, relay is one of the primary functions of an MTA. However, this functionality has been abused by distributors of junk e-mail, who often use 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 system's memory and processing capacity, disrupting normal mail operation for hours, and can cause the recipients to blame the unwanted messages on the site that was used to relay them. This type of relay—in which both the sender and the recipients of the mail are unrelated to the MTA used to relay it—is known as *third-party relay*.

For distributors of junk e-mail, third-party relaying allows them to avoid the cost of the hardware and software needed to support their level of mail activity. Relay also provides them with anonymity, which allows them to avoid backlash from users who do not want to receive junk e-mail. It effectively allows junk e-mailers to distribute messages at little or no cost to themselves, with the expense borne by the sites that relay and receive the messages.

Relay Prevention Tactics

Because relay is both a necessity and a vulnerability, preventing unwanted relay—while still maintaining mail server activity for your site's users—can be difficult. It is extremely important to understand the role of all MTAs in the system, their potential for being exploited via third-party relaying, and the flow of messages between them.

For instance, in a common mail server configuration, one MTA runs behind a network firewall accepting messages from users within the network, while a second MTA running outside the firewall accepts messages from the Internet. When messages are received by the external MTA for users in the site's local mail domains, it routes them to the internal MTA. In this case, the external MTA is neither the source nor final destination of the messages that it receives. Technically, the external MTA is used for third-party relaying. Configuring the external MTA to deny all third-party relay would prevent the internal MTA from ever receiving mail directly from the Internet.

In general, it is desirable to deny almost all third-party relay for MTAs that run outside of a network firewall, with specific 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 required only if you want to prevent users within your network from sending mail to particular domains.

4.1.2 Anti-Relay Options

InterMail includes a large number of configuration options for controlling the flow of relay mail. 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 and destination.
- **Unrestricted with exceptions**. This policy maintains a mostly open relay policy, with restrictions that prevent relay only if sent from specific hosts and/or users.
- **Completely restricted**. Because this policy restricts all relay, it causes the MTA to accept only messages that are addressed to users in local mail domains, or to other specific domains.
- **Restricted with exceptions**. This option allows you to maintain a mostly closed system, with relay allowed under certain circumstances. This is the most commonly used relay policy.

These relay policies can be defined for InterMail as a whole, or for each MTA independently. The level of relay restriction that you choose should be based on the intended use of the MTA and its role in your mail system.

Restricted vs. Prevented

Be aware that *restricting* relay in InterMail does not necessarily mean *preventing* relay. In InterMail terminology, “restricted” means that a message is suspected of being junk e-mail, and is a candidate for denial. Mail that is restricted by the assorted relay policies may still be allowed to be delivered 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 only for mail that is addressed to the designated domain. In general, you will restrict relay because you want to prevent it, but it is important to understand that relay is denied because of the combination of its source and destination.

When determining whether potential relay should be allowed, InterMail uses the following logic:

1. **Is this message addressed to a local mail domain?** If yes, accept the message; if not, proceed to step 2.
2. **Is the source of this message a system and/or user that is allowed to relay mail, as defined in the current relay 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 is allowed to receive restricted relay mail, as defined in the current relay policies?** If yes, accept the message and send it to the destination domain; if not, reject the message.

Relay Source Policies

The principal method of preventing mail relay in InterMail is to restrict the systems and users that are allowed to relay. You can define the source of restricted relay in four ways:

- **IP address**. When relay is restricted by IP address, any system whose IP address does not match a list of allowable addresses (or is defined in a list of denied addresses) is restricted from sending relay mail. IP addresses can be specified using a wildcard character, which lets you allow (or restrict) relay from all systems in a range of IP addresses.
- **E-mail address**. Relay can be restricted based on the e-mail address of the sender. In this case, relay is allowed if the address given by the `MAIL FROM` command is in the list of allowed e-mail addresses (or is not in the list of restricted e-mail addresses). When granting relay privileges by e-mail address, you can optionally choose to verify that each sender address exists in the Integrated Services Directory before allowing the relay.
- **Domain**. Similar to e-mail address restriction, restricting by domain also uses the `MAIL FROM` address of the message to determine relay privileges. Relay is allowed if the domain of the `MAIL FROM` address is in the list of allowed domains (or is not in the list of restricted domains). Domains can be specified using a wildcard character to define 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 (that is, the local portion of the address, given before the '`@`' character). This allows you to restrict relay by the same username from multiple domains.

Because e-mail addresses and domain names can be easily forged, restricting by IP address is by far the most secure of these methods. Whenever possible, define your relay restrictions by IP address.

Relay Destination Policies

If a message is restricted by relay source policies, the fate of that message then depends on the domain of its destination address. If the destination domain is one that is allowed to receive restricted relay mail, normal delivery of that message occurs; if the domain is among those that are barred from receiving restricted mail, the message is rejected.

As with relay source restrictions, allowable destination domains can be defined in multiple ways:

- **All domains (with specific exceptions)**. This allows delivery of restricted relay to occur unless the destination domain is specified in a deny list.
- **No domains (with specific exceptions)**. This prevents delivery unless the destination domain is specified in 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 your site is an `MX` backup).

Note: *If a restricted message is addressed to multiple recipients, and only a subset of them are in allowable domains, delivery is permitted for only those users, and the sender will be informed that the denied recipients did not receive the message.*

4.1.3 Configuration Options

InterMail relay policies are defined through a combination of configuration keys. 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 are prevented from relaying 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`	This key defines the minimum number of recipients that a message must have before relay restrictions are applied to it. This allows you to exempt messages from your anti-relay policies if they are addressed to one (or 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 will be allowed regardless of other relay policies. *If you intend to prevent all third-party relay, set this key to* `1` *to eliminate this exemption*.
`relaySourcePolicy`	This key defines the overall relay policy. This policy can be defined to allow all relay (`allowAll`), allow relay except from specific users/hosts/domains (`denyListed`), restrict relay except from specific users/hosts/domains (`allowListed`), and restrict all relay (`denyAll`). When defining relay restrictions, you should always start by setting the value of this key, or by verifying its value.

Defining Relay Sources

When relay is restricted, the following configuration keys define restrictions on the sources of relay. These keys define either the users/hosts/domains that are *allowed* to relay (when `relaySourcePolicy` is set to `allowListed`), or the users/hosts/domains that are *restricted* from relaying (when `relaySourcePolicy` is `denyListed`):

`relaySourceDomainList`	A list of domains to which the relay policy defined by `relaySourcePolicy` is applied. When the return address of a message (defined by the `MAIL FROM` command) includes a domain given here, the message will be restricted or allowed based on the value of `relaySourcePolicy`.
`relayLocalDomainsOk`	Option to include local mail domains in the list of domains specified by `relaySourceDomainList`. This option applies only if `relaySourcePolicy` restricts relay except for given hosts, domains, and users (`allowListed`). When this key is set to `true`, all messages whose return address includes a local mail domain are relayed without restriction.
`relayLocalMustExist`	Option to verify local senders before allowing relay. When this key is set to `true`, and the return address of a submitted message includes a local mail domain, InterMail confirms the existence of the sender in the Integrated Services Directory. If the address exists, relay is allowed; if the address does not exist, relay is denied.
`relaySourceLocalIpList`	A list of local IP addresses to which the relay policy defined by `relaySourcePolicy` is applied. When a message is received by an InterMail MTA from a host whose IP address is given here, the message will be restricted or allowed based on the value of `relaySourcePolicy` for this host. For example, say you have an MTA running on a single system that 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`, while 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 our 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` is applied. When a message is received from a host whose IP address is given here, the message will be restricted or allowed based on the value of `relaySourcePolicy`.
`relayNullRestricted`	Option for restricting messages that have a null (<>) return address. This option applies only if `relaySourcePolicy` allows relay except from specified host, domains, and users (`denyListed`).

Defining Relay Destinations

Relay mail that is allowed by the defined source policies is accepted regardless of its destination domain. However, delivery of restricted relay mail can be allowed or denied on a domain-by-domain basis (again, *restricted* does not necessarily mean *prevented*). The following keys define the domains that are allowed to receive messages that are restricted by relay source policies:

`relayDestAllowList`	A list of domains to which messages can be relayed, regardless of restrictions of the source of the messages. Including domains in this list implies that no other domains receive mail that is restricted by your relay source policies.
`relayDestDenyList`	Destination domains for which restricted relay mail is denied. Any message whose source is restricted (according to your relay source policies) is not sent to these domains. Including domains in this list implies that all other destination domains receive relay mail regardless of source restrictions.

When defining allowable destination domains, it is not necessary to specify local mail domains in `relayDestAllowList`. Local mail domains are assumed to be included in this list.

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 by* `relayDestDenyList`.

Responses to Denied Relay

When relay is denied by InterMail, the MTA must notify the connected client that the message was not accepted. The following configuration keys define this response:

`relayReplyCode`	The three-digit SMTP error code that is sent to the client when a relayed message is denied. By default, this value is 550, the standard SMTP code to indicate that the client operation was not successful.
`relayReplyText`	The error text that is returned 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.`

4.1.4 Sample Scenarios

This section contains a series of typical relay-prevention scenarios, with instructions for setting the required configuration keys. Although not all relay prevention strategies are discussed here, these examples illustrate the interaction between the many relay prevention configuration keys. An understanding of these keys, and the ways in which they work together, is a prerequisite for creating InterMail relay prevention policies.

Preventing Third-Party Relay

Because the source of third-party relay is not within your network, only an MTA that is outside a firewall is vulnerable to this type of relay. To prevent third-party relay, you should define configuration options for MTAs outside the firewall to establish the following policies:

- Mail received from systems within your network (as defined by IP address) is not restricted. This allows users who are connected to your network to send mail through these MTAs without restriction.
- All mail received from systems outside of your network is restricted.
- If restricted mail is addressed to an account within your local mail domains, delivery of this mail is allowed. This allows normal delivery of messages from throughout the Internet that are addressed to your users.
- All other restricted relay mail—which, by definition, is both sent from and addressed to users from outside of 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 where the MTA is running, and set the value of this key to `allowListed`. This sets a relay policy for this MTA that is restricted except for specified systems or domains. For example:

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

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

   ```
   /pluto/mta/relaySourceLocalIpList: [10.2.3.0]
                                      [10.3.21.0]
                                      [127.0.0.1]
   ```

3. Repeat the above steps for each MTA that is outside of the network firewall.

When these changes are committed to the configuration database, the specified MTAs subsequently reject all third-party relay attempts made by users outside of your network.

Note: *Modifying relay prevention configuration keys does not require the MTAs to be restarted before the changes can take effect.*

Allowing Delivery of Restricted Relay Mail

By default, restricted relay mail is denied regardless of its destination domain. However, there are conditions in which you may want to allow delivery of mail that was restricted. For example, if your site is an MX backup for another domain, you should allow mail addressed to that domain to be relayed.

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. Because delivery should be allowed regardless of the MTA, define this key for all InterMail MTAs (i.e., specify the key as `/*/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:

```
/*/mta/relayDestAllowList: [softwarenow.com]
                          [*.accordance.com]
```

The values in this example specify that all InterMail MTAs handle messages addressed to users in the domain `softwarenow.com`, and to any subdomain in `accordance.com` (or to `accordance.com` itself), even if these messages are restricted by the relay source policies.

4.2 Mail Blocking

Mail blocking is used to prevent specific users and/or systems from sending mail—*any* mail—to your site. Unlike relay restrictions (described in Section 4.1), mail blocking is an extreme measure, and is typically used only after a particular sender has flooded your site with unwanted e-mail. Because blocking is an all-or-nothing strategy for handling mail, care should be taken to avoid blocking legitimate mail.

4.2.1 Blocking Options

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

When mail is blocked, 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 (i.e., you may find that your site is inadvertently blocking mail from legitimate senders).[1]

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 Section 4.3), but allows the MTA to log information on each rejected message.
- **Sender's e-mail address**. This blocking method rejects a message if its sender address (given by the `MAIL FROM` command) matches a list of blocked addresses.
- **Sender's domain**. This option is similar to blocking by e-mail address, but blocks messages based on the domain portion of the sender's return address (given by the `MAIL FROM` command). This allows you to block any sender whose return address includes a particular domain.
- **Sender's username**. This option is also similar to blocking by e-mail address, but blocks messages based on the username portion of the `MAIL FROM` address (that is, the local portion of the address, given before the '`@`' character). This allows you to block mail from users who send mail from multiple domains using the same username.

Note: *Because e-mail addresses can be easily forged, blocking by IP address is the most secure of these methods.*

System-Wide Blocking vs. Per-Account Blocking

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

System-wide mail blocking causes messages to be blocked based solely on their sender information. In this mode, all incoming mail that matches any of the blocking criteria is rejected, regardless of the recipient address(es). This is the default blocking behavior.

Per-account mail blocking, meanwhile, allows the system's mail blocking policies to be overridden for individual e-mail accounts. This allows messages to be blocked or allowed based on both the sender and the recipient; if the sender is blocked, but the recipient account does not have blocking enabled, the message is delivered normally. For instance, you can define policies to block all mail sent from a particular host, but allow certain end users to receive mail from that host. Per-account mail blocking is among the InterMail class of service options, which are discussed in the *Integrated Services Directory Reference Guide*.

Note: *Per-account mail blocking simply allows accounts to accept or ignore the entire set of administrator-defined blocking policies. New blocking criteria—such as additional domains and addresses to block—cannot be defined on an account-by-account basis.*

---

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

4.2.2 Configuration Options

Mail blocking options are set by using configuration keys to enable particular blocking methods—by IP address, sender e-mail address, sender domain, or sender username—and specifying 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 want.

Blocking Mode

A single configuration key defines the mode of mail blocking:

`blockPerAccount`	This key determines whether the blocking policy is system-wide (blocking policies are applied globally to all incoming messages), or per-account (blocking policies are enforced 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 set to `false`.

Blocking by E-mail Address

There are three configuration keys that define mail blocking based on the e-mail address of the sender:

`blockAddresses`	This key enables (or disables) mail blocking by e-mail address. When this key is set to true, the `MAIL FROM` address of each incoming message is checked against the list of addresses defined in `blockTheseAddresses`. If the address matches a listed address, the message is blocked.
`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`).
`blockLocalNoAcct`	An option for verifying 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 set to true, the domain of the `MAIL FROM` address is checked against the list of local mail domains. If this address includes a local mail domain, the Directory Cache Server is asked to verify the existence of the sender address in the Integrated Services Directory. If the address does not exist, the message is blocked.

Blocking by Domain

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

`blockDomains`	This key enables (or disables) mail blocking by sender domain. When this key is set to `true`, the domain of the `MAIL FROM` address of each incoming message is checked against the list of domains defined in `blockTheseDomains`. If the address includes a listed domain, the message is blocked.
`blockTheseDomains`	A list of sender domains to be blocked. 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`	This key enables (or disables) mail blocking by client IP address. When this key is set to `true`, the IP address of a connected client is compared to the list of IP addresses defined in `blockTheseIps`. If the client IP address matches a listed address, all messages sent by the client are blocked.
`blockTheseIps`	A list of IP addresses to be blocked. IP addresses can include a zero (0) as a wildcard to specify all systems within a network (for example, `10.3.21.0`). You can also enter an IP address as a subnet to specify all systems within a range of IP addresses (for example, `10.3.21.16/24`).

Blocking by Username

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

`blockUsers`	This key enables (or disables) mail blocking by sender username. When this key is set to `true`, the username portion of the `MAIL FROM` address of all incoming messages is checked against the list of usernames defined in `blockTheseUsers`. If the address includes a listed username, the message is blocked.
`blockTheseUsers`	A list of usernames to be blocked (for example, `make-money-fast`).

Mail Blocking Responses

When mail is blocked, the MTA must notify the connected client that the message was not accepted. The following configuration keys define this response:

`blockReplyCode`	The three-digit SMTP error code that is sent to the client when a message 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 that is returned with the `blockReplyCode` value. This text informs the client of the nature of the message failure. By default, this message is "You are not allowed to send mail to <recipient>."

4.2.3 Sample Scenarios

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

Blocking All E-mail From a Particular System

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

1. Determine the IP address of the mail server that is responsible for sending unwanted e-mail to your site. This address can be found in the MTA log files, as well as in the `Received:` headers of the unwanted messages. Identify any other messages that have been sent from this site to ensure that blocking this host would not cause legitimate mail to be rejected.
2. Execute the `imconfedit` administrative command to edit the configuration database (as described in Chapter 3).
3. Locate the key `/*/mta/blockConnections` in the configuration database. If this key is set to `false`, change its value to `true`. If the key does not exist in the configuration database, add it as a new configuration key entry:

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

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

   ```
   /*/mta/blockTheseIps: [10.3.21.0]
                        [21.5.117.4]
                        [21.5.117.5]
   ```

5. Save your changes, committing the new blocking policies to the configuration database.
6. Carefully monitor all MTAs 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 issue for Internet service providers is the problem of users who send e-mail from non-existent addresses within the service provider's local domains. For example, 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 be sent to the non-existent address. This causes your mail system to handle undeliverable mail, which occupies system resources.

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

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

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

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

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

Blocking Mail From All Users in a Domain

Some distributors of commercial e-mail consistently use addresses from their own domains as the return addresses of junk e-mail. To stop messages from such groups from being delivered 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 this key is set to `false`, change its value to `true`. If the key does not exist in the configuration database, add it as a new configuration entry:

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

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

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

After these changes are committed to the configuration database, any message whose `MAIL FROM` address includes a blocked domain is rejected by the MTA.

4.3 Connection Dropping

Another tool for combating e-mail abuse is *connection dropping*. This feature allows the MTA to immediately terminate client connections that are made by a specific host, or which are being used to send mail to a large number of recipients. As with mail blocking, connection dropping is an extreme measure that can disrupt the flow of legitimate mail to your site, and should be used with caution.

The main use of connection dropping is to combat *denial-of-service attacks* on MTAs. 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, use connection dropping to terminate any future SMTP connections from the site from which the attack was launched.

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

When a connection is dropped, the MTA sends the client a 421 error response code to indicate that the connection was closed.

4.3.1 Configuration Options

The following configuration keys control connection dropping options:

`dropConnections`	This key enables (or disables) connection dropping. When this key is set to `true`, the IP address of every connecting client is compared to the list of IP addresses specified by `dropTheseIps`. Also, if there is a recipient limit defined by `dropMaxMessageRCPTs`, the message recipients are counted and compared against this limit. If the connection is in violation of either of these policies, the server immediately terminates the connection.
`dropTheseIps`	A list of IP addresses of systems that should be dropped. When `dropConnections` is set to `true`, the IP address of each connecting client is compared to the values in this list. If a match is found, the connection is immediately terminated. Use a zero (0) as a wildcard to specify all hosts within a network.
`dropMaxMessageRCPTs`	The maximum number of recipient addresses that can be given for a message before the client connection is terminated. If this key is set to `0`, connection dropping based on number of recipients is disabled.
`dropRCPTsReplyText`	The text returned with the 421 error code to clients whose connections are dropped. The default value for this message is "`Service unavailable.`"

4.3.2 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, execute the following steps:

1. Determine the IP address of the system responsible for the attack, which can be found in the MTA log files. Identify any other connections that have been made by this site to ensure that dropping all connections from this site would not cause legitimate mail to be blocked.
2. Execute the `imconfedit` administrative command to edit the configuration database (as described in Chapter 3).
3. Locate the key `/*/mta/dropConnections` in the configuration database. If this key is set to `false`, change its value to `true`. If the key does not exist in the configuration database, add it as a new configuration entry:

   ```
   /*/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 needed, with each enclosed in [square brackets]. For example:

   ```
   /*/mta/dropTheseIps: [10.3.21.0]
                       [21.5.117.4]
                       [21.5.117.5]
   ```

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

Dropping Connections From Distributors of Junk E-mail

Combating the distribution of junk e-mail is best accomplished with the mail blocking features discussed in Section 4.2. 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 are used to transmit junk e-mail, make the following changes to the configuration database:

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

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

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

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

3. Again, carefully monitor your MTA logs and the behavior of your system after setting a connection dropping policy. These policies can inadvertently cause disruption in the flow of mail to your site, so use caution.

4.4 Message Sidelining

An alternative to mail blocking is *message sidelining*, which allows you to "hold" messages that are likely to be unsolicited commercial e-mail. Unlike mail blocking, which rejects e-mail based on the source of the mail, sidelining stops a message based on characteristics of the message itself. Once sidelined, a message can be evaluated (either manually or by a filtering agent) to determine if it should be delivered to its intended recipients or discarded.

There are two message characteristics that can be used to sideline mail:

- **Total number of recipients**. Because junk e-mail is typically addressed to hundreds or thousands of recipients, any message that is 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 they are distributed.
- **Null return address**. Mail servers often send automatic responses—such as bounce notifications, auto-replies, etc.—using the null return address (<>). These automatic notifications are important to mail server operation, so mail from the null address is never blocked. However, this provides a loophole for junk e-mailers, who can use the null return address as the `MAIL FROM` address of their messages, thereby avoiding address and domain blocking rules. For this reason, the InterMail mail sidelining feature includes an option for sidelining messages from the null address that are sent to more than one recipient. Because legitimate automatic responses are always addressed to only one user, this allows potential junk e-mail to be sidelined while allowing legitimate notifications to be sent.

If message sidelining is enabled, and an incoming message is covered by either of these sidelining criteria, the message is moved to the `sideline` directory. You can then inspect these messages to determine if they should be delivered. If a message is considered legitimate, you can use the `immsgprocess` administrative command to reintroduce it to the system. Unwanted messages can be simply deleted from the sideline directory.

Note: *Mail in the* `sideline` *directory is not processed automatically. If you choose to sideline potential junk e-mail, be sure to review and handle the sidelined messages periodically.*

4.4.1 Configuration Options

The following configuration keys are used to define message sidelining policies:

`sidelineMessages`	This key enables (or disables) message sidelining. If this key is set to `true`, messages that violate the `sidelineNumRcpts` or `sidelineNullToMany` values are moved to the sideline directory.
`sidelineNullToMany`	Option to sideline all messages sent from the null address (<>) that are sent to more than one recipient.
`sidelineNumRcpts`	The number of recipients that a message must have to be sidelined as potential junk e-mail. If set to zero (the default value), this feature is disabled.

4.4.2 Viewing and Processing Sidelined Mail

When sidelining is enabled, and an incoming message violates one of the sidelining policies, the Header, Control, and Body files of the message are moved to the sideline directory.[2] For example, when a message with the Id

```
19970114235158898.AAA250@mars.software.com
```

is sidelined, the following files are moved to the `$INTERMAIL/queue/sideline` directory:

```
19970114235158898.AAA250@mars.software.com-Control
19970114235158898.AAA250@mars.software.com-Header
19970114235158898.AAA250@mars.software.com-Body
```

To review the contents of a message, simply use a text editor to open the header and/or body file. Once you have reviewed a sidelined message, you should either delete the mail or reprocess it to allow normal delivery. Deleting mail simply requires you to 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, execute the following command:

```
rm 19970114235158898.AAA250@mars.software.com-*
```

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 it will be processed normally by the MTA. Meanwhile, the Body and Header files of the message are moved 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 19970114235158898.AAA250@mars.software.com-Control
```

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

---

[2] For an introduction to Control, Header, and Body files, refer to the discussion of message files in Chapter 7.

4.5 Mail Filters

InterMail allows you to extend its relay prevention, mail blocking, and message sidelining features through the use of custom mail filters. These filters are run against all incoming mail, and can specify that the mail be rejected, bounced, sidelined, forwarded, thrown away, or delivered normally, based on its content. Filter actions can depend on the presence or absence of certain headers, the sender, the recipients, or any other text contained in the headers or body of the message. Message content can be tested by searching for exact string matches, by using simple patterns, or by using complex regular expressions.

Mail filters are defined in the Configuration Database with the configuration key `incomingMailFilter`. Each MTA can have one associated `incomingMailFilter` key, which defines all of the filtering criteria used by that MTA.

4.5.1 Filter Syntax

InterMail mail filtering is based on the SIEVE filtering language. 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 Section 4.5.2 more information.
`action`	Defines an action taken against the message by the MTA. See Section 4.5.3 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. Blocks of syntax can be delimited by {curly braces}. 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>;
}
```

Logical Operators

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

`ANY-OF ( <test>, ...)`	Defines a logical OR. If any of the specified tests is true, the entire expression is true.
`ALL-OF ( <test>, ...)`	Defines a logical AND. If any of the specified tests is false, the entire expression is false.
`NOT <test>`	Defines an action taken against the message by the MTA. See the following section for the possible values for this parameter.
`TRUE`	Defines an action taken against the message by the MTA. See the following section for the possible values for this parameter.
`FALSE`	Defines an action taken against the message by the MTA. See the following section for the possible values for this parameter.

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

For example:

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

4.5.2 Tests

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

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

Number Expressions

The syntax of a number expression is:

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

Where:

`message-element`	Specifies an attribute of the message that is 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 `>=` greater than or equal to `<` less than `<=` less than or equal to
`number`	Specifies an integer value, with an optional unit of measurement. Because the `SIZE` element is measured in bytes, specifying a unit of measurement with a number value provides a convenient method to specify kilobytes, megabytes, or gigabytes. The possible values are: `K` kilobytes; value is multiplied by 1,024 `M` megabytes; value is multiplied by 1,048,576 `G` gigabytes; value is multiplied by 1,073,741,824

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 is expressed as a string. The possible values are: `HEADER "<header>"` the value of the specified header `HEADER ("<hdr>", ...)` the value of the specified headers `SENDER` the full sender address `SENDER.DOMAIN` the domain of the sender address `SENDER.LOCAL-PART` the username of the sender address `RECIPIENTS` all recipients of the message `BODY` 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
	All of these operators can be used with the `-NOCASE` suffix to indicate a case-insensitive query. For example:
	`CONTAINS-NOCASE`
	`IS-NOCASE`

`string`	Specifies the text string to which the message element is compared, and is 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 `*` and `?` to find partial matches. In the case of `CONTAINS-RE`, this value is 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 are not present, the expression is false. For example:

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

4.5.3 Actions

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

`KEEP`	Causes the message to be delivered normally.
`TOSS`	Causes the message to be deleted.
`STOP`	Causes the mail filter to stop execution immediately. If no action has been run yet, then the message is delivered normally. Otherwise, the filter will complete the actions that have been executed thus far.
`FORWARD "<address>"`	Causes the message to be forwarded to a specific address. Multiple `FORWARD` actions can be included in a filter. If `FORWARD` is specified along with `KEEP`, then the mail is both forward and delivered normally.
`SIDELINE "<reason>"`	Causes the message to be sidelined. (See Section 4.4 for information on sidelined mail.)
`REJECT ["<reason>"]`	Causes the message to be rejected by the server. This typically means that the sending client will bounce the message back to the sender. The text entered for the "`reason`" is passed back to the client in the SMTP response to the DATA command.

`BOUNCE ["<reason>"]`	This action operates in two different modes, based on the value of the configuration key `blockPerAccount`. If this key is set to `false`, `BOUNCE` operates identically to `REJECT`, and causes the message to be rejected. However, if this key is set to `true`, the MTA checks account information for each recipient to determine if the account is using the MTA Filtering class-of-service option. If Filtering is enabled for the recipient, then the message is rejected for that recipient. If the account does not use Mail Filtering, then normal delivery occurs for that recipient.
	
	If the mail is rejected, the bounce message that is returned to the sender is:
	
	`An incoming mail filter has determined that this message should not be delivered.`
	
	`<reason>`

Any filter actions other than normal delivery (`KEEP`) are logged, and MTA statistic files will reflect the number of times that each filter action has been taken.

4.5.4 Sample Filters

The following examples illustrate simple mail filters. Because InterMail sites typically have a 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 "*@bar.com" BOUNCE;
```

If an incoming message includes a recipient address in the domain `bar.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("Is-Junk:") IS-NOCASE "yes" 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 it 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`, then 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("Is-Junk:") is-nocase "yes" TOSS;
   ```

   If the message includes the header `Is-Junk`, and the value of this header is `yes` (regardless of case), then 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 message is deleted from the system (`TOSS`). If not, the filter continues with the next test

5. The fifth line specifies that if the message has passed steps 2-4, it should be sidelined:

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


4.5.5 Verifying Filters

Because they are defined in the configuration database, mail filters are added to InterMail by using `imconfedit`. However, `imconfedit` performs no validation checks on mail filters, so it is important that you verify your filters before using them. The administrative command `imfiltercheck` provides this verification.

The usage of this command is:

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

Where:

`-v`	Validates that one or more filters (as specified in files) are valid.
`-vc`	Validates that a filter (as 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 validate.
`key`	Specifies a configuration key that contains the filter to validate. The value of this parameter is typically `incomingMailFilter`.
`message-file`	Specifies a message to execute the filter on.

Examples

1. To validate a filter that is 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 they are defined correctly. If they are not, the line number and position of the first error is reported.

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

   ```
   imfiltercheck -vc incomingMailFilter
   ```

   This example checks the mail filter defined in the `incomingMailFilter` configuration key for the current host to determine if it is defined correctly.

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

   ```
   imfiltercheck -vi
   ```

4. 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`.

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

   ```
   imfiltercheck -es filter1 19970114235158898.AAA250@mars.software.com-Header 19970114235158898.AAA250@mars.software.com-Body
   ```

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

6. 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 is given from standard input.

4.6 Authenticated SMTP

An issue related to the sending of 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 the relay-prevention and mail blocking policies 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 is allowed to continue only if the given authentication data matches that of an existing account. When messages are transmitted, the sender address of the messages—both in the `MAIL FROM` command and `From:` header—are compared to the addresses associated with the account. If the sender address is not valid for the account (that is, the address is a forgery), the message is rejected.

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

4.6.1 Related Class of Service Options

SMTP authentication is set for users on a class of service level. The following class of service options are related to SMTP authentication, and can be enabled or disabled for each class of service:

- **Authenticated SMTP**. This option specifies that the user *must* provide authentication information when sending mail via SMTP. When this option is enabled, and the user submits mail to an MTA, the message is accepted only if the user provides the appropriate username and password information. If authentication information is not given (or is incorrect), the message is rejected.
- **SMTP with SSL**. This option is used only when the Authenticated SMTP option is enabled, and controls user access to sending e-mail via the SSL (secure) SMTP port.
- **SMTP**. This option is used only when the Authenticated SMTP option is enabled, and controls user access to sending e-mail via the standard (non-secure) SMTP port.

4.6.2 Configuration Options

The following configuration keys affect SMTP authentication:

`checkAuthentication`	The key enables (or disables) 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. If it is, then the MTA requires the sender to authenticate using the `AUTH LOGIN` command.
`requireSecureAuth`	This option does not cause authentication to be required, but rather, requires the user to have a secure connection. If this key is set to `true`, the MTA allows users to transmit the `AUTH LOGIN` command only if they are connected to the SSL port. If set to `false`, the MTA allows `AUTH LOGIN` regardless of whether SSL is enabled.

4.7 Password Protection

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

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

1. **Limited number of authentication attempts**. When a user has submitted more than a certain number of incorrect passwords, the client connection is terminated.
2. **Increasing delays after incorrect passwords are given**. If user authentication fails because of an incorrect password, the client is kept waiting for a defined number of seconds before a subsequent authentication attempt is allowed. With each failed attempt, the delay time increases.
3. **Sources of incorrect passwords tracked**. Hosts responsible for sending failed login attempts are tracked for a specific period of time. This allows the system to enforce increasing password delays on single users who attempt to guess the passwords of multiple accounts during the same transaction.
4. **Logging of password failures**. Several logging options allow for recording information about failed authentication attempts. This information alerts you to incidents of password guessing from particular systems or against particular accounts.

4.7.1 Formula for Authentication Delays

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

- The number of failed password attempts that have been submitted for this account.
- The number of failed password attempts that have been submitted 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 given in the following equation:

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

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

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

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:

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

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

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

These delay times continue to escalate until the configurable maximum delay time is reached. The delay time is reset to zero if a valid password is given for the account, or if the window for tracking specific IP addresses and accounts has expired (also set at a configurable interval).

4.7.2 Configuration Options

The following configuration keys control password protection policies used by both the POP Server and IMAP Server:

`maxBadPassword`	The number of failed authentication attempts allowed before the client connection is dropped.
`badPasswordDelay`	The number of seconds that notification of a failed authentication attempt is delayed. 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`).
`maxBadPasswordDelay`	The maximum delay time for authentication attempts. If the delay time reaches this limit, and subsequent failed authentication attempts are made, there are no further increase on the delay time.
`badPasswordWindow`	The number of minutes that users and IP addresses are tracked after an incorrect password has been given. 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 used to calculate password delay time).
`maxBadPasswordAddrs`	Each time a failed authentication occurs, InterMail stores the IP address of the client that issued the incorrect login information. This key controls the maximum size of this IP address list. By default, the size of this list is 10,240. When this limit is reached, the event is logged and the list size is reset to zero.
`maxBadPasswordUsers`	Each time a failed authentication occurs, InterMail stores the login name of the account for which the incorrect password was given. This key controls the maximum size of this login name list. By default, the size of this list is 10,240. When this limit is reached, the event is logged and the list size is reset to zero.
`maxPasswordFailures`	This key is used to generate log events. When a user submits this many failed login attempts during the number of minutes specified by `badPasswordWindow`, an `AcctBadPswdMaxFailures` event is recorded in the POP or IMAP server log.

4.8 SSL (Secure Socket Layer) Authentication

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

Note: *InterMail does not currently support SSL for the IMAP Server.*

4.8.1 Introduction to SSL

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 is shown in the following illustration:

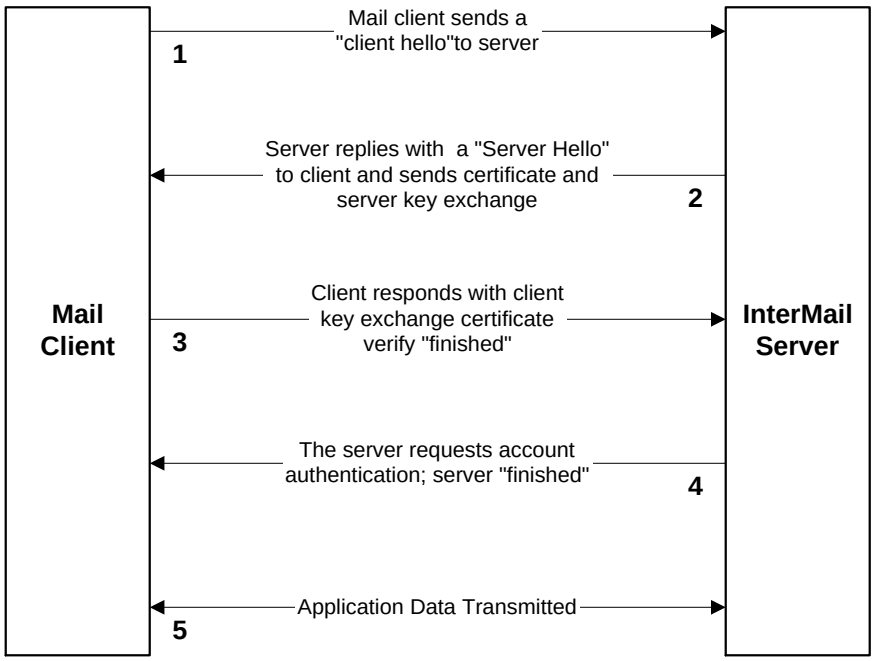

Figure 9. SSL Data Flow

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

4.8.2 Connections With SSL Clients

The configuration of SSL allows one additional SMTP port and one additional POP port to be assigned for secure SSL sessions. When an SSL-enabled client is detected, the SSL handshake procedure (verification) is performed on the defined SSL ports.

When enabling SSL, the `sslPop3Port` and/or `sslSMTPPort` configuration keys are used. These are the SSL-defined ports and will accept connections from SSL clients. By default, these keys are set to 995 for POP SSL and 465 for SMTP SSL. These port assignments should not be changed, as SSL clients will expect these assignments.

In addition to specifying the port numbers, information must be supplied on the certificate/key pair used for SSL sessions. As shown in Figure 9, the SSL process involves certificate exchange and communication of a "shared secret" by client and server. Therefore, a certificate must exist in the InterMail system; the file containing this certificate must be defined in the `sslTrustedCertPathAndFile` configuration key.

Also, a file containing an encrypted private key (defined in `sslCertChainPathandFile`) and the password to decrypt this private key (defined in `sslCertPassword`) must be set prior to SSL operation.

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

4.8.3 Transport Layer Security

Based on SSL, InterMail provides an additional extension to the SMTP server: Transport Layer Security (TLS). TLS provides private, authenticated communication over the Internet to help protect messages from eavesdroppers and attackers. TLS allows a SMTP server/client pair to activate SSL on the normally non-secure SMTP Port.

Several conditions must be met in order for TLS to be enabled in an SMTP session. First, as with SSL in general, the connected client must be SSL-compliant. Second, the successful negotiation of data (shown in Figure 9) must be completed. Lastly, the MTA configuration keys `SMTPPort` and `sslSMTPPort` must be set to the same port number (typically, port 25).

4.8.4 Configuration Options

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

`sslSMTPPort`	Defines the port for secure SMTP operation. If this key has a value of `-1`, or if the key does not exist, secure SMTP operation is disabled. If this key is assigned a valid, unused port number, the MTA will operate in secure mode on the specified port.
`sslPop3Port`	Defines the port for secure POP server operation. If this key has a value of `-1`, or if the key does not exist, secure POP server operation is disabled. If this key is assigned a valid, unused port number, the POP server will operate in secure mode on the specified port.
`sslCacheAgeSeconds`	Defines the lifetime (in seconds) of an SSL session cache entry. Entries older than the number of seconds specified here are expired.
`sslCacheBucketLen`	Defines the maximum number of entries in each bucket of the SSL session cache. If a bucket reaches the maximum number, then the first entries are removed to make room for new entries.
`sslCacheBucketNum`	Defines the number of buckets in the SSL session cache.
`sslTrustedCertChainPathAndFile`	Specifies 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. The encrypted private key and certificates are delimited by "`_____Begin`" and "`_____End`" PEM syntax. If this configuration key does not exist, of if there are errors reading the file, then POP and IMAP server operation on the secure port is automatically disabled.
`sslCertPassword`	The password used to decrypt the server private key specified in `sslCertChainPathAndFile`.
`sslUseSessionCache`	Option for using the SSL session cache.

5

Mail Routing

InterMail offers a variety of mail routing options. This chapter discusses the various options and their intended use. A thorough understanding of the subjects covered here will assist you in configuring your mail system to route mail appropriately.

This chapter covers the following topics:

- Domains and Accounts
- Envelope and Message Addressing
- Address Completion
- Rewriting Incoming Mail
- Rerouting and Rewriting Outgoing Mail
- Proxying Mail Between Hosts
- Delivery Status Notification

5.1 Domains and Accounts

Domains, accounts and addresses play a major role in mail routing. InterMail accounts and domains are maintained in the Integrated Services Directory (ISD), and are discussed in this chapter only as they apply to mail routing.

5.1.1 Local and Non-Authoritative Domains

The first InterMail objects that must be defined in the Integrated Services Directory are domains. In very general terms, a domain identifies a unique Internet site, for example:

`software.com`.

In InterMail, domains are “hosted” by the system. A “hosted” domain is simply one about which InterMail has some direct knowledge. Hosted domains are also referred to as “known domains.” A single InterMail installation may host just one domain, or it may host multiple domains. A single account can be associated with just one domain, or with any number of domains (through additional addresses).

Accounts cannot be added to your InterMail messaging system until the related domain information has been added to the ISD; however, before setting up your domains in InterMail, it is important to understand the difference between local, non-authoritative, and rewrite domains.

Note: *Domain information is stored in the Integrated Services Directory. For a discussion of how to create and manage domains and accounts in InterMail, see the* Integrated Services Directory Guide.

Local Domains

A local domain is one over which the InterMail system claims complete authority. All users within a local mail domain have accounts established in the Integrated Services Directory.

All mail addressed to any user in a local mail domain will be handled directly by the InterMail MTA, which will either deliver it to the appropriate mailbox, forward it to a forwarding address, or else return it to its sender if the intended recipient does not have an account in the Integrated Services Directory. Most InterMail accounts are associated with a local domain.

Creating a Local Domain

Domains are created with the `imdbcontrol` command using the `CreateDomain` option. Unless otherwise specified, `imdbcontrol` creates all domains as a local. The syntax for creating a local domain is:

```
imdbcontrol CreateDomain <DomainName>
```

Where:

`DomainName` The name of the local domain

To create a local domain for `software.com`, you would issue an `imdbcontrol` command as follows:

```
imdbcontrol CreateDomain software.com
```

Non-Authoritative Domains

A non-authoritative domain is a domain over which the InterMail system claims partial authority. InterMail accepts messages for all users in a non authoritative mail domain, but only some of those users have accounts established in the ISD.

Non-authoritative domains allow InterMail to accept mail for a domain, but relay it to another mail host if the user to whom a message is addressed does not have an account in the ISD. In order to establish a non authoritative domain, you must identify the relay host to which mail should be passed if it cannot be delivered directly.

When mail arrives for a known user in a non-authoritative domain, InterMail simply delivers it to the appropriate local mailbox. When mail arrives for an unknown user in a non-authoritative domain, InterMail passes the message on to the designated relay host.

It is assumed that the relayed message can be delivered by the mail server on the receiving host, although InterMail has no way of verifying this. All InterMail needs to know is the name of the host to which it should send the mail. From that point forward, the message becomes the responsibility of the other mail server.

For example, assume `accordance.com` is created as a non-authoritative domain, and is associated with a host named `pluto.accordance.com`. When mail arrives for an address within this domain (e.g., `jane.doe@accordance.com`), InterMail first checks for the existence of this address in the Integrated Services Directory. If a match is not found, the message is relayed to the mail server at `pluto.accordance.com`.

Note: *When mail addressed to a non-authoritative domain is relayed, the operation affects mail routing only. No modifications are made to the message's original address information because of this routing change.*

Creating a Non-Authoritative Domain

The syntax for creating a non-authoritative domain is:

```
imdbcontrol CreateDomain <DomainName> nonauth <relayHost>
```

Where:

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

To create a non-authoritative domain for `accordance.com`, issue an `imdbcontrol` command as follows:

```
imdbcontrol CreateDomain accordance.com nonauth pluto.accordance.com
```

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

```
imdbcontrol CreateDomain accordance.com nonauth pluto
```

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

5.1.2 Rewrite Domains

Rewrite domains are not really domains in the sense of local and non-authoritative domains; rather, they are rules for rewriting the recipient address of incoming mail. Essentially, a rewrite domain says something like: "send all mail for `<any.user>@accordance.com` to `<same.user>@software.com`."

No accounts may be associated with a rewrite domain, but a rewrite domain must be associated with a local or non-authoritative domain—that is to say, the rewrite domain rule must specify the name of the local or non-authoritative domain to which mail should be delivered.

When an incoming message is received, InterMail checks to see if the domain portion of the address has been defined as a rewrite domain. If it has, the address is rewritten to replace the original domain with the local or non-authoritative domain specified in the rewrite domain rule.

Creating a Rewrite Domain

The syntax for creating a rewrite domain is:

```
imdbcontrol CreateDomain <DomainName> rewrite <RewriteValue>
```

Where:

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

To create a rewrite domain whose rule rewrites `accordance.com` to `software.com`, issue an `imdbcontrol` command as follows:

```
imdbcontrol CreateDomain accordance.com rewrite software.com.
```

When mail arrives addressed to `<anyone>@accordance.com`, the domain portion of the recipient address will be rewritten to `software.com`. This means that a message sent to `john.doe@accordance.com` would have its envelope address rewritten as `john.doe@software.com`. InterMail would then attempt delivery to the rewritten address.

5.1.3 Accounts

The basic information associated with each user is contained in an InterMail account. This information includes the name of the user and the domain with which the account is associated. These are used to create the primary address for the user. Each account may also contain one or more additional e-mail addresses.

Note: *Accounts also contain a variety of other data types that do not pertain directly to mail routing. For a discussion of other types of account data, please see the* Integrated Services Directory Guide.

The following sections offer brief discussions of the different account address options.

Primary Addresses

Each InterMail account has at least one e-mail address, known as the *primary address*.

The primary e-mail address of an InterMail account is defined by combining the username of the account with its domain to form a *fully qualified SMTP address*.

The format for such an address is:

```
<any_user>@<any_domain.ext>
```

where

`any_user`	Any user's username, for example: `john.doe`
`any_domain.ext`	Any domain name (including the extension), for example: `accordance.com`, or `anyplace.net`

If an account has the username `john.doe`, and is created with the domain `accordance.com`, then the primary address of this account is

`john.doe@accordance.com`.

Using a postal mail analogy, the *username* portion of the address says *who* a message is for, while the *domain* portion of the address says *where* this individual lives.

A fully qualified SMTP address must be unique within a domain. There may be any number of people on the Internet whose usernames are `john.doe`, but there can be only one `john.doe@accordance.com`.

SMTP Aliases

SMTP aliases are additional e-mail addresses for an account. They 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@accordance.com` might have the following SMTP aliases:

```
jd@accordance.com
john@accordance.com
john@software.com
```

Mail addressed to an alias is delivered to the account exactly as if it had been addressed to the account's primary address. An account may have an unlimited number of SMTP aliases but (as with primary addresses) each SMTP alias must be unique.

Forwarding

Forwarding of e-mail 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 sends it to the new location.

Forwarding can be enabled or disabled for an account at any time. Accounts that use forwarding delivery must also be associated with one or more forwarding addresses. There is no limit to the number of forwarding addresses for an account.

One application of forwarding might be where an end user has two different accounts but would like to have mail addressed to both accounts delivered to a single mailbox for a time. For example, if John Doe goes on vacation, he may wish to have his work e-mail forwarded to his personal account. In this case, mail addressed to

`john.doe@accordance.com`

could be forwarded to (for example)

`jd@anyplace.net`

InterMail allows you to turn forwarding on or off at any time. In addition, it provides the option of receiving mail at *both* the primary address and the forwarding address. This would allow John Doe to receive the same messages at both

`john.doe@accordance.com`, and
`jd@anyplace.net`

5.1.4 Wild Card Accounts

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

Note: *Delivery to a wildcard account is, from InterMail's perspective, a last resort; it occurs only when the destination address of a message does not exist as an account primary address or as an SMTP alias.*

Wildcard delivery can be enabled or disabled for an existing local domain at any time. However, a domain can only have one wildcard account associated with it at any time. To change the wildcard account for a domain, you must disable wildcard delivery before defining the new account.

The most useful application of wildcard accounts is in servicing "vanity domains" for smaller companies. For example, adding a wildcard account for `accordance.com` would allow its customers to send e-mail to such intuitive addresses as `sales@accordance.com`, `techsupport@accordance.com`, or even `suggestions@accordance.com`.

It may be that e-mail accounts for some of these addresses already exist, but if the wildcard account were omitted and someone addressed a message to `support@accordance.com` (an account that does not exist) instead of `techsupport@accordance.com` (an account that does exist), the message would be considered undeliverable. With the wildcard account, the message will be delivered to some designated account at the `accordance` domain.

Note: *Wildcard delivery can be enabled or disabled for an existing local domain at any time. However, a domain can only have one wildcard account associated with it at any time.*

Implementing Wildcard Accounts

A wildcard account is a normal e-mail account, except for the fact that it receives all e-mail that is sent to unknown users within a local domain. For example, if the account `john.doe@accordance.com` is defined as a wildcard account for the local mail domain `software.com`, then any message sent to a non-existent address within `software.com` will be delivered to `john.doe@accordance.com`.

To set a wildcard account for a local domain, use `imdbcontrol` with the `SetWildCardAccount` option. This option takes two parameters: the domain name, and the primary SMTP address for an existing account that will be used as the wildcard account. The syntax is:

```
imdbcontrol SetWildcardAccount <DomainName> <PrimarySMTPAdddress>
```


Where:

`DomainName`	The name of the local domain for which the wildcard account will be created
`PrimarySMTPAddress`	The address of the e-mail account that will be used as the wildcard account for this domain.

Assume that John Doe has been given the job of handling mail for unknown users sent to the `software.com` domain. To create a wildcard account for John, you would issue an `imdbcontrol` command as follows:

```
imdbcontrol SetWildcardAccount software.com john.doe@accordance.com
```

Note: *This example assumes that John Doe already has an existing account in the* `accordance.com` *domain.*

What will happen now is that all messages for <`unknown_user`>`@software.com` will be delivered to John Doe's mailbox, along with his own e-mail. In cases where the volume of mail for unknown users is light, this might be fine. But if `software.com` receives a lot of mail for unknown users, John Doe's regular mailbox could fill up very quickly. And most of the mail might be messages that John Doe doesn't need or want to see pouring into his mailbox in a steady stream.

An reasonable alternative might be to create an account in the `software.com` domain called `unknown@software.com.` You would then create the wildcard account for `unknown:`

```
imdbcontrol SetWildcardAccount software.com unknown
```

Now, instead of having all mail for unknown users filling up his regular mailbox, John Doe could log into the `unknown` account and check these messages at his own convenience.

Unsetting Wildcard Accounts

If a wildcard account has been defined for a local domain, you can disable wildcard delivery by issuing an `imdbcontrol` command with the `UnsetWildCardAccount` option. This key takes only the domain name parameter as follows:

```
imdbcontrol UnsetWildcardAccount <DomainName>
```

Where:

`DomainName`	The name of the local domain for which wildcard delivery will be disabled.

To disable wildcard delivery for the `software.com` domain, you would issue an `imdbcontrol` command as follows:

```
imdbcontrol UnsetWildcardAccount software.com
```

5.2 Envelope and Message Addressing

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 two "addresses:" the `RCPT TO:`, which is the address of the intended recipient, and the `MAIL FROM:`, which is the address of the message's sender. InterMail uses these address to deliver messages to their proper recipients, however the person reading the message never sees them.

The address lines a reader sees in a message are called "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 are not used to route mail. Their chief purpose is to give the reader important information about who sent the message, who else received a copy, and also the subject of the message.

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

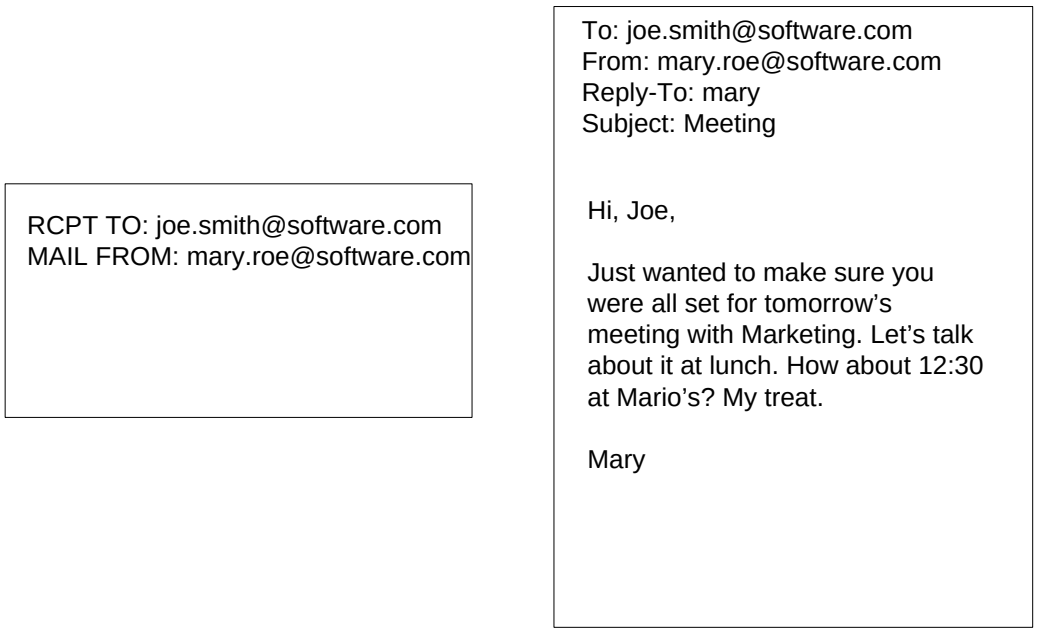

Figure 10. Envelope Addresses and Message Headers

In order for InterMail to deliver a message, the envelope's RCPT TO: address must be SMTP-compliant (e.g., `john.doe@accordance.com`).

InterMail provides configuration options for completing addresses and domains that are not fully qualified—i.e., they are either missing the domain name entirely, or the domain name is incomplete (e.g., because it consists of just a host name).

Among other things, 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 `accordance.com` domain, Mary would not necessarily have to enter Joe's full address in order to send him a message. Instead of addressing a message to `joe.smith@accordance.com`, she could simply send it to `joe.smith`. However, this will only work if InterMail's automatic address completion features are set.

5.3 Address Completion

As described in Section 5.1.3, a "complete" e-mail address consists of a username, plus a fully qualified domain name, in the form:

```
mary.roe@software.com
```

For a variety of reasons, some messages may contain addresses that lack all or a portion of the domain name, as for example:

```
mary.roe@
```

Normally, messages addressed in this way could never be delivered, but InterMail offers several options for completing such addresses automatically, whenever possible. But before discussing these options, it might be useful to briefly review how envelopes and messages are addressed.

There are two types of address that may require completion:

- Addresses that contain no domain information to the right of the `@` sign, as for example:

  `joe`, or
  `joe@`

- Addresses with only partial domain information to the right of the `@` sign, as for example:

  `mary@neptune`

InterMail provides the ability to automatically complete either or both of these incomplete addresses types.

Completing Addresses With No Domains

For addresses that don't include a domain at all, the default domain defined in the `defaultDomain` configuration key is appended in order to create an SMTP-compliant address. This is the most common type of address completion.

If the value for `defaultDomain` is defined as `accordance.com`, then an address in either of the following forms:

`john.doe`, or
`john.doe@`

would be automatically completed to read:

```
john.doe@accordance.com
```

If the `defaultDomain` key is missing, or its value is undefined (null), InterMail will attempt to complete the address by taking the values for the `confServHost` and `domainName` configuration keys and concatenating them as follows:

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

where

`any_user`	Any user's username, for example: `john.doe`
`confServHost`	The name of the host on which the InterMail Configuration Server is running
`domainName`	The domain name used for completing addresses with partial domains

Assume a scenario where there is no setting for `defaultDomain`, but the values for `confServHost` and `domainName` are set to `neptune` and `software.com` respectively. In this case, InterMail will complete John Doe's address as follows:

`john.doe@neptune.software.com`.

InterMail uses this as a backup method for address completion, just in case no `defaultDomain` is specified; however, there is no guarantee that mail for John Doe could be delivered to this address.

Note: *This is an unusual scenario, since in most cases, there will always be a valid setting for* `defaultDomain`.

Completing Addresses With Partial Domains

InterMail also offers an option for completing recipient addresses that contain only partial domains. For example:

```
mary.roe@venus
```

In this example, a message is sent to Mary Roe using only a partial domain (in this case, a host name) in the address.

To complete partial domain addresses, InterMail uses the value defined in the `domainName` configuration key. If the value for `domainName` were set to `software.com`, a message addressed to `mary.roe@venus` would be completed to read:

```
mary.roe@venus.software.com
```

5.3.1 Address Completion Options

The are some options for address completion that provide this feature with added flexibility. You can have InterMail:

- Complete all recipient addresses as needed
- Bounce all messages with incomplete recipient addresses
- Complete recipient addresses only when the sender is a local user

These methods apply both to completion of addresses with no domains and completion of addresses with partial domains.

The two configuration keys that play a role in address completion methods are `canonicalize` and `completionMethod`.

canonicalize

The `canonicalize` configuration key is used to complete an envelope's `MAIL FROM:` address if that address is incomplete. When `canonicalize` is set to `true`, incomplete `MAIL FROM:` addresses are completed using the value set in `defaultDomain` or `domainName`, exactly as specified earlier in Section 5.3. If `canonicalize` is set to `false`, incomplete `MAIL FROM:` addresses will not be completed.

completionMethod

The `completionMethod` configuration key is used to specify a completion method for any incomplete addresses other than the envelope's `MAIL FROM:` address. There are three methods to choose from: `default`, `sender` and `bounce`.

- The `default` setting means that incomplete addresses are completed using the values specified in `defaultDomain` or `domainName`.
- The `bounce` setting means that incomplete addresses are not completed at all. Instead, messages whose addresses are in the form:

  `john.doe`
  `john.doe@`, or
  `john.doe@neptune`

  are bounced back to the sender.
- The `sender` setting means that incomplete addresses are only completed if the `MAIL FROM:` envelope address contains a known domain (i.e., a domain that is specified in the ISD as local, non-authoritative, or rewrite)

  If the domain in the `MAIL TO:` address is known to InterMail, then the incomplete address is completed using the domain name that appears in the `MAIL FROM:` address.

Using the "sender" option

The `completionMethod` key's `sender` option is a bit more complex than `default` and `bounce`, so an example of how it can be used might be helpful.

Assume software.com and accordance.com are both known domains, and the value in the `completionMethod` configuration key is set to `sender`.

Mary Roe works for `software.com` with Joe Smith, whose e-mail address is `joe@software.com`. John Doe works at `accordance.com` with Joe Jones, whose e-mail address is `joe@accordance.com`. Both Mary and John send messages addressed simply to: `Joe.` The envelope information on Mary's message appears as follows:

```
RCPT TO: joe
MAIL FROM: mary@software.com
```

The envelope information on John Doe's message is as follows:

```
RCPT TO: joe
MAIL FROM: john@accordance.com
```

InterMail recognizes that both `RCPT TO:` addresses are incomplete, and attempts to complete them via the `sender` method specified in the `completionMethod` configuration key.

When completing the `RCPT TO:` address for Mary's message, InterMail compares the domain in her `MAIL FROM:` address (i.e., `software.com`) to its list of known domains. The domain `software.com` is recognized as a known domain, meaning that `sender`-based address completion should take place. Therefore, the domain name in Mary's `MAIL FROM:` address is used to complete the incomplete `RCPT TO:` address. The resulting envelope information is completed as follows:

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

However, when InterMail performs the same type of address completion on John's message, it uses the domain in John's `MAIL FROM:` address when performing the test for address completion and when completing the `RCPT TO:` address. As a result, the envelope information is completed as follows:

```
RCPT TO: joe@accordance.com
MAIL FROM: john@accordance.com
```

Despite the fact that both messages were addressed to `Joe`, they were completed differently based on the sender's `MAIL FROM:` address.

5.4 Rewriting Incoming Mail

Every message, whether addressed to a *local user* or destined for a *remote recipient* is initially considered incoming mail. Only after the rules for incoming mail have been applied are additional checks made to determine if a message's final destination is local or remote.

InterMail provides the ability to do both domain and header rewriting for incoming mail.

Header rewriting does not reroute mail in any way. It is used primarily to clean up the addresses readers see in messages, and also to hide proprietary origination addresses when necessary.

Domain rewriting, which modifies the domain portion of envelope addresses, reroutes mail destined for one domain to another domain.

The sections that follow discuss header and domain rewriting in detail.

5.4.1 Header Rewriting for Incoming Mail

Header rewriting for incoming mail affects message headers only, leaving the `RCPT TO:` and `MAIL FROM:` envelope addresses untouched. No matter which incoming headers are rewritten, messages are still delivered exactly as specified by the unaltered `RCPT TO:` address in the message envelope.

Identifying which headers should be rewritten

The first step in rewriting incoming headers is to tell InterMail which headers you want to rewrite. This is done using the `rewriteHeaderList` configuration key.

There are six header types, any of which can be rewritten:

```
To:
Cc:
Bcc:
Reply-To:
From:
Sender:
```

You can choose to rewrite one or any combination of these headers, or else you can do no header rewriting at all.

Rewriting Incoming Headers

To have InterMail rewrite the `To:` `From:` and `Cc:` headers, do the following:

- Use `imconfedit` to set the `rewriteHeaderList` configuration key as follows:

```
/*/mta/rewriteHeaderList:  [To:]
                           [Cc:]
                           [From:]
```

This setting does not necessarily mean that InterMail will *automatically* rewrite the `To:`, `Cc:` and `From:` headers for every incoming message. It simply means that these headers will be *eligible* for rewriting if certain other conditions are met. These conditions are described in the next sections.

Note: *A null value in* `rewriteHeaderList` *means that no header rewriting for incoming message will be done, regardless of any other settings.*

Selecting a rewriting method

Once you have selected the headers that will be eligible for rewriting (as defined in `rewriteHeaderList`), you must now select a rewriting *method* for these headers. There are two choices here. You can:

- Rewrite eligible headers using the primary SMTP addresses of local users (those users who have accounts on the InterMail System)
- Rewrite eligible headers using rules specified in a rewrite domain

You can elect to use one or both of these methods.

Rewriting headers with primary SMTP addresses

To rewrite eligible headers for incoming mail with local users' primary SMTP addresses, use `imconfedit` to set the value of the `rewritePrimary` configuration key to `true`.

Now, if a message for Mary Roe in the `software.com` domain arrives addressed to her alias address as follows:

```
mary@software.com
```

its `To:` header would be rewritten with Mary Roe's primary SMTP address, so that it reads:

```
mary.roe@software.com
```

Primary SMTP header rewriting can only occur if there is an account in the ISD for this user. If no account (and therefore, no primary SMTP address) is found, then no rewriting for eligible headers is done.

Rewriting headers with rewrite domain rules

To rewrite eligible headers for incoming mail using a rule defined in a rewrite domain, use `imconfedit` to set the value of the `rewriteDomains` configuration key to `true`.

Assume you had a rewrite domain whose rule specified that all mail addressed to any user in the `accordance.com` domain should be rewritten to that same user in the `software.com` domain. Now, if a message for John Doe arrives addressed as follows:

```
john.doe@accordance.com,
```

its `To:` header would be rewritten as follows:

```
john.doe@software.com
```

Remember that this form of rewriting affects only message headers, not the `RECPT TO:` envelope address.

Domain header rewriting can only occur if there is an applicable rewrite domain. If no rewrite domain is found in the ISD, then no domain rewriting for eligible headers is done.

Using both methods

If the values for `rewritePrimary` and `rewriteDomains` are *both* set to `true`, InterMail first checks to see if there is an account for this user. If there is, it rewrites eligible headers with the user's primary SMTP address and does no further rewriting.

If no account (and, hence, no primary SMTP address) is found, InterMail next checks for an applicable rewrite domain. If a rewrite domain exists, the domain portion of eligible headers is rewritten according to the rule specified in the rewrite domain. If no applicable rewrite domain is found in the ISD, no rewriting of eligible headers is done.

If the values for `rewritePrimary` and `rewriteDomains` are *both* set to `false`, then no header rewriting will be done, regardless of which headers are specified in `rewriteHeaderList`.

Note: *Both these methods of header rewriting are never done on the same header address. Primary rewriting is attempted first. If it succeeds, no further rewriting is attempted. However, individual headers within the same message may be rewritten differently. For example, the* `To:` *header might pass the test for primary rewriting, but the* `Cc:` *header may fail that test but pass the test for domain rewriting. Meanwhile, the Bcc header may fail* both *tests not be rewritten at all.*

Controlling "when" headers are rewritten

In addition to choosing a method for rewriting eligible headers (as explained in the previous section), you can also determine the circumstances under which eligible headers will be rewritten. You can:

- Rewrite eligible headers only for those messages that have been handled by less than a specified number of mail servers
- Rewrite eligible headers only if the sender of the message is in a known domain that is hosted by InterMail

Using MTA hops to limit header rewriting

In certain cases, you may wish to rewrite eligible headers only if a message has passed through fewer than a specified number of mail servers. Each time a message passes through another mail server, it is called an "MTA hop." The total number of MTA hops a message has taken is determined by the number of "received" lines it contains.

To rewrite only those eligible headers for messages that have been handled by less than a specified number of mail servers, use `imconfedit` to bring up the configuration database and change the value of the `rewriteMaxMtaHops` configuration key. For example, setting the key as follows:

```
/*/mta/rewriteMaxMtaHops: [2]
```

would mean that eligible message headers would not be rewritten if the message has passed through more than two mail servers.

Limiting rewriting based on whether the sender of the message is "local"

To limit eligible header rewriting to messages that originate from a local sender, use imconfedit to set the value of the `rewriteOnlyLocal` configuration key to `true`. With this setting, eligible header rewriting for incoming mail will limited those messages whose senders are in domains that are hosted by InterMail.

If this key is set to `false`, rewriting of eligible headers occurs regardless of whether the sender is known to InterMail or not.

Saving the Original Header Information

Another option for header rewriting allows you to save the original headers (in `X-Original` format) 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 set to `false`, original headers will not be saved.

Incoming message header rewriting example

At this point, it may be useful to illustrate a brief example of how these header rewriting features work together.

Assume that you want to rewrite only the `To:` headers for incoming mail. Furthermore, assume that you want to rewrite these headers with users' primary SMTP addresses, and that you will rewrite eligible headers only when message senders are local. Finally, you will not save the original headers as part of each message.

To achieve these results, you would use `imconfedit` to do the following:

1. Enter the eligible header in the `rewriteHeaderList` configuration key (e.g. `rewriteHeaderList [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`.

5.4.2 Incoming Mail Header Rewriting (Global View)

The following diagram provides a "bird's eye view" of how these features work together. The numbered steps indicate the sequence in which InterMail performs the various checks required to rewrite incoming mail headers.

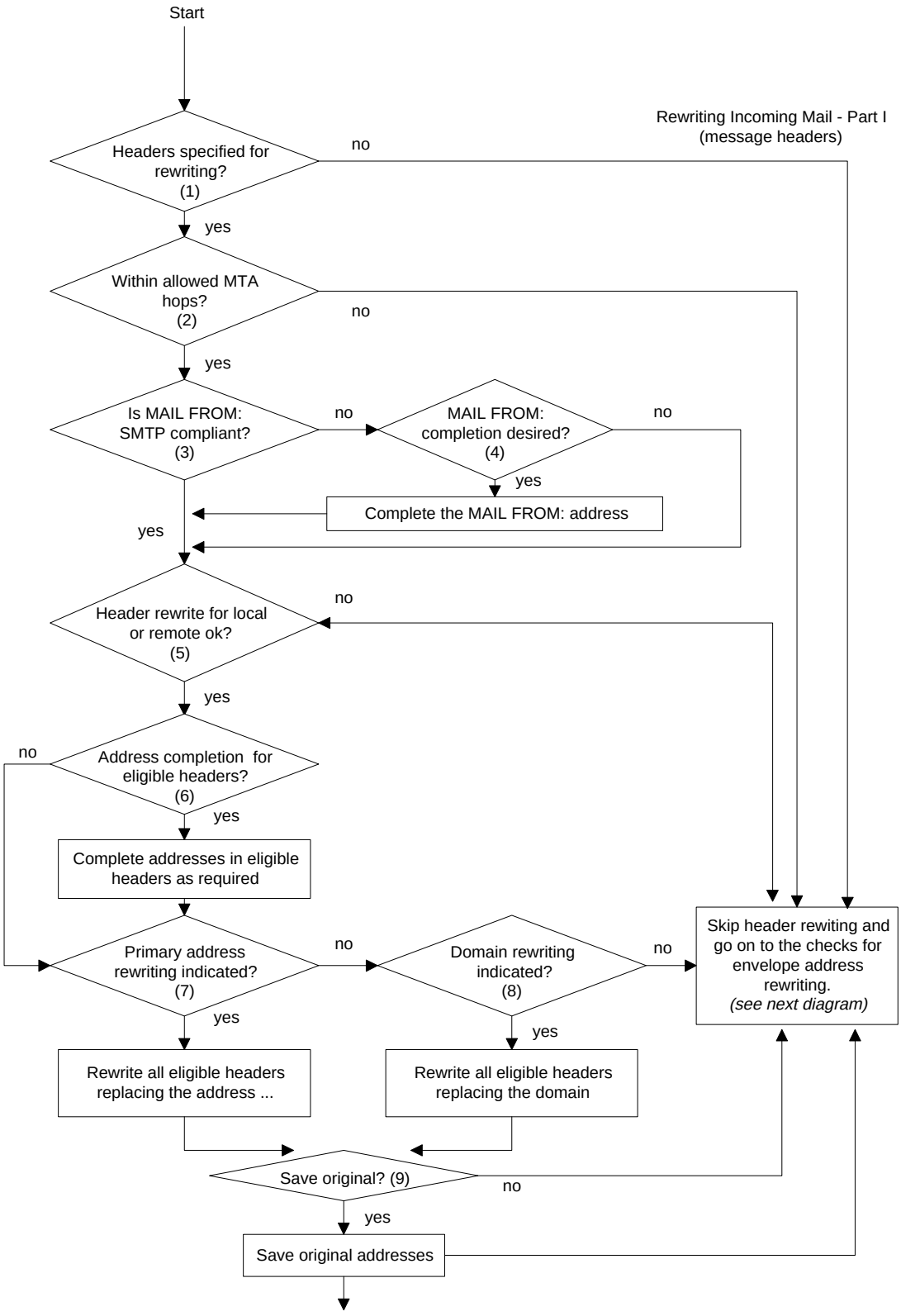

Figure 11. Header Rewriting for Incoming Mail

1. Check the `rewriteHeaderList` configuration key. This key indicates the headers on incoming mail that are specified (eligible) for header rewriting.
   - If one or more headers is listed in the `rewriteHeaderList` configuration key, those headers are eligible for rewriting. Continue with the header rewriting operation.
   - If the value of the `rewriteHeaderList` configuration key is null, no header rewriting will be performed. Go to Step 10.
2. Check the `rewriteMaxMtaHops` configuration key. This key restricts application of desired header rewriting on incoming messages. Compare the number of received lines in the message header with the value set in the `rewriteMaxMtaHops` key.
   - If the number of received lines is less than or equal to the value of the `rewriteMaxMtaHops` key, the header rewriting operation can proceed. Go to Step 10.
   - If the number of received lines exceeds the value of the `rewriteMaxMtaHops` key, the message headers cannot be rewritten.
3. Check the `MAIL FROM:` address on the envelope for SMTP compliant formatting.
   - If the `MAIL FROM:` address on the envelope is SMTP compliant, the header rewriting operation can proceed. Go to Step 5.
   - If the `MAIL FROM:` address on the envelope is incomplete, determine if address completion is desired. Go to Step 4.
4. Check the `canonicalize` configuration key. This key determines whether or not incomplete `MAIL FROM:` addresses should be completed.
   - If the `canonicalize` configuration key is set to false, address completion is not performed.
   - If the `canonicalize` configuration key is set to true, the `MAIL FROM:` addresses are completed (typically with the value in the `defaultDomain` configuration key).
5. Check the `rewriteOnlyLocal` configuration key. This key further restricts application of desired header rewriting for incoming mail.
   - If the value of the `rewriteOnlyLocal` key is set to false, continue with the rewrite operation.
   - 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, the header rewriting operation can proceed.
   - If the value of the `rewriteOnlyLocal` key is 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, no header rewriting is done. Go to Step 10.
6. Check if addresses in eligible headers require completion. Only the addresses which appear in eligible headers (those listed in the `rewriteHeaderList` key), are checked for address completion.
   - If the address in the eligible header is complete, address completion is not performed.
   - If the address in the eligible header is incomplete it is completed (typically with the domain identified in the `defaultDomain` configuration key).

Note: *At this point, the treatment of the various headers within a message may differ. The addresses in some eligible headers may require completion, while others may not; and still others may be ineligible for address completion (because they are not specified in the list of eligible headers established via the* `rewriteHeaderList` *key).*

---

7. Check the `rewritePrimary` configuration key. This key indicates whether or not primary address rewriting is desired for headers on incoming messages.

   - If the value of the `rewritePrimary` key is set to false, no primary address writing is done and all eligible message headers (those listed in the `rewriteHeaderList` key) are subject to a further test for domain rewriting (go to Step 8)
   - If the value of the `rewritePrimary` key is set to true, all eligible message headers (those listed in the rewriteHeaderList key) are examined for primary address rewriting.
     - If the address in an eligible header matches an address in any account in the Integrated Services Directory (whether it matches a primary address or SMTP alias address), the address in the eligible header will be replaced with the primary address for the account in which the match was found.(go to Step 9).
     - If the address in an eligible header does not match any address in any account in the Integrated Services Directory, the header will be subject to a further test for domain rewriting (go to Step 8).

Note: *Again, at this point, the treatment of the various headers within a message may differ. Some may pass this test and be rewritten, others that were eligible for rewriting may fail and be subject to further rewriting tests, and still others may be ineligible for rewriting of any sort (because they were not specified in the list of eligible headers established via the* `rewriteHeaderList` *key).*

---

8. Check the `rewriteDomains` configuration key. This key indicates whether or not domain rewriting is desired for headers on incoming messages.

   - If the value of the `rewriteDomains` key is set to `false`, no domain writing is done. Go to Step 10.
   - If the value of the `rewriteDomains` key is set to `true`, all eligible message headers (those listed in the `rewriteHeaderList` key) are examined for domain rewriting.

     Only those addresses (in designated headers) whose domain matches an established rewrite domain are subject to this type of rewriting. The domain used for rewriting is the replacement value indicated by the rewrite domain entry in the ISD.
     - If the address in an eligible header includes a domain that matches a rewrite domain established in the Integrated Services Directory, the domain in the eligible header is overwritten with the replacement value indicated for the rewrite domain (go 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 Integrated Services Directory, no domain rewriting is performed on the header (go to Step 9).

Note: *The various headers are considered individually. As a result, some may be rewritten while others are not.*

---

9. Check the `rewriteSaveOrig` configuration key. This key determines whether or not a record of the original header information should be saved.
    - If the value of the `rewriteSaveOrig` key is set to `false`, none of the original header information is saved (go to Step 10).
    - If the value of the `rewriteSaveOrig` key is set to `true`, a new `X-Header` is added with the original header information as its value. (go to Step 10).

All incoming mail is subject to checks for both header and domain rewriting. Once the check for header rewriting are complete, the checks for domain rewriting begin.

This scenario is continued in Section 5.4.4, beginning with Step 10.

5.4.3 Domain Rewriting for Incoming Mail

Domain rewriting for incoming mail affects only the `RCPT TO:` address in the envelope. If the `RCPT TO:` address is, in fact, rewritten, it means that the message will also be rerouted. This is very different from header rewriting for incoming mail, which *does not* in itself reroute messages.

Domain rewriting is done automatically, whenever a rule specified in one of the ISD's rewrite domains matches the domain portion of incoming message's `RCPT TO:` address. When such a match is found, the domain portion of the envelope's `RCPT TO:` address is rewritten, and the message is rerouted to the recipient in the new domain. See Section 5.1.2, and also the *Integrated Services Directory Guide* for discussions of rewrite domains.

Unlike other types of rewriting, there are no configuration keys to set for incoming mail domain rewriting. All that is required is a rewrite domain match in the ISD.

5.4.4 Incoming Mail Domain Rewriting (Global View)

The following diagram continues the scenario illustrated in Section 5.4.2, beginning with Step 10. The numbered steps indicate the sequence in which InterMail performs the various checks required for domain rewriting, *after* it has completed the checks for incoming mail header rewriting.

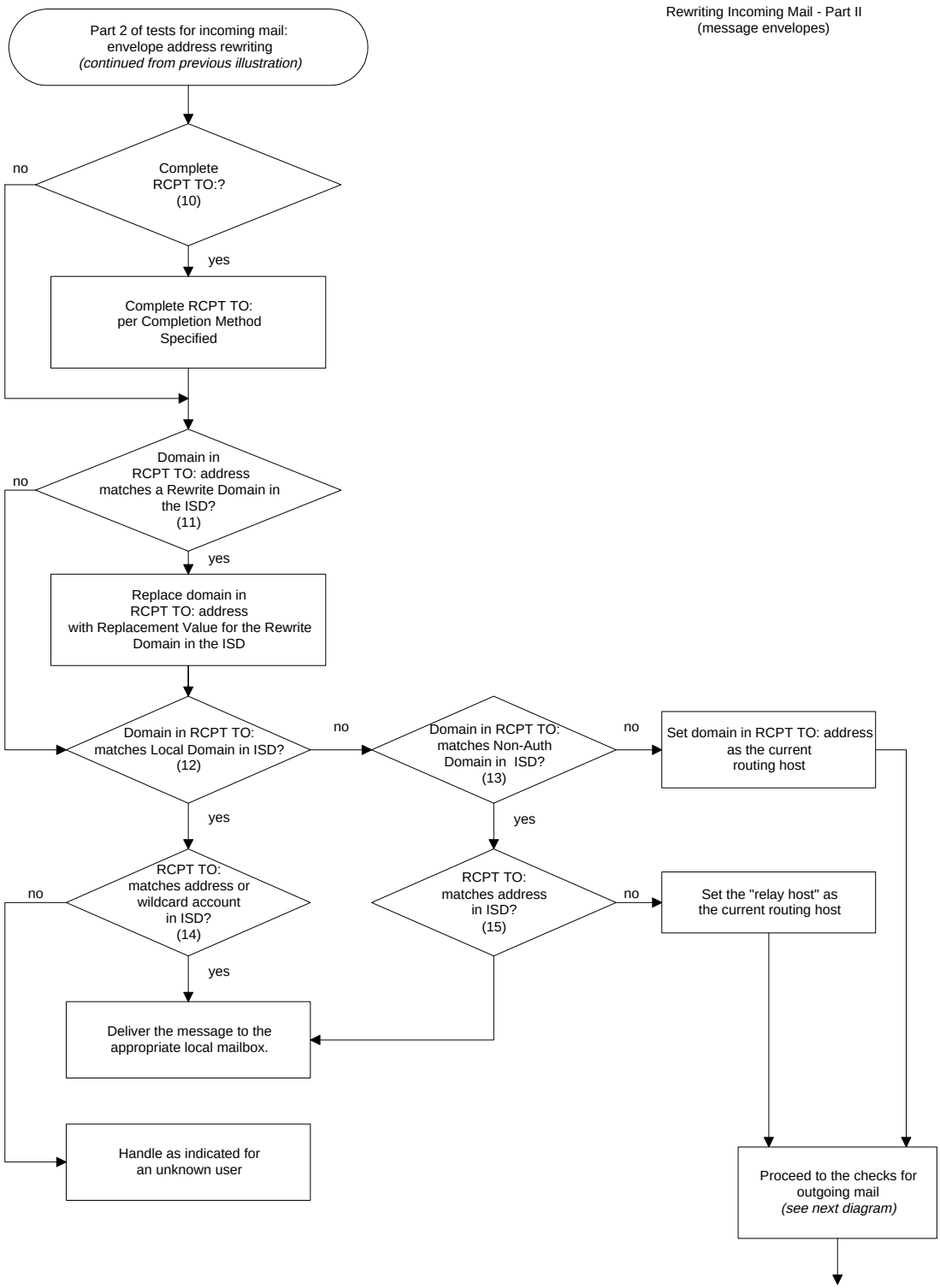

Figure 12. Header Rewriting for Incoming Mail

10. Check the `RCPT TO:` envelope address to see if it is complete.
    - If `RCPT TO:` is complete, go to Step 11.
    - If `RCPT TO:` is *not* complete, complete it (typically using the value set in the `defaultDomain` configuration key). Go to Step 11.
11. Check the complete `RCPT TO:` address on the envelope against the rewrite domains identified in the Integrated Serves Directory (ISD). This controls envelope address rewriting for incoming mail.
    - If the domain in the `RCPT TO:` address matches a rewrite domain established in the Integrated Services Directory, the domain in the `RCPT TO:` address is overwritten with the replacement value indicated for the rewrite domain. Go to Step 12.
    - If the domain in the `RCPT TO:` address on the envelope does not match any of the rewrite domains established in the Integrated Services Directory, no envelope address rewriting is performed. Go to Step 12.

Note: *At this point the* `RCPT TO:` *addresses for all envelopes destined for local delivery will include a domain which is either local or non-authoritative.*

---

12. Check the `RCPT TO:` address on the envelope against the local domains identified in the Integrated Serves Directory (ISD).
    - If the domain in the `RCPT TO:` address on the envelope matches a local domain established in the Integrated Services Directory, the address is looked up in the ISD. Go to Step 13.
    - If the domain in the `RCPT TO:` address on the envelope does not match any of the local domains established in the Integrated Services Directory, go to Step 14.
13. Check the `RCPT TO:` address on the envelope against the account addresses in the Integrated Serves Directory (ISD).
    - If the `RCPT TO:` address on the envelope is an exact match for an address in the Integrated Services Directory, deliver the message to the appropriate local mailbox (the mailbox associated with the account in which the matching address was found).
    - If the `RCPT TO:` address on the envelope does not match any of the addresses in the Integrated Services Directory, but there is a wildcard account specified for the domain used in the `RCPT TO:` address, the message is delivered to the wildcard account.
    - If the `RCPT TO:` address on the envelope does not match any of the addresses in the Integrated Services Directory, and there is no wildcard account specified for the domain used in the `RCPT TO:` address, the recipient is considered an unknown user and the message is handled according to the instructions set in the `Error-Action/acctInvalidUser` configuration key.

14. Check the `RCPT TO:` address on the envelope against the non-authorized domains identified in the Integrated Serves Directory (ISD).
    - If the domain in the `RCPT TO:` address on the envelope matches a non-authorized domain established in the Integrated Services Directory, go to Step 15
    - If the domain in the `RCPT TO:` address on the envelope does not match any of the non-authorized domains established in the Integrated Services Directory, set the routing host using the value of the domain in the `RCPT TO:` address. Go to Step 16.

Note: *The routing host is established at the point a message is determined to be outgoing mail (i.e., mail destined for delivery to an external mail host). The routing host identifies the external host to which this message should be passed. The value of the routing host is independent of the domain contained in the RCPT TO: address on the message envelope.*

---

15. Check the `RCPT TO:` address on the envelope against all addresses stored in the ISD (primary or alias).
    - If the `RCPT TO:` address on the envelope matches an address in the ISD, deliver the message to the local mailbox associated with that account.
    - If the `RCPT TO`: address on the envelope does not match any of the addresses in the ISD, set the routing host using the value of the Relay Host associated with the non-authorized domain. Go to Step 16.

At this point all local delivery is complete. Any remaining messages are considered outgoing mail. Outgoing mail is subject to a separate series of routing and rewriting checks.

This scenario is continued in Section 5.5.6, beginning with Step 16.

5.5 Rerouting and Rewriting Outgoing Mail

Outgoing mail is defined as mail whose final destination is a *remote mail server*. InterMail provides features for rewriting outgoing mail headers, rewriting domains, and also for rerouting outgoing mail.

Although supplemented by other configuration keys, the principal control for outgoing mail rewriting and rerouting is the `mailRoutingTable` configuration key.

5.5.1 The Mail Routing Table

Normally outgoing mail is routed using the MX records in DNS, or by delivering a message to the actual host name specified in the envelope address. The SMTP Mail Routing Table offers another routing option that you can use to override normal delivery.

A single entry in the Mail Routing Table consists of one required element and two optional elements, with each having a different routing or rewriting function for outgoing mail.

The three elements of the Mail Routing Table entry are:

- The routing host element (`<rtg.host>:<new.rtg.host>`), which is used to route mail from one host to another, without changing envelope addresses or message headers. *This element is required for each Mail Routing Table entry.*
- The optional header rewriting element (`[header-rewrite]`), which may be used to specify header rewriting for outgoing mail.
- The optional domain rewriting element (`[rewrite-domain=new.domain]`), which may be used to specify domain rewriting 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 current destination (the routing host to which the outgoing message is initially directed).
`<new.rtg.host>`	Specifies the desired destination to which the outgoing message will be rerouted.
`header-rewrite`	An optional entry indicating that header rewriting is desired for outgoing mail. This impacts only the message headers.
`rewrite-domain`	An optional entry indicating that domain rewriting is desired.
`=<new.domain>`	Specifies the new domain name which will replace the original in the `RCPT TO:` address.

Note: *Domain rewriting for* outgoing *mail has nothing to do with rewrite domains in the ISD. It is specified only in the Mail Routing Table.*

The Mail Routing Table is highly flexible, so the discussion of this feature is somewhat complex.

5.5.2 Rerouting Mail

Rerouting of outgoing messages is controlled by the `<rtg.host>:<new.rtg.host>` element in a Mail Routing Table entry. The purpose of rerouting is simply to redirect mail from one host to another, *without rewriting anything*.

A single routing host entry in the Mail Routing Table might look like this:

```
/*/mta/mailRoutingTable: [venus.software.com:gateway.com]
```

This setting specifies that any message that would normally be delivered to a host called `venus.accordance.com` will, in fact, be delivered to a machine called `gateway.com`.

How Rerouting Entries are Matched

The Mail Routing Table can have multiple entries, with each one specified on its own line, and within its own set of square brackets. For example

```
/*/mta/mailRoutingTable: [venus.software.com:gateway.com]
                        [*.software.com:internal_server.com]
                        [default:firewall.com]
```

Each of the above entries has a different effect:

- The first entry affects only those messages destined for the host whose full name is `venus.software.com`. These messages are rerouted to another host called `gateway.com`.
- The second entry uses a wildcard (*) for pattern matching. It specifies that all mail initially destined for `<any_host>.software.com` will be rerouted to `internal_server.com`.

  This pattern matching means that mail sent to (for example) `mercury.software.com` or `neptune.software.com` will be rerouted to `internal_server.com`.

Note: *Mail sent simply to* `software.com` *would* not *be rerouted in this scenario. This is because* `*.software.com` *requires that* some *host name appear before* `software.com`. *In other words, a three part name is required for a match.*

- The third entry uses `default` to indicate that all messages for *any host* will be rerouted to the host called `firewall.com`. No pattern matching of any kind is done.

Of these three entries, the first is the most specific, since it stipulates that *only* messages destined for `venus.software.com` will be rerouted to `gateway.com` (i.e., a literal match is required). The third entry is the most general, since it stipulates that messages destined for *any host* will be rerouted to `firewall.com`.

Taken as a whole, the example table gives InterMail the following rerouting instructions:

1. First, reroute all mail destined *specifically* for `venus.software.com` to `gateway.com`.
2. Next, reroute messages destined for any other named `software.com` host (i.e., `*.software.com`) to the host called `internal_server.com`.
3. Finally, reroute mail destined for all other hosts (regardless of host name) to the machine called `firewall.com`.

How Rerouting Entries are Read

Multiple entries in the Mail Routing Table are read from top to bottom. *They are not read from most specific to least specific!* In other words, the first possible match found is the one used.

This is an important fact to keep in mind, because table entries that are not specified in the correct order may yield undesired results.

To help illustrate this point, let's begin with the same Mail Routing Table previously described:

```
/*/mta/mailRoutingTable: [venus.software.com:gateway.com]
                        [*.software.com:internal_server.com]
                        [default:firewall.com]
```

When InterMail reads this table to see if an outgoing message requires rerouting, the entry it will to match is `venus.software.com` (just because it is the first one in the list).

Now, assume the `RCPT TO:` address in an outgoing message is `joe@software.com` (the routing host portion being `software.com`.)

Since the `venus.software.com` entry requires an *exact* match to do rerouting, InterMail ignores it and moves down to the next entry in the table.

The second table entry (`*.software.com:internal_server.com`) does not create a match either (i.e., there is nothing in `joe@software.com` to substitute for the wildcard (`*`) in the table entry). So InterMail 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*. This being the case, InterMail reroutes the message for `joe@software.com` to the `firewall.com` host machine (with no changes to the envelope address or message headers).

How incorrect table entries can misroute mail

The following examples helps show how an incorrectly ordered Mail Routing Table can produce unintended results. Assume that the order of the entries in our previous example is reversed:

```
/*/mta/mailRoutingTable: [default:firewall.com]
                        [*.software.com:internal_server.com]
                        [venus.software.com:gateway.com]
```

Here the `default` entry would always be read first, and every outgoing message would be rerouted to `firewall.com`. Since every host name would be an acceptable match for `default`, no further entries in the Mail Routing Table would ever be read.

The following example presents a more complex situation that would also produce incorrect rerouting:

```
/*/mta/mailRoutingTable: [*.software.com:internal_server.com]
                        [venus.software.com:gateway.com]
                        [default:firewall.com]
```

The problem here is that messages destined for `venus.software.com` will never be rerouted to `gateway.com`, as the second entry in the table stipulates. This is because `*.software.com` is the first entry in the Mail Routing Table. That means it will be read first—and, since `venus.software.com` is a match for `*.software.com`, messages destined for `venus.software.com` will be rerouted to `internal_server.com`, rather than to `gateway.com`.

All messages for other `*.software.com` matches (e.g., `neptune.software.com`, and `saturn.software.com`,) would be correctly rerouted to `internal_server.com`.

But if `venus.software.com` is to be a successful exception to the pattern matching rule defined by `*.software.com`, then `venus.software.com` must be listed *before* `*.software.com` in the Mail Routing Table. This way, outgoing messages destined for `venus.software.com` will first be matched against the `venus.software.com` entry in the Mail Routing Table, and be correctly rerouted to `gateway.com`.

Rule-of-thumb for Mail Routing Table Entries

You should always enter the most *specific* rerouting instruction first, and work your way down to the most general.

Note: *If the* `mailRoutingHost` *configuration key is missing, or if its value is null, it means that DNS will be used deliver all mail destined for external hosts.*

5.5.3 Header Rewriting for Outgoing Mail

Rerouting mail from one host to another (as described in Section 5.5.2) in itself does nothing to rewrite message headers or the domain portion of envelope addresses. It simply resets the routing host (the destination to which the message will be passed).

Taken by itself, the Mail Routing Table entry

```
/*/mta/mailRoutingTable: [venus.software.com:gateway.com]
```

reroutes all messages destined for `venus.software.com` to `gateway.com`. But, the headers and envelope address remain unchanged. For example, a message addressed in the following manner:

```
joe@venus.software.com
```

would be delivered to the `gateway.com` host, but Joe will still see the original headers when he reads the message.

There are two additional steps you must take in order to rewrite message headers for outing mail:

- Add the `header-rewrite` option to the Mail Routing Table entry
- Define a list of headers that are eligible for rewriting. This step is similar to the one described for incoming mail header rewriting (see Section 5.4.1). However, in this case you will define the eligible headers in the `rewriteGatewayHeaderList` configuration key.

Adding the header rewriting element to a Mail Routing Table entry

To add header rewriting for the table entry that reroutes messages from `venus.software.com` to `gateway.com`, you would enter the following line in the Mail Routing Table:

```
/*/mta/mailRoutingTable: [venus.software.com:gateway.com header-
rewrite]
```

Adding the `header-rewrite` element after the rerouting element specifies that header rewriting is desired for all outgoing mail that is rerouted from `venus.software.com` to `gateway.com`, however header rewriting will not occur unless you have defined some headers that will be eligible for rewriting.

Defining the headers that are eligible for rewriting on outgoing mail

To define the headers that will be eligible for rewriting, use the `rewriteHeaderGatewayList` configuration key. You can request rewriting for one or more of the following headers:

```
To:
Cc:
Bcc:
Reply-To:
From:
Sender:
```

Each header must appear on its own line, within its own set of square brackets. To rewrite outgoing `To:`, `Cc:` and `From:` headers, do the following:

- Use `imconfedit` to set the `rewriteGatewayHeaderList` configuration key as follows:

```
/*/mta/rewriteGatewayHeaderList: [To:]
                                 [Cc:]
                                 [From:]
```

This setting does not necessarily mean that InterMail will *automatically* rewrite the outgoing `To:`, `Cc:` and `From:` headers for every incoming message. It simply means that these headers will be *eligible* for rewriting if the `header-rewrite` element is added to the Mail Routing Table entry for which header rewriting is desired.

When both of these conditions are met, InterMail checks the addresses of qualifying messages (those the Mail Routing Table says require host rerouting). If an address in the eligible headers is found in an account that appears in the ISD (*and* there is a forwarding address for that account), the eligible header is rewritten with the forwarding address. If all of these conditions are not met, no rewriting is done for that header.

Note: *Rewriting of outgoing headers can only be done if the eligible header matches an address in the ISD* ***and*** *a forwarding address has been established for that account.*

Rewriting headers without rerouting outgoing mail to another host

Header rewriting for outgoing mail requires an entry in the Mail Routing Table. But suppose you want to rewrite headers without rerouting mail from one host to another? To do this, create a Mail Routing Table entry that routes mail from a named host to itself, then perform the rest of the header rewriting steps outlined in this section.

The syntax for this would be:

```
/*/mta/mailRoutingTable: [<HostA>:<HostA> header-rewrite]
```

For example, the following entry requests header rewriting for all eligible headers in outgoing mail destined for the `gateway.com` host. But, since `gateway.com` routes back to itself, no actual rerouting is ever done. All that happens is that eligible headers are rewritten when all the rewriting requirements are met:

```
/*/mta/mailRoutingTable: [gateway.com:gateway.com header-rewrite]
```

You can set up a Mail Routing Table that requests header rewriting for some entries, but not for others. For example, following table:

```
/*/mta/mailRoutingTable:
                   [venus.software.com:gateway.com header-rewrite]
                   [*.software.com:internal_server.com header-rewrite]
                   [default:firewall.com]
```

specifies that eligible headers will be rewritten for all outgoing messages that are rerouted from `venus.software.com` to `gateway.com`, and also for those messages that are rerouted from `*.software.com` to `internal_server.com`. However, all other rerouted mail (those messages that are rerouted to `firewall.com`) will not have their headers rewritten.

5.5.4 Domain Rewriting for outgoing mail

The final option in the Mail Routing Table is to use domain rewriting. This is roughly equivalent to the idea of using ISD rewrite domains for incoming mail (see Section 5.1.2). However, in the case of *outgoing* messages, InterMail *does not check the ISD* for rewrite domain values. Instead, you specify the rewrite domain values right in the Mail Routing Table.

To specify domain rewriting for outgoing mail, you would use the last optional element of a Mail Routing Table entry:

```
[rewrite-domain=<new.domain>]
```

where

`rewrite-domain`	Indicates that domain rewriting is desired.
`=<new.domain>`	Specifies the new domain name (to which the original will be rewritten).

The Mail Routing Table syntax to add domain rewriting for outgoing mail is:

```
mailRoutingTable: [<hostA>:<hostB> rewrite-domain=<new.domain>]
```

The following example specifies that outgoing messages destined for the host called `venus.software.com` should be rerouted to the host called `gateway.com`. *In addition*, it tells InterMail to rewrite `RCPT TO`: addresses in messages to include a different domain name:

```
/*/mta/mailRoutingTable:
[venus.software.com:gateway.com rewrite-domain=software.com]
```

Remember that, while domain rewriting for outgoing mail changes the `RCPT TO:` address in the envelope, it does not override the rerouting instructions specified in the `<rtg.host>:<new.rtg.host>` portion of a Mail Routing Table entry.

Domain rewriting also has no effect on either an envelope's `MAIL FROM:` address or on the message headers.

Domain rewriting without rerouting outgoing mail to another host

Just as with header rewriting, domain rewriting for outgoing mail requires an entry in the Mail Routing Table. But as with the header rewriting option, you can specify domain rewriting for outgoing mail *without* having to reroute this mail to another host.

To do this, create a Mail Routing Table entry that routes mail from a named host to itself, then specify domain rewriting as you normally would.

The syntax for this would be:

```
[<HostA>:<HostA> rewrite-domain=<new.domain>]
```

For example, the following entry specifies that outgoing mail for accordance.com have the domain portion of the `RCPT TO:` address rewritten to `software.com`. However, the routing host will not change:

```
/*/mta/mailRoutingTable:
[accordance.com:accordance.com rewrite-domain=software.com]
```

You can set up a Mail Routing Table that requests domain rewriting for some entries, but not for others. For example, the table defined as:

```
/*/mta/mailRoutingTable:[venus.accordance.com:gateway.com]
[*.accordance.com:firewall.com rewrite-domain=software.com]
```

specifies that mail for the host `venus.accordance.com` be rerouted to the host called `gateway.com`, without any domain rewriting. It then specifies that outgoing messages destined for `*.accordance.com` should be rerouted to the `firewall.com` machine, *and* have the domain portion of their `RCPT TO:` addresses rewritten to `software.com`.

5.5.5 Using All the Mail Routing Table Elements Together

The flexibility of the Mail Routing Table allows you to combine all its elements into a single entry, so that you can reroute outgoing messages from one routing host to another, while also rewriting headers and domains.

In this example, the goal is to reroute all outbound messages for the `accordance.com` host to the `software.com` host, while simultaneously rewriting the `To:`, `Cc:` and `From:` message headers and also the domain portion of the envelopes' `RCPT TO:` addresses. The procedure for doing this is as follows:

1. Using `imconfedit`, create an entry in the Mail Routing Table that reads:

   ```
   /*/mta/mailRoutingTable
   [accordance.com:software.com header-rewrite
   rewrite-domain=software.com]
   ```

2. Using `imconfedit`, set the `rewriteGatewayHeaderList` configuration key as follows to make the `To:`, `CC:` and `From:` outbound message headers eligible for rewriting:

   ```
   /*/mta/rewriteGatewayHeaderList: [To:]
                                    [From:]
                                    [Cc:]
   ```

5.5.6 Outgoing Mail Rerouting and Rewriting (Global View)

The following diagram continues the scenario illustrated in Sections 5.4.2 and 5.4.4, beginning with Step 16. The numbered steps indicate the sequence in which InterMail performs the various checks required for rerouting and rewriting headers for outgoing mail.

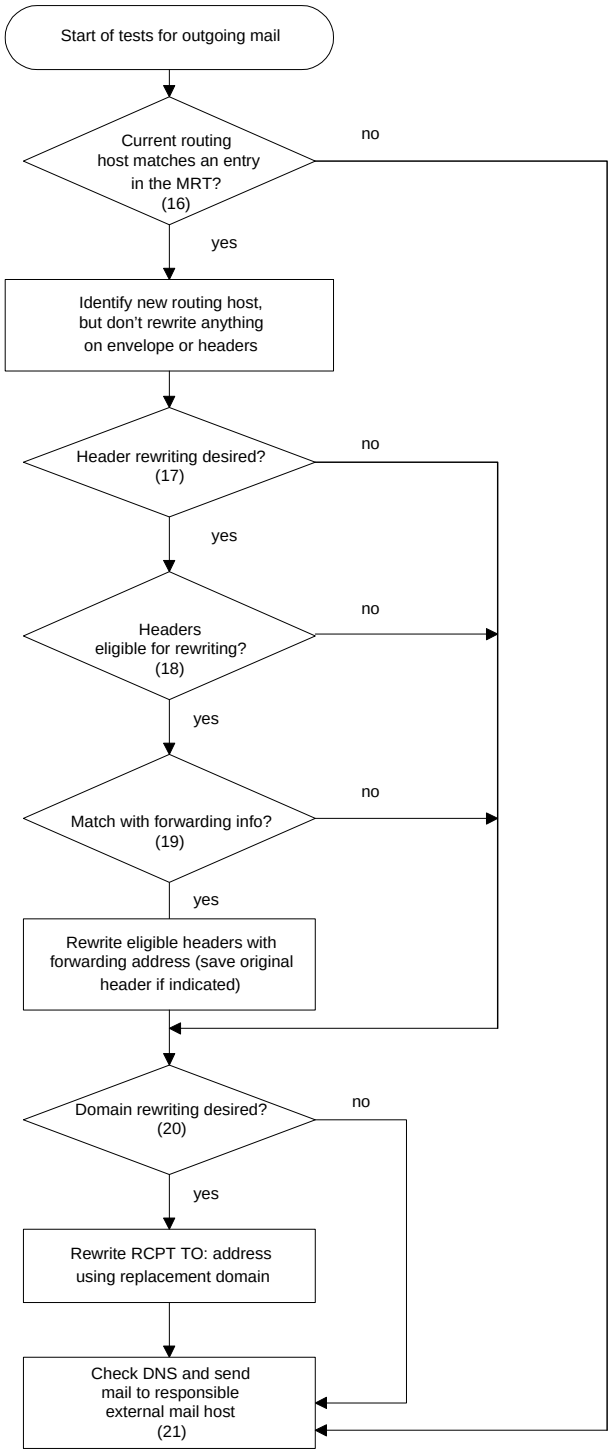

Figure 13. Rerouting and Rewriting for Outgoing Mail

16. Check the routing host against the entries in the `mailRoutingTable` configuration key. Find the first match between the routing host and the leftmost portion of a Mail Routing Table entry.

For the sake of this discussion, Mail Routing Table entries are identified as follows:

```
<rtg.host>:<new.rtg.host> [header-rewrite] [rewrite-domain=new.domain]
```

- If the current routing host does not match any of the routing hosts in the leftmost portion a Mail Routing Table entry (the `rtg.host` portion), no rewriting of the outgoing message occurs. Go to Step 21.
- If the current routing host matches one of the routing hosts listed in the leftmost portion a MRT entry (the `<rtg.host>` portion), the destination of the message is changed. The new destination address is the value in the `<new.rtg.host>` portion of the MRT entry. This modification of the intended destination changes the value of the routing host only; it does NOT result in any changes to envelope or header addressing! Go to Step 17.

Note: *The entries in the Mail Routing Table are order dependent. The table is read from top to bottom, and the first match found is the one that takes precedence.*

---

17. Check the selected entry in the `mailRoutingTable` configuration key to see if header rewriting is desired (i.e., the `header-rewrite` option is being used).
    - If the selected Mail Routing Table entry (as identified in Step 16) includes the `header-rewrite` option, go on to check for headers to be rewritten in Step 18.
    - If the selected Mail Routing Table entry (as identified in Step 16) does not include the `header-rewrite` option, skip header rewriting for outgoing mail and go on to Step 20 (to check domain rewriting).
18. Check message headers against the entries in the `rewriteGatewayHeaderList` configuration key. The entries in this key 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, then skip header rewriting for outgoing mail and go on to Step 20 (to check domain rewriting).
    - If one or more of the headers in the message are listed in the `rewriteGatewayHeaderList` key, then those headers listed are subject to a further test for header rewriting (see Step 19).

19. Check the addresses in message headers eligible for rewriting against the address entries in the Integrated Services Directory.
    - If no match is found for a particular header, then (despite its initial eligibility) that header is not rewritten. Go to Step 20.
    - If the address in an eligible header matches an address in the ISD, the associated account record is examined for forwarding information.
        - If the associated account includes a forwarding address, the address in the eligible header is replaced with the forwarding address found in the Integrated Services Directory account record. The original header info is recorded in a new `X-Header`, if the `rewriteSaveOrig` configuration key is set to `true`. Go to step 20.
        - If the associated account does not include a forwarding address, the address in the eligible header is not rewritten. Go to Step 20.

Note: *The various headers are considered individually. As a result, some may be rewritten while others are not.*

20. Check the selected entry in the `mailRoutingTable` configuration key to see if domain rewriting is desired.
    - If the selected Mail Routing Table entry (the one identified in Step 16) does not include the `rewrite-domain` option, skip domain rewriting for outgoing mail and go on to Step 21.
    - If the selected Mail Routing Table entry includes the `rewrite-domain` option, replace the domain in the `RCPT TO:` address on the message envelope with the `<new.domain>` indicated in the rightmost portion of the Mail Routing Table entry. Go to Step 21.

Important: Rewriting the envelope addresses at this point has no effect on the routing of the outgoing message. The mail will continue to be routed to the destination address identified in Step 16.

21. Request the DNS records associated with the appropriate external destination and hand the message off to the external mail host identified.

5.6 Proxying Mail Between Hosts

Proxying is special form of mail routing that allows an InterMail host to act as a kind of "surrogate" for a legacy mail host by receiving messages and relaying them to the legacy host. It is used primarily during migration (transferring user accounts and messages from a Post.Office or sendmail system to InterMail). Proxying makes it possible for end users on the legacy system to continue accessing their mailboxes during part of the migration process.

For a complete discussion of using proxy mode, see the *InterMail Migration Guide*.

5.7 Delivery Status Notification (DSN)

InterMail supports Delivery Status Notification (DSN). A DSN is a special notice that is delivered to e-mail senders on other systems that also support DSN. These notifications can advise senders that:

- Mail has been delivered successfully, or that
- Mail delivery has failed or been delayed.

DSNs are not sent on a per-user basis. Rather, these messages are automatically sent on behalf of all InterMail accounts, and are received by any senders whose mail clients and service providers are DSN-enabled.

DSNs help companies that maintain large mailing lists keep their lists up-to-date. This is because DSN messages are far more detailed than ordinary bounce notices. 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 if the problem was due to an expired e-mail addresses that is no longer valid.

Enabling/Disabling Delivery Status Notification

To enable or disable Delivery Status notification for a particular condition, you would use `imconfedit` to set the value of one or more `Error-Action/mtaMessage` keys. The possible values for these keys are `log, hold`, and `return.` If you specify a `return` value, a DSN will be sent to the requesting client when this condition occurs. If you specify a value or values other than `return`, DSN is disabled for this condition.

Examples

To enable Delivery Status Notification where a message was not delivered because of bad delivery information, you would use `imconfedit` to create the following setting:

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

Specifying the`[log]` value means that this error will also be logged.

To disable Delivery Status Notification for the same condition, you would use `imconfedit` to remove the return value from the configuration key, so that the setting looked something like this:

```
/*/mta/Error-Actions/mtaMessageDelivered: [log]
```

With the above setting, a DSN would never be sent for this condition, regardless of whether or not a client requests it.

You can set Delivery Status Notifications for the following conditions by adding a `[return]` value to the indicated configuration key:

- To send a DSN for messages having bad delivery information, use `imconfedit` to set a value of `return` in the `Error-Actions/mtaMessageDelivered` configuration key.
- To send a DSN when a deferred message has been queued longer than the configured limit, use `imconfedit` to set a value of `return` in the `Error-Actions/mtaMessageQueuedTooLong` configuration key. (Queuing time limits are set using the `maxQueueTimeInDays` configuration key)
- To send a DSN saying that the message reached its destination and was forwarded to at least two mailboxes or recipients, use `imconfedit` to set a value of `return` in the `Error-Actions/mtaMessageExpanded` configuration key.
- To send a DSN when a messages was rejected by the receiving SMTP server, use `imconfedit` to set a value of `return` in the `Error-Actions/mtaMessageRejected` configuration key.
- To send a DSN stating that a messages has been relayed to a machine that does not handle Delivery Status Notification, use `imconfedit` to set a value of `return` in the `Error-Actions/mtaMessageRelayed`
- To send a DSN because a message is larger than the SMTP server is willing to accept, use `imconfedit` to set a value of `return` in the `Error-Actions/mtaMessageTooLarge` configuration key.

6

Mailbox Management

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 processing is in progress, any of these messages may be stored temporarily. However, once the delivery operation is complete, all messages addressed to local users must be stored persistently until they are retrieved and/or deleted.

This chapter discusses the operations surrounding persistent storage of mail. The topics covered include:

- Physical vs. logical message storage
- Creating, moving, and deleting mailboxes
- Mailbox quotas
- Message aging
- Garbage collection of deleted messages

Note: *For information about temporary storage of mail that is in process, please refer to Chapter 7.*

6.1 Persistent Message Storage

A typical InterMail installation includes hundreds of thousands of users with millions of messages. In installations of this size, efficient storage of message data and fast access to message information are critical. To accommodate both goals, InterMail distinguishes between physical and logical mail storage and provides separate storage mechanisms for each.

The Message Store Server (MSS) is the InterMail component responsible for persistent storage of mail. Each MSS host contains:

- a *Message File System*, which accommodate physical message storage
- a *Message Store Database*, which manages logical message storage

For each message delivered to a local user, information must be stored 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, etc.).

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 14 illustrates storage of a 5 Mb message addressed to local users John Doe and Susie Queue, both of whose mailboxes reside on the same MSS host.

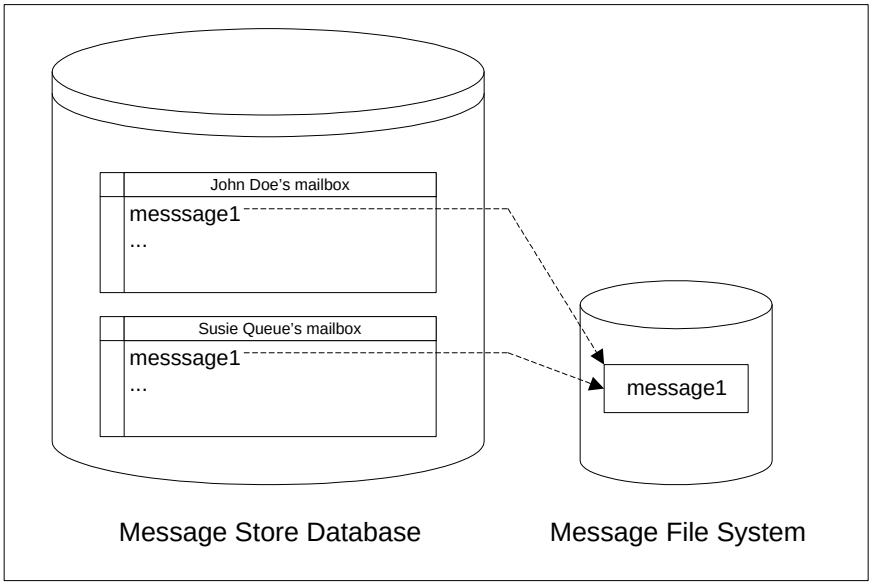

Figure 14. When a single message is stored for multiple users on an MSS host.

The 5 Mb of message data, which is common to both recipients, is stored only once—as a single message file in the Message File System. However, multiple entries are made 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.

Note: *There is a separate Message File System and Message Store Database for each MSS host. If your InterMail installation includes three MSS hosts, there will be one Message File System and one Message Store Database on each of those three hosts.*

The sections that follow provide additional information on the Message File System and Message Store Database, including an explanation of the specific storage mechanisms employed for each.

6.1.1 Physical Message Storage

The body and header of each message are stored as a single binary file in the Message File System. The body of the message includes the text and any attachments. 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 it is stored by InterMail, it never changes.

One file per message is stored in the Message File System, regardless of the number of intended recipients for that message. Message files are spread throughout a large directory tree created for this purpose. In very large installations, there may be thousands of directories in this tree, together containing millions of stored messages.

The top of the Message File System hierarchy on each MSS host is specified by the configuration key `messageFilesDir` (by default, `$INTERMAIL/msgfiles`). This directory contains three items:

- `messages`, a directory which contains the thousands of "bucket" directories that store individual message files.
- `buckets.dir`, a file which lists the name of the messages directory (by default, `messages`).
- `buckets`, a file which 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 accommodate a balancing of message volume between bucket directories, this directory list excludes the bucket directories that contain the greatest message volume.[3]

Note: *The* `buckets` *file is regenerated on a regular basis (via* `cron`*) to ensure that the balancing of message volume between bucket directories continues over time.*

The process of creating this Message File System directory tree takes place automatically during installation. However, if you later need to create another directory structure to accommodate greater message activity, use the `imbucketscreate` administrative command, which is described in Chapter 12 of the *InterMail 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`.

6.1.2 Logical Message Storage

While the contents of a message are stored in the Message File System, information *about* a message is stored in the Message Store Database. 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. Meanwhile, the actual body of the message 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.

Within the Message Store Database, the following data objects define a logical relationship between end user accounts and individual messages:

- mailboxes
- folders
- messages

Note: *It is important to remember that 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. This distinction is import to keep in mind, especially when moving mailboxes from one MSS host to another.*

---

[3] A total of one-third of all bucket directories (i.e., the most full 33% of directories) are excluded from the `buckets` listing.

Mailboxes

A *mailbox* is a storage area for messages that are sent to an end user's account. Typically, each account in the Integrated Services Directory is associated with a mailbox.[4] Each mailbox "contains" a series of folders, which in turn contain the messages that have been received for the end user. Mailboxes are at the top of the MSS database object hierarchy, and as such, they are the objects on which most administrative operations in the MSS database are performed.

Folders

A *folder* is a container for messages. Each folder exists within a specific mailbox (and therefore, for a specific end user), and can also exist within another folder. By default, all InterMail mailboxes contain the following folders:

- `INBOX`
- `SentMail`
- `Trash`

The first of these, `INBOX`, is the folder in which all new messages are placed as they are received by the MSS. Although folders are not visible to end users who retrieve mail via the POP Server, their messages are always retrieved from the `INBOX` folder.

The other default folders—`SentMail` and `Trash`—have special meaning to end users who access their mailboxes via the IMAP Server. IMAP clients allow end users to create new folders in the MSS, copy and move messages between these folders, and delete existing folders. Many clients allow users to save a copy of all outgoing messages to a specific folder, while moving "deleted" messages to another folder. The `SentMail` and `Trash` folders are meant to be used for these purposes by IMAP clients.

Warning! In IMAP terminology, folders are sometimes referred to as "mailboxes". Be careful not to confuse these terms in InterMail. Again, *mailboxes* are the top-level object in the Message Store Database (one per account), while *folders* are contained in mailboxes (many per account).

An additional folder may be created automatically by InterMail: the `.ERRORS` folder. This folder is created when InterMail encounters a message which—for one reason or another—could not be retrieved by the POP or IMAP Server. Events that could cause this rare error condition are a loss of the corresponding message file or a corruption of database information. When this occurs, the message is moved from whichever folder it was stored and into the mailbox's `.ERRORS` folder (which is created if it does not already exist). This means that the troubled message is moved aside so that normal processing can occur for subsequent messages.

Note: *Reprocessing messages that have been moved to the* `.ERRORS` *folder is accomplished with the* `immsgprocess` *administrative command. See Chapter 12 in the InterMail Reference Guide for information on using this command.*

---

[4] 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.

Messages

Message objects in the Message Store Database correspond to an individual message that is "in" one or more mailboxes. Among the information stored for each message object in the Message Store Database are:

- The path to the corresponding message file in the Message File System.
- Pointers to the mailbox(es) and folder(s) 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. Note that this header information is the only data that is stored in *both* the Message File System and Message Store Database.
- Message status flags, which indicate whether the message has been read or deleted by the end user. If the message is stored in more than one folder (for example, a single message received by two different end users), 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.*

6.2 Creating Mailboxes

Although mailboxes are associated with accounts, an account's mailbox is not created when the account itself is created in the Integrated Services Directory. The creation of mailboxes is therefore an operation that is distinct from account creation. Mailbox creation can be accomplished in several different ways, which fall into two general categories:

- automatic mailbox creation
- manual mailbox creation

Because it requires minimal administrative effort, automatic creation is by far the most commonly-used method. However, you should also be aware of the operations required for manual mailbox creation if it becomes necessary.

Note: *Each method of mailbox creation also creates the default folders (*`INBOX`*,* `SentMail`*, and* `Trash`*) within the created mailboxes.*

6.2.1 Automatic Creation

InterMail allows a mailbox to be automatically created for an existing account the first time that the mailbox is needed. There are two events that can trigger automatic creation:

- A message is received for the account and needs to be stored in the account's mailbox.
- The end user attempts to access his or her mailbox via the POP or IMAP Servers.

Until one of these events occurs, the existence of the account's mailbox is irrelevant, so the creation of the mailbox need not occur when the account is created. By waiting to create mailboxes until they are needed for message storage or retrieval, you can avoid subjecting the MSS to unnecessary processing, and also avoid creating mailboxes for accounts that do not use the local delivery method.[5]

A single configuration key controls the automatic creation of mailboxes: `createsMboxes`. If set to `true`, a mailbox will be created for an account the first time that it is needed. If this key is set to `false`, mailboxes must be created manually (as described in the following section). This key can be defined separately for the MTA, POP Server, and IMAP Server, but is typically defined once on a global level (i.e., by the configuration database entry `/*/common/createsMboxes`).

6.2.2 Manual Creation

Mailboxes can be manually created in the Message Store Database in two ways:

- by executing the `imboxcreate` administrative command
- by executing the `imboxsync` administrative command
- by creating a program that calls the appropriate InterMail API functions

This manual covers only the use of `imboxcreate` and `imboxsync`; for information on using the InterMail API libraries, refer to the *Integrated Services Directory Reference Guide*.

Executing imboxcreate

The `imboxcreate` command is used for the creation of a single mailbox in the Message Store Database. The usage of this command is as follows:

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

Where:

`-help`	Provides a usage statement.
`<host>`	Specifies the name of the MSS host on which the mailbox will be created.
`<internalID>`	Specifies the internal ID number of the account associated with the mailbox.

[5] 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), then the account does not require a mailbox. Refer to the Integrated Services Directory Reference Guide for more information on the available account delivery methods.

Note: *To obtain the internal ID number of an account, use the* `imdbcontrol` *or* `imaccountquery` *administrative commands.* `imdbcontrol` *is documented in the Chapter 3 of the Integrated Services Directory Reference Guide, while* `imaccountquery` *is described in Chapter 12 of the InterMail Reference Guide.*

For example, to create a mailbox on the MSS host `venus` for the account whose internal ID number is `123456`, you would execute:

```
imboxcreate venus 123456 1000000
```

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

```
Message store 123456 created!
```

Executing imboxsync

The `imboxsync` command is used to synchronize mailboxes on an MSS host with accounts in the Integrated Services Directory. This command is executed on a specific MSS host, and performs the following operations:

- Verifies that a mailbox exists for each account in the Integrated Services Directory that uses the current system as its MSS host. If a mailbox does not exist for an account, `imboxsync` creates it.
- Verifies that all mailboxes on the current MSS host correspond to existing accounts in the Integrated Services Directory. If the Message Store Database includes a mailbox for an account that does not exist in the Integrated Services Directory, `imboxsync` deletes the mailbox.

This command is typically used only during migration or system maintenance, and is *not* intended for regular use. Because of the extensive database operations performed by `imboxsync`, this command requires a significant amount of time to complete. Also, all MSS processes on the target MSS host must be shut down prior to executing `imboxsync`.

The usage of this command is as follows:

```
imboxsync [-help] [-n|-y|-c|-d]
```

Where:

`-help`	Provides a usage statement.
`-n`	Modifies the Message Store Database without prompting.
`-y`	Prompts for confirmation before creating or deleting mailboxes. (This is the default behavior if `imboxsync` is run with no parameters.)
`-c`	Prompts only before creating new mailboxes.
`-d`	Prompts only before deleting mailboxes.

For example, to synchronize mailboxes on the MSS host `venus` with accounts in the Integrated Services Directory, execute the following steps:

1. Log in to the target MSS host as the InterMail user.
2. Shut down all MSS processes on the MSS host:

   ```
   imctrl venus stop mss
   ```

3. Execute the imboxsync command:

   ```
   imboxsync
   ```

   When this command executes, it will display the results of its operations and prompt for the creation and deletion of mailboxes:

   ```
   imboxsync: Fetching master account information from Oracle...
   imboxsync: Fetching InterMail MSS account information...
   imboxsync: Comparing account information...
   imboxsync: need to CREATE 4 MSS mailboxes:
   imboxsync: Ready to run immsscall to create the mailboxes [y/n]?
   ```

4. To commit the creation of new mailboxes, type `y` at this prompt. `imboxsync` then confirms the creation of the new mailboxes, and exits:

   ```
   imboxsync: creating mailboxes...
   ```

6.3 Mailbox Quotas

Each InterMail mailbox is subject to a variety of limits, or *quotas*. Mailbox quotas are used to control the size of end user mailboxes, and also control the size of messages that may be stored in these mailboxes. The quotas defined for each mailbox are:

- A limit on the total size of all messages in the mailbox.
- A limit on the maximum size of a message that will be accepted for the mailbox.
- A limit on the total number of messages in a mailbox.

When a message is received for an account, but the storage of the message in the account's mailbox would cause one of the above limits to be exceeded, the message is held for subsequent delivery attempts or returned to sender.

An additional mailbox attribute defines a percentage of the mailbox's storage limit at which the end user will be notified of the situation. This *quota warning threshold* is an important method of alerting end users of the possibility of exceeding their mailbox quota.

6.3.1 Setting the Over-Quota Policy

The overall quota bounce policy is defined in the configuration key `bounceOnQuotaFull`. The value of this configuration key determines the action taken by InterMail when a message cannot be delivered because it would exceed one or more of the recipient's mailbox quotas. If set to `true`, this option causes messages to be returned to sender if they would exceed the recipient's mailbox quotas. If set to `false` (the default setting), such undeliverable messages are held 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.

6.3.2 Setting Mailbox Quotas

Although mailbox quotas are enforced in the Message Store Database, the quota values themselves are defined in the Integrated Services Directory. Quotas are typically defined at the level of a class of service, which defines quotas for the accounts associated with it. However, quota values can also be set at the account level, which overrides the quota values defined for the account's class of service.

To set mailbox quotas, use one of the available methods for accessing data in the Integrated Services Directory:

- the `imdbcontrol` utility
- InterManager
- applications that use the InterMail API libraries

Refer to the *Integrated Services Directory Reference Guide* for more information on `imdbcontrol` and the InterMail API libraries. For information on InterManager, refer to the *InterManager Administration Guide*.

6.3.3 Setting Over-Quota Notifications

When a message is bounced because it would have caused the recipient's mailbox to exceed one of its mailbox quotas, the recipient is notified of the event with an automatic notification. This informs the user that a message could not be delivered, and also suggests that the user resolve the situation by removing mail that is currently in his or her mailbox.

Note: *Over-quota notices are not themselves subject to being bounced due to mailbox quotas. This means that an account can receive bounce notifications even if its mailbox is beyond its capacity.*

The text of the over-quota notification message is configurable, and is defined in the configuration key `bounceQuotaNotice`. A default message is defined for this configuration key upon installation, but you may alter this message if 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.
```

Figure 15. The initial over-quota notification message. Items shown between <angle brackets> are replaced by actual data values when the message is sent.

To make changes to this over-quota notice, edit the configuration database (as described in Chapter 3) and modify the value of `bounceQuotaNotice`.

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

Note: *For more information about the* `bounceQuotaNotice` *and* `maxBounceNotices` *configuration keys, see Chapter 11 of the InterMail Reference Guide.*

6.4 Message Aging

A topic related to mailbox quotas in *message aging*. Message aging allows old mail to be automatically deleted form the Message Store Database, even if the messages have not been read by their recipients. This feature is extremely useful for controlling the size of mailboxes and preventing over-quota conditions.

Note: *Message aging is not the same thing as garbage collection (described in Section 6.7). A message can be "aged" only if it currently exists in an end user mailbox and has not been marked for deletion. Meanwhile, garbage collection affects only messages that have been deleted by all recipients on an MSS host.*

Message aging options are particularly important for preventing a common problem: end users keeping mail in their server mailboxes, despite regularly retrieving their mail via the POP Server. Most POP3 clients allow this, but because it can cause read messages to be left on the server indefinitely, it presents a serious concern for the use of server storage.

There are two InterMail message aging policies:

- lifetime limit for messages that have been retrieved via the POP Server
- lifetime limit for all messages

Each of these policies can be set globally for the entire system or for a specific MSS, and can also be independently disabled. The aging policy for read messages is typically set much lower than the limit for all messages, but there is no dependency between the two.

Note: *Mail that is stored in administrative mailboxes, such as the new account welcome message, should be exempted from message aging requirements. This can be done by specifying the name of the administrative mailbox in the configuration key* `adminMessageStoreName`*, which is described in the following section.*

6.4.1 Configuration Options

The following configuration keys are used to define message aging policies:

`lifetimeOnRetrievedMsgsOption`	Enables or disables the aging of messages that have been retrieved via the POP Server. If set to `true`, retrieved messages will be deleted by the `imoldretrmsgdel` command if they are older than the number of days specified in `lifetimeForRetrievedMsgsDays`.
`lifetimeForRetrievedMsgsDays`	Specifies the number of days that a message may be left in a mailbox if it has been retrieved by the end user via the POP Server. This key is applicable only if `lifetimeOnRetrievedMsgsOption` is set to `true`.
`lifetimeOnMsgsOption`	Enables or disables message aging for all mail, regardless of status. If set to `true`, messages will be deleted by the `imoldmsgdel` command if they are older than the number of days specified in `lifetimeForMsgsDays`.
`lifetimeForMsgsDays`	Specifies the number of days that a message may be left in a mailbox. This key is applicable only if `lifetimeOnMsgsOption` is set to `true`.
`adminMessageStoreName`	Specifies the name of the administrative mailbox that should be exempted from message aging policies. This is the administrative mailbox in which special messages—such as the new account welcome message—are stored. Because these messages should not be "aged," you should specify the name of all administrative mailboxes in this configuration key.

6.4.2 Message Aging Administrative Commands

The actual deletion of messages that exceed message aging policies is not carried out by the MSS itself. This task is executed by a pair of administrative commands:

- `imoldretrmsgdel`, which deletes messages that are older than the message aging limit for messages retrieved via the POP Server.
- `imoldmsgdel`, which deletes messages that are older than the message aging limit for all messages.

If you intend to use message aging options, you should set these commands to run on a regular basis as `cron` jobs. The `imoldretrmsgdel` command is typically run once a day, while `imoldmsgdel` is run less frequently (once per week or month).

Note: *For complete information on these administration commands, refer to Chapter 12 of the InterMail Reference Guide.*

6.4.3 Sample Scenario

The following example illustrates the steps required to create the most common message aging configuration for the entire InterMail system. To set message aging policies only for a particular MSS host, replace "`*`" with the desired host name in each configuration database entry shown in this example.

Expiring Retrieved Messages

To define a policy that causes messages to be deleted if they were retrieved by end users more than two weeks ago, execute the following steps:

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

   ```
   /*/mss/lifetimeOnRetrievedMsgsOption: [true]
   ```
3. Locate the entry `/*/mss/lifetimeForRetrievedMsgsDays` in the configuration database, and change its value to 14. This defines a message aging limit of two weeks for mail that has been previously retrieved via the POP Server. If the key does not exist in the configuration database, add it as a new configuration key entry:

   ```
   /*/mss/lifetimeForRetrievedMsgsDays: [14]
   ```
4. Locate the key `/*/mss/adminMessageStoreName` in the configuration database, and enter the names of all administrative mailboxes:

   ```
   /*/mss/adminMessageStoreName: [admin]
   ```
5. Set the `imoldretrmsgdel` administration command to be executed daily as a `cron` job during a time of low mail traffic. The following `crontab` entry executes `imoldretrmsgdel` daily at 3:00 am:

   ```
   * 3 * * * $INTERMAIL/bin/imoldretrmsgdel
   ```

6.5 Moving Mailboxes

Once a mailbox has been created in a Message Store Database, it remains permanently stored on that MSS host. However, there are circumstances in which you may need to relocate mailboxes from one MSS host to another. Among the operations that require the moving of mailboxes are:

- distributing existing mailboxes between a greater number of MSS hosts
- transferring the control of mailboxes stored on an MSS host that will be taken permanently off-line
- temporarily relocating mailboxes while an MSS host undergoes upgrade or maintenance

In each of these cases, a variety of operations must be carried out before mailboxes are accessible on the new MSS host. For each affected account, the following must occur:

1. Delivery of mail must be temporarily suspended for the account.
2. The account's mailbox, folders, and messages must be copied to the destination MSS host.
3. The account must be updated to reflect the new mailbox location.
4. Normal delivery of mail to the account must be restored.

The administrative command `imboxmove` automates the process of moving mailboxes by performing all of these steps for a given set of accounts. Although it is possible to relocate mailboxes with a combination of other administrative commands or operations, `imboxmove` should be used for this operation.

Note: *When moving mail between MSS hosts, the messages may not have the same UIDL—a unique identifier that mail clients use to distinguish messages have been seen and/or read in previous sessions—when moved to the new location. This occurs only when a message has the same UIDL as an existing message on the new MSS host. In this case, end users' clients may consider the moved messages as new and retrieve a second copy during a subsequent session.*

6.5.1 Executing imboxmove

The `imboxmove` command must be executed on the MSS host from which mailboxes are being moved. The general usage of this command is as follows:

```
imboxmove [-i|-b] <destHost> <file> [<file>...]
```

Where:

`-i`	Specifies that `imboxmove` runs in interactive mode, which requires you to type `<Enter>` after each operation.
`-b`	Specifies that `imboxmove` runs in batch mode, which disables all prompts.
`destHost`	Specifies the name of the destination MSS host. This is the MSS host to which mailboxes will be moved.
`file`	Specifies the name of a file that contains the primary e-mail address (`username@domain`) of each account whose mailbox is being moved to the destination MSS host.

Note: *A second usage of* `imboxmove` *(not shown here) is used to move empty mailboxes only. See Chapter 12 of the InterMail Reference Guide for complete syntax of this command.*

When executed, `imboxmove` performs the following operations:

1. Retrieves the list of addresses form the address file specified as a command parameter.
2. Verifies that the status of each of the specified accounts in the Integrated Services Directory is Active. If an account is not active—for example, if the account has been locked—then that account's mailbox cannot be moved.

3. Changes the status of each account to Maintenance, which causes mail sent to these users to be queued internally and not delivered to the MSS. This step is done for all accounts simultaneously, so all of the affected accounts will remain unavailable until `imboxmove` completes its operations.
4. Copies the database records associated with each account—the mailbox, folders, and messages—to the destination MSS host's Message Store Database.
5. Copies the message files associated with the moved mailboxes to the destination MSS host's Message File System.
6. Changes the value of each account's MSS host attribute in the Integrated Services Directory to the new MSS host.
7. Resets the status of each account to Active.

Depending on the number of mailboxes being moved and the volume of messages that they contain, these operations may take a long time to complete. To reduce the time required to move mailboxes, it is recommended that you move mailboxes only a few hundred at a time.

Note: `imboxmove` *does not delete mailboxes from the original MSS host after they have been moved. If you want to delete mailboxes after moving them, use the* `imboxdelete` *command described in Section 6.6.*

6.5.2 Sample Scenario

In this example, the MSS host `venus` is being decommissioned and will be replaced by a new MSS host named `mercury` (this example assumes that InterMail has already been installed on the new MSS host). To move all mailboxes from `venus` to `mercury`, you would execute the following steps:

1. Log in to the MSS host `venus`.
2. Create a file that contains a list of e-mail addresses for the accounts whose mailboxes should be moved. Because `imboxmove` can take a long time to complete, it is recommended that mailboxes be moved in groups of fewer than 1,000. Each line of the file should contain an address in the format:

   ```
   username@domain
   ```

3. Execute `imboxmove`, specifying the address file created in Step 2 as the source for account information:

   ```
   imboxmove -b mercury addressfile
   ```

4. During execution, `imboxmove` will report on the mailboxes that could not be moved (for example, mailboxes whose accounts are locked). You should review this information, resolve the issues that prevented the mailboxes from being moved, and then move these mailboxes in a subsequent execution of `imboxmove`.
5. Repeat the above steps for additional blocks of accounts until all mailboxes have been moved successfully.

6.6 Deleting Mailboxes

Unlike mailbox creation, which can occur automatically, mailbox deletion must be carried out manually. This operation is accomplished with the `imboxdelete` administration command, which deletes a 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 defined in the Integrated Services Directory, which is unaffected by* `imboxdelete`. *Refer to the Integrated Services Directory Reference Guide for information on deleting accounts.*

The general usage of this command is as follows:

```
imboxdelete <host> <mailboxID>
```

Where:

`host`	Specifies the name of the MSS host from which the mailbox is being deleted.
`internalID`	Specifies the internal ID number of the mailbox being deleted.

For example, to delete a mailbox that has the internal ID number `010671` from the MSS host `mercury`, you would execute the following command:

```
imboxdelete mercury 010671
```

When it completes the mailbox deletion, `imboxdelete` confirms that the operation was successful:

```
imboxdelete:  Message store 010671 deleted!
```

6.7 Removing Deleted Messages

An important operation related to managing InterMail message storage is the removal of deleted messages, also known as *garbage collection*. Garbage collection is the mechanism that removes message files from the Message File System when the associated message is considered deleted by the Message Store Database.

To understand how a message becomes “deleted” in the Message Store Database, recall the definition of a message object in this database (as described in Section 6.1.2). For each message, the Message Store Database contains (among other things) a list of pointers to the mailbox(es) and folder(s) in which the message resides. As recipients delete a particular message from their mailboxes, the list of pointers for that message object is updated to reflect these deletions. As long as there is at least one folder that contains a message, that message object—and its associated message file—are still needed. However, once every recipient of the message has deleted it, and the final pointer is removed for the message object in the Message Store Database, the message is considered deleted; at this point, its associated message file must be garbage-collected from the Message File System.

The administration command that deletes messages from the Message File System is `immssgc`. This command is run regularly—by default, once an hour—as a `cron` job. When executed, `immssgc` queries the Message Store Database for a list of messages that have been marked for deletion in the database, and permanently deletes both the database object and the message file associated with each message. Because it may take a long time to complete, `immssgc` is intended to delete only a certain number of messages in one execution.

The following is the syntax of this command:

```
immssgc [-altrb][-c <number>][-h <hours>][-f][-p][-v]
```

Where:

`-altrb`	Uses the alternate rollback segment `RB_IMMSSGC`.
`number`	Specifies the maximum number of messages that should be deleted in a single execution. If no value is given, the default value of 1,000 is used.
`hours`	Specifies the minimum age (in hours) for messages that will be deleted.
`-f`	Causes a “full” garbage collection. This mode checks every message in the Message File System to determine if it is not referenced in any mailbox; if it is not, the message file is deleted from the Message File System. This option is generally not recommended because it is a very slow process that may use system resources inefficiently.
`-p`	Prompts before each batch of messages (implies `-v`).
`-v`	Requests verbose execution.

Warning! Failing to run `immssgc` on a regular basis may cause the size of the Message File System to grow at alarming rates. It is extremely important that you implement regular garbage collection for the Message File System.

7

Mail In Process

Mail in process is mail that has been received by an MTA, but not yet delivered to its intended recipients. While mail is in process, temporary message storage may be required.

This chapter explores the concept of mail in process and covers the following topics:

- Why temporary storage of mail is required
- The location and organization of temporary mail storage
- Automatic vs. manual handling of deferred mail
- A review of mail queuing options
- A discussion of mail throttling and how it can be implemented

Note: *For a discussion of persistent mail storage and related mailbox management, please refer to Chapter 6.*

7.1 Temporary Storage of Mail in Process

All incoming mail is processed through a series of steps to determine the appropriate response from the mail system (delivery, forwarding, automatic response, etc.).

A significant percentage of incoming mail is processed entirely in memory; however, there are circumstances which require the temporary storage of mail in process.

Mail in process may be stored to disk temporarily for any of the following reasons:

- A message exceeds configured limits on size, number of recipients, or the amount of time required to deliver it
- A local server (usually an MSS or Directory Cache Server) is temporarily unavailable
- A remote mail host is temporarily unavailable
- The message is sidelined as suspected junk mail, and requires examination before a decision is made on whether to deliver or bounce it
- Something in the message itself causes an error that prevents delivery

The sections that follow explore each of these areas in detail. The reason for temporary storage is explained and all related controls are discussed.

7.1.1 Mail that Exceeds Size, Time, or Recipient Limits

If a message is large or if it is addressed to a large number of recipients, it is assumed that delivery of the message will 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 for establishing limits at which mail in process is secured to disk. You can set:

- the number of seconds the MTA should wait for a message to be delivered before writing it to disk
- the maximum size (in kilobytes) for a message that can be delivered without being written to disk
- the maximum number of recipients for any message that can be delivered without being written to disk

For each option there is an associated configuration key. The `timeoutServerDelivery` key defines the limit by time, the `maxDirectKb` key defines the limit by size, and the `maxDirectDelivery` key defines the limit by number of recipients.

If a message exceeds any one of these limits it will be secured to disk during processing.

Maximum Delivery Time

Maximum in-memory delivery time is controlled via the `timeoutServerDelivery` configuration key, which specifies the number of seconds the MTA should wait for a message to be delivered before writing it to disk.

If the time required to deliver a message exceeds the limit defined in this key, the message is secured to disk while the delivery process continues without interruption.

For example, if the `timeoutServerDelivery` key is set as follows:

```
/*/mta/timeoutServerDelivery: [3]
```

a client will be kept waiting for no more than three seconds for delivery to be completed. If delivery cannot actually be completed in that time, the message is temporarily stored and acceptance of the message is signaled to the sending server.

Note: *The maximum recommended value for `timeoutServerDelivery` is 5 seconds.*

Maximum Message Size (KB)

The `maxDirectKB` configuration key, sets the maximum size (in kilobytes) for a message that can be delivered in memory. Any message that is larger than this size will be secured to disk, though delivery will normally continue from memory.

For example, to set the maximum message size for in-memory handling to 120 kilobytes, enter the value of the `maxDirectKB` key as follows:

```
/*/mta/maxDirectKB: [120]
```


Maximum Number of Recipients

The `maxDirectDelivery` key limits the number of recipients for a message that will be delivered directly from memory. Any message with more than this number of recipients will be written to disk before delivery is attempted.

For example, if the `maxDirectDelivery` key is set as follows:

```
/*/mta/maxDirectDelivery [20]
```

then any message addressed to more than 20 recipients, is saved to disk (although the actual delivery process continues from memory).

7.1.2 Local Server is Unavailable

Occasionally, mail delivery may be delayed because an MSS or Directory Cache Server is temporarily unavailable. On these occasions temporary message storage is required until the connection is restored, at which time delivery will be re-attempted automatically.

The `deferProcessInterval` configuration key controls the frequency with which delivery is re-attempted for messages deferred for another InterMail server. Setting the key as follows:

```
/*/mta/deferProcessInterval: [20]
```

would result in delivery being re-attempted every 10 minutes (600 seconds).

7.1.3 Remote Mail Server is Unavailable

InterMail must also store mail temporarily for remote mail domains when the remote mail server is unavailable. Delivery of mail deferred for this reason is re-tried periodically until the problem is resolved. No direct intervention is required.

You can determine the length of time between attempts to deliver outbound mail. The more frequent the re-processing attempts, the sooner mail is likely to be delivered. The tradeoff on more frequent delivery attempts is that it places a greater demand on the MTA's resources.

To change the outbound retry interval from one minute to some other value, use `imconfedit` to modify the `outboundDeferProcessInterval` configuration key. The following syntax would change the setting to one retry every 3 minutes:

```
/*mta/outboundDeferProcessInterval: [180]
```

7.1.4 Sidelined Mail

Message sidelining is a method of deferral over which you have total control. Basically, it allows you to "hold" messages that you suspect are unsolicited commercial e-mail (UCE).

When InterMail's mail sidelining option is activated, messages that meet the criteria you define are moved to a temporary location, from which you can view and process them at your leisure.

See Chapter 4 for a complete discussion of the options regarding sidelining mail.

7.1.5 System Errors

Sometimes a problem with the message itself may cause mail to be deferred, as when mail for an unknown user cannot be bounced because the `MAIL FROM:` address is invalid.

Note that mail held in error is considered part of temporary mail storage. One of the obligations of a system administrator is to review and dispose of mail held in error.

7.2 Format of Mail in Process Files

Mail in process is stored differently from mail that has been delivered to users. (For a discussion of permanent mail storage, see Chapter 6 in this manual).

While in process, there are three files associated with each message:

- Control file
- Header file
- Body file

These three files are associated via a common prefix. The prefix is unique for each set of files and consists of a time stamp, a 3-letter count of the number of messages received (the first is AAA, the next is AAB, etc.), the process id (pid), the MTA hostname, and the HELO/EHLO or the IP address, assuming that one or the other exists.

An example prefix might appear as follows:

```
19980601171253.AHA6542.mta1@pluto.software.com
```

Each of the three files bears this prefix, and ends with a label that identifies the particular message component. The Control file will have the prefix followed by "-Control". The Header file will have the prefix followed by "-Header." The Body file will have the prefix followed by "-Body." Viewed together, the files might look like this:

```
19980601171253.AHA6542.mta1@pluto.software.com-Control
19980601171253.AHA6542.mta1@pluto.software.com-Header
19980601171253.AHA6542.mta1@pluto.software.com-Body
```

Control Files

Control files contain information about the message, including its current status in the delivery process and the names of the corresponding Header and Body files.

Header Files

Header files contain the header information for messages, including `TO:, CC:, BCC:, FROM:, SENDER:`, and `REPLY-TO:` addresses.

Body Files

Body files contain the body of the message—the text and any attachments.

Note: *Upon successful delivery of a message, the temporary files (Control, Header, and Body) are deleted.*

7.3 Storage of Temporary Mail in Process

Mail in process is generally stored on the Queue Server, in a directory whose path is indicated by the `queueDir` configuration key (typically `$INTERMAIL/queue`).

The only exception occurs if the Queue Server host is temporarily unavailable, in which case mail that must be stored temporarily is spooled locally on the MTA host. In both cases, the structure of the queue directory is identical. However, if mail is being temporarily stored on the MTA, the root for the entire structure is called `spool` rather than `queue`.

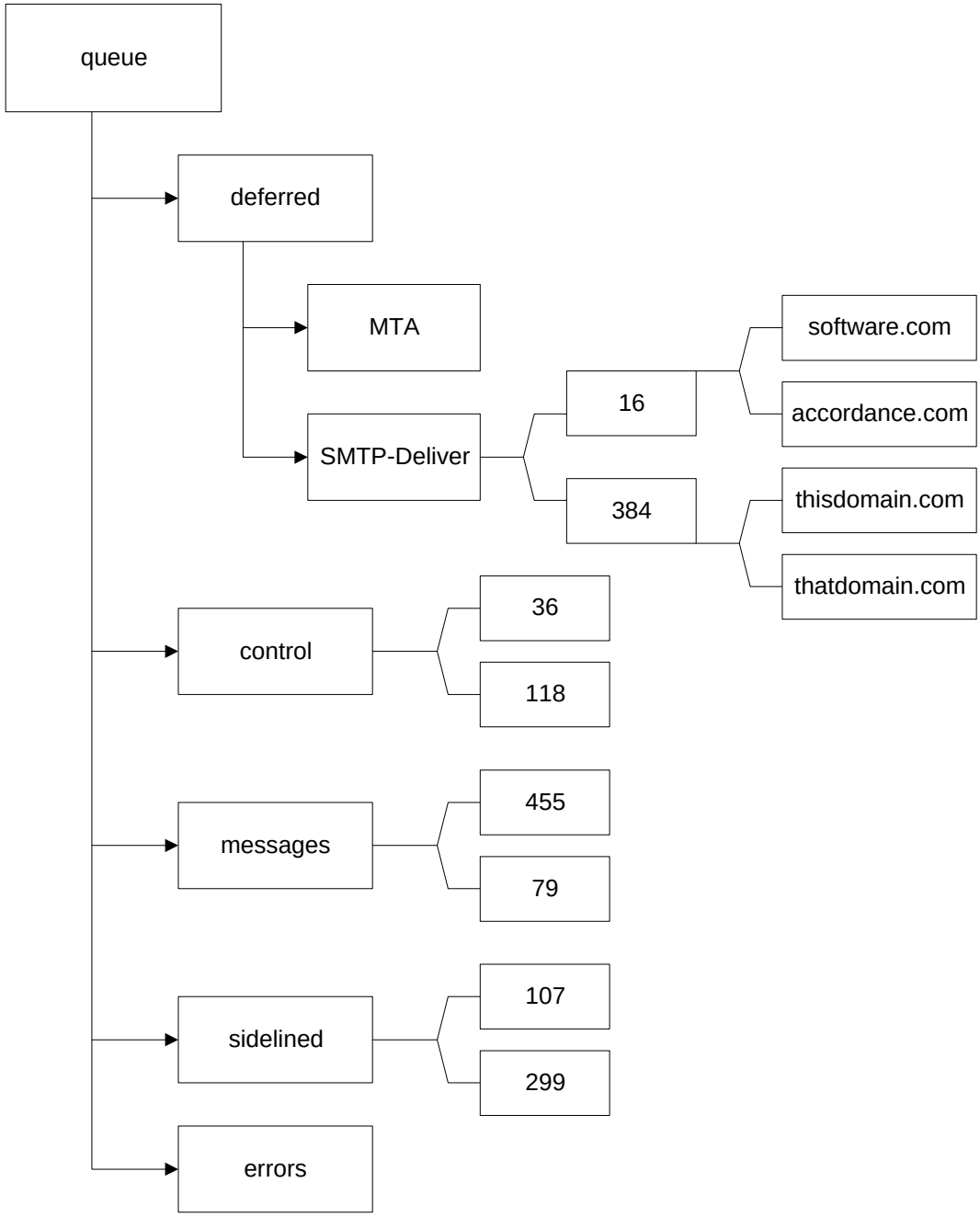

Figure 16. The Queue Server's queue directory

Figure 16 illustrates the structure of the `queue` directory. The numbered items represent *buckets*—a series of numbered subdirectories.

Buckets allow further subdivision of files within the directory structure. As having many processes/threads trying to access the same directory would be slow, the use of buckets yields a significant performance gain.

Note: *The `bucketCount` configuration key controls the number of bucket directories. Its value is set at time of installation and should never be changed once the system is operational.*

The location of files within the `queue` directory varies based on the reason for storage. The sections that follow define the various storage arrangements.

Mail that Exceeds Size and Recipient Limits

When mail is deferred because it exceeds the maximum delivery time, message size, or number of recipients allowed for direct delivery, its component files are stored in the following manner:

- Control files are placed in the `queue/control` directory.
- Body and Header files are stored in the `queue/messages` directory.

Mail that is Undeliverable Locally

When a message is deferred because of an unavailable MSS or Directory Cache Server (as might happen when there is a temporary problem with a local host), its component files are stored in the following manner:

- Control files are stored in `queue/deferred/mta`
- Body and Header files are stored in within a bucket (a subdirectory named with a random number) in the `queue/messages` directory. For example: `queue/messages/1257`.

Mail for an Unavailable Remote Mail Host

When a message is deferred because a remote mail host is unavailable, its component files are stored in the following manner:

- Control files are stored in the `queue/SMTP-Deliver` directory. Queued mail for each domain is stored in its own subdirectory within one of the numbered buckets. For example: `queue/SMTP-Deliver/345/accordance.com`.
- Body and Header files are stored in numbered buckets in the `queue/messages` directory.

Sidelined Mail

When mail is sidelined, the Control, Body and Header files are all stored in the `queue/sidelined` directory. Each message has its own numbered bucket (subdirectory), containing the Control, Header and Body files for that message.

Mail that is Undeliverable Due to an Error Condition

When mail is deferred because of an error condition, Control, Body and Header files are all stored in `queue/errors`.

7.4 Managing Mail In Process

InterMail handles most mail in process automatically. Mail that is stored because it exceeds message size limits continues to be processed and is delivered as soon as possible. Mail that is 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 that has been deferred due to error.

It is also a good idea to keep an eye on your domain mail queues to make sure they aren't becoming so large as to compromise system performance.

7.4.1 Reviewing and Reprocessing Sidelined Mail

This section describes methods for reviewing and processing mail that has already been sidelined. A discussion of when and how to sideline mail can be found in Chapter 4.

The Sideline Directory

When you enable one of InterMail's sidelining options, incoming mail that meets the conditions specified is neither delivered nor bounced. Instead, it is moved to the `queue\sidelined` directory, where it will remain until you take some sort of action. Sidelined mail is never deleted automatically, so,, if you elect to use sidelining, it is important that you check this directory on a regular basis in order to prevent these messages from piling up.

When a message is sidelined, the Control, Header, and Body files for that message are all moved to the sideline directory, into a numbered bucket (subdirectory).

For example, when a message with the id

```
19980114235158898.AAA250@venus.software.com
```

is sidelined, the following message component files are moved to a numbered bucket in the `queue/sidelined` directory:

```
19980114235158898.AAA250@venus.software.com-Control
19980114235158898.AAA250@venus.software.com-Header
19980114235158898.AAA250@venus.software.com-Body
```

Evaluating Sidelined Mail

Let's say that a message for more than 100 recipients has landed in your sideline directory and you must now decide what to do with it. If the message is actually junk mail, you'll probably want to delete it. But if the message turns out to be legitimate, you'll want to see that it's delivered to all of its intended recipients.

As described in the previous section, the sidelined message has three components: Control file, Header file and Body file. The most useful information for evaluating the message will probably be in the Body file, since this contains the actual text of the message. To review this file, you simply open it using any text editor.

Deleting Sidelined Mail

If the message is junk mail, you can delete it with a normal operating system command. The following example removes all three files associated with our sample message:

```
rm 19980114235158898.AAA250@venus.software.com-*
```

Delivering Sidelined Mail

If the message is valid and should be delivered to its intended recipients, you will need to use the `immsgprocess` administrative command to reintroduce the message for normal delivery. The `immsgprocess` utility moves the Control file of the message to the `queue/deferred` directory, while the Header and Body files are moved to the appropriate bucket in the `queue/messages` directory. The message is now treated as ordinary deferred mail, and the MTA will attempt to deliver it to all its intended recipients at the next configured `ouboundDferProcessInterval`.

Usage

The format for `immsgprocess` is in the form:

```
immsgprocess <Control file>
```

Where:

`Control File`	The ID of the control file

Example

The syntax for reprocessing our sidelined example message so that it will be delivered to all intended recipients would be:

```
immsgprocess 19980114235158898.AAA250@venus.software.com-Control
```

7.4.2 Reprocessing Mail Deferred Due to System Errors

Most messages received by the MTA are either delivered to their intended recipients, or else bounced when the MTA attempts delivery to a local domain where a recipient is unknown. But situations may arise in which the MTA is unable to return a message because the original sender's return address is invalid. In such cases, all three message components (Control, Body and Header files), are placed in the `queue/errors` directory. In most cases, you will simply want to delete these files, since there is generally no way to fix the problem. It is important to keep an eye on this directory, though, because with large mail installations it can fill up very quickly.

7.4.3 Reprocessing Mail Deferred Due to Unavailable Hosts

Messages that are deferred because a remote mail domain is temporarily unavailable are reprocessed automatically by InterMail. At configurable intervals, the MTA requests these messages from the Queue Server and attempts to deliver them again. However, even though this process is automatic, there are times when some human intervention may be useful, as explained in the following sections.

Splitting Queues

InterMail creates a separate queue directory for each remote domain to which it cannot deliver mail immediately. In very large mail systems, it is not uncommon for some queues to become quite large. This is especially true when a very busy remote site is off-line for a considerable length of time. While this remote site is down, thousands of deferred messages may pile up in its queue directory; and the larger the queue becomes, the longer it will take for additional messages to be written there. Splitting a very large queue into two or more smaller queues can thus speed the actual queuing process and enhance overall system performance. Splitting large queues can also help speed delivery once an unavailable mail host comes back on line. This is true because InterMail opens just one connection per queue when it re-attempts delivery. If there are several queues per domain, InterMail can open several connections to that domain and deliver messages from these multiple queues simultaneously.

The administrative command for splitting queues is `imqueuesplit`.

Splitting a queue

The syntax for `imqueuesplit` is:

```
imqueuesplit <destdomain> <newdest> [<newdest>]
```

Where:

`destdomain`	The name of the original queue directory
`newdest`	One or more new queue directories you wish to create.

Example

To wind up with a total of four queue directories, you only need create three new ones, since the original directory will still remain:

```
imqueuesplit accordance.com a.mx.accordance.com b.mx.accordance.com
c.mx.accordance.com
```

The line begins with `imqueuesplit`, which initiates the command. The `accordance.com` portion specifies the destination domain (the original queue directory for `accordance.com`). `a.mx.accordance.com`, `b.mx.accordance.com`, and `c.mx.accordance.com` are the names for the three additional queue directories you wish to create.

In this example, `imqueuesplit` first creates the specified new queue directories. It then moves approximately 75% of the messages from the original `accordance.com` queue into the three new directories, balancing the load as equally as possible between the four total directories. The `imqueuesplit` utility always seeks to balance the message load equally, no matter how many queues you create. With four queue directories for `accordance.com`, the MTA would now establish four connections to this domain when it begins to deliver deferred mail.

A Queue Splitting Strategy

How many queues you split off may depend on a number of factors, including the number of threads your system has available for MTA mail delivery. It will also depend on how many connections the domain that has been off line is able to accept.

Suppose that `accordance.com`, a large remote domain, has been unavailable for several hours and then comes back on line. You have a couple of megabytes worth of mail queued for `accordance.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 will need to know how many other network addresses are delivery points for `accordance.com`. You can then create one new `accordance.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 accordance.com
```

Let's say this returns the following

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

This tells you that there are four MX records for `accordance.com`, which means that you can (at least theoretically) have four queues—and four simultaneous connections—to `accordance.com`.

But before you split your `accordance.com` into four different queues, there are a few things you may want to consider. First of all, determine how much mail do you actually have to send. If the amount isn't that large, opening four connections may be a waste of resources for both you and `accordance.com`. Part of the equation here might be to ping the `accordance.com` servers to see how quickly they are responding. If you get quick responses, can probably get your mail delivered in an acceptable time with just a couple of connections. On the other hand, if the `accordance.com` servers respond very slowly, it may indicate that they are getting a lot of traffic. (Remember they've been down for a while and now everyone in the world is trying to deliver mail there.) In this case, it may make sense to spread your delivery load over a larger number of connections in order to avoid server timeouts.

It is also possible that `accordance.com` might not permit you to make four simultaneous connections to their site for some reason. Another possibility is that that `accordance.com` does not even publish the other MX records that point to its address, in which case your `nslookup` would not return anything but the address of its main mail exchanger.

Another possibility is that the above MX search may turn up several records with different preference levels. That is, instead of all four having a preference value of 10, they may be 10, 15, 20 and 25. This would probably indicate that the three servers with higher numbers are actually backup mail servers, and it might create a problem for the `accordance.com` domain if you tried aimed your queues there. This is because the backup servers at `accordance.com` would then have to reprocess this mail for delivery.

But assume that all the MX records for `accordance.com` have the same preference rating (i.e., they are all primary mail servers) and you decide to use four simultaneous connections to deliver your queued mail. To enable the MTA to open four simultaneous connections, you will need to split `accordance.com's` current large queue into four smaller, more manageable queues.

Queue Splitting Reports

When it is finished executing, `imqueuesplit` prints out the number of messages it has processed and the domain to which messages are being routed.

Usage

The format is in the form:

```
Routing <number> messages to <domain>
```

Where:

`number`	The number of messages routed from the original queue to the new queue
`domain`	The name of the new queue

Example:

A typical `imqueuesplit` command, followed by a report for a successful operation might look like this:

```
imqueuesplit software.com a.mx.software.com b.mx.software.com
c.mx.software.com
```

```
Routing 264 messages to a.mx.software.com
Successful. 264 of 264 messages were processed.
Routing 264 messages to b.mx.software.com
Successful. 264 of 264 messages were processed.
Routing 263 messages to c.mx.software.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>!
```

Each time `imqueuesplit` is done with a domain, it prints one of the following messages:

```
Successful. <number> of <number> messages were processed.
Failed! Couldn’t move the new queue into place.
Failed! No messages were processed.
```

7.5 Queuing Options

InterMail offers a variety of other options that affect how mail is queued and re-processed. Queuing behavior is determined by setting a variety of configuration keys.

Each of the following sections discusses a queuing option and the configuration keys used to set that option.

7.5.1 For Servers that Are Unavailable

You can determine the length of time between the MTA's checks for local or outbound queued mail. After each configured interval has elapsed, the MTA asks the Queue Server for a list of domains with deferred mail and attempts to deliver that mail. If delivery is again unsuccessful, the mail is re-queued until the next processing interval.

The more frequent the re-processing attempts, the sooner mail is likely to be delivered. The tradeoff on more frequent delivery attempts is that it places a greater demand on the MTA's resources.

Modifying the Defer Process Interval for Local Mail

To change the retry interval for local from 10 minutes to some other value, use `imconfedit` to modify the `deferProcessInterval` configuration key. The following syntax would change the setting to one retry every 5 minutes:

```
imconfedit deferProcessInterval [300]
```

Modifying the Defer Process Interval for Outbound Mail

To change the outbound retry interval from one minute to some other value, use `imconfedit` to modify the `outboundDeferProcessInterval` configuration key. The following syntax would change the setting to one retry every 3 minutes:

```
imconfedit outboundDeferProcessInterval [180]
```

Note: *In most cases, both the* `deferProcessInterval` *and* `outboundDeferProcessInterval` *configuration keys reprocess queued mail from Queue Server's* `queue` *directory. However, if mail is queuing in the MTA's* `spool` *directory because the Queue Server is unavailable, both keys will reprocess mail from the* `spool` *directory.*

Minimum Queue Idle Time

The setting for `minQueueIdleTime` tells the MTA queue scanner how much time must elapse before a given queue can be processed again. Queues are scanned every `outboundDeferProcessInterval`, but they are not actually processed unless their `minQueueIdleTime` has been exceeded.

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, if a large queue for `accordance.com` was processed just ten minutes ago, during the last `outboundDeferProcessInterval`, you might want to devote more resources to processing other queues when the next `outboundDeferProcessInterval` comes around. In that case, you would set `minQueueIdleTime` to some value that is higher than the value for `outboundDeferProcessInterval`. Now, during each `outboundDeferProcessInterval`, the queue scanner checks to see how long each queue has been idle, and only those queues that have been idle longer than the time specified in `minQueueIdleTime` are processed.

Note: *Unless* `minQueueIdleTime` *is set to a higher value than* `outboundDeferProcessInterval`*, this feature will have no affect. This is because the allowed idle time for queues will always have been exceeded each time they are scanned.*

Modifying the Minimum Queue Idle Time

To change the setting for how long queues must remain idle, use `imconfedit` to modify the `minQueueIdleTime` configuration key. The following syntax would change the setting to twenty minutes (values are in seconds):

```
/*/mta/minQueueIdleTime: [1200]
```

Maximum Time in Queue

Occasionally it will be impossible to deliver a message, as when a domain to which a message is addressed becomes unavailable and does not come back on line within a reasonable amount of time.

Exactly what a "reasonable amount of time" might be is basically for you to decide. However, Internet standards recommend a period of at least four or five days. The initial value set by InterMail is four days. Once the maximum period that you have defined expires, the MTA generates a bounce notice and returns the message to its sender.

Modifying the Minimum Queue Idle Time

To change the maximum amount of time a message can stay in the queue from the default value of one half day (.5) to some other value, use `imconfedit` to modify the `maxQueueTimeInDays` configuration key. The following syntax would change the setting to two days:

```
/*/mta/maxQueueTimeInDays: [2]
```

Always Try Delivery Before Queuing

If mail is already queuing for a domain, InterMail *will not* (by default) attempt to deliver additional messages to this domain. Instead, InterMail assumes that repeated delivery attempts would be a waste of time, and sends any new messages for that domain directly to the Queue Server, which adds them to one of the domain's existing queues. This setting (`false`) is the default behavior for the entire system, but you can change it. You should bear in mind, though, that changing this behavior affects all domains for which mail is queuing at any given time.

Modifying the Always Try Setting

The following syntax tells InterMail to always attempt to deliver a message before adding it to an existing queue. (Since InterMail will open a connection for each of these attempts, doing this may have some impact on system performance if a domain is unavailable for a long period of time):

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

Always Queue Before Attempting Delivery

There may be situations in which you never want the MTA to attempt immediate delivery, regardless of whether or not there is a problem that would ordinarily cause messages to queue. One example might be where there is a remote domain to which you send large volumes of mail. If, for example, your MTA sends one message to `software.com` every five or ten seconds, you may decide that it is a waste of system resources to establish so many connections to the `software.com` domain.

You can limit the number of connections to a domain over time by telling InterMail to create a queue for every domain, and store all messages there until the next configured `outboundDeferProcessInterval`—at which time, the MTA attempts to send all queued messages for a domain during a single connection.

Bear in mind that there is a tradeoff here. Setting `alwaysQueue` to `true` causes all outgoing messages to be queued, regardless of whether or not there is a problem that would ordinarily cause mail to be deferred. This implies that all mail delivery will be delayed, since no messages are ever processed immediately.

A setting of `true` may be most useful in cases where your MTA has an intermittent connection, and you need to queue mail until the next opportunity to deliver it. Another practical application of setting this to `true` might be in cases where a large, remote host is hanging, since you might wind up with a thread hanging for several minutes for each outgoing connection.

***Note**: If* `alwaysQueue` *is set to* `true` *and* `alwaysTryFirst` *is also set to* `true`*, then* `alwaysQueue` *takes precedence, and mail will always be queued.*

Modifying the Always Queue Setting

The following syntax tells InterMail to always queue messages before attempting delivery, regardless of whether or not a domain is already queuing mail:

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


Processing Interval Configuration Keys

The following table summarizes the configuration keys used to set options for mail that is deferred due to unavailable local or remote domains. A more complete description can be found in Chapter 11 of the *InterMail Reference Guide.*

`deferProcessInterval`	This key sets the interval between delivery attempts for queued local mail. Local mail is deferred when a mailbox or user account information is temporarily unavailable. The default setting is 600 seconds, which means that the MTA will retrieve deferred messages from the queue and try to deliver them again once every ten minutes. ***Note:*** *Delivery is re-attempted regardless of whether mail is queued on the Queue Server or in the MTA's* `spool` *directory.*
`outboundDeferProcessInterval`	This key sets the interval 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 messages that it had previously deferred. ***Note:*** *Delivery is re-attempted regardless of whether mail is queued on the Queue Server or in the MTA's* `spool` *directory.*
`minQueueIdle`	This key sets the minimum time (in seconds) between attempts to deliver queued mail to a given domain. The default value is 60 seconds.
`maxQueueTimeInDays`	This key specifies the maximum number of days that a message can be kept in the queue. Once a message has been in the queue for the time defined here, InterMail assumes that it cannot be delivered and returns it to its sender.
`alwaysTryFirst`	Use this key to tell InterMail whether or not to attempt delivery for each message before queuing. The possible values are either `true` or `false`. Changing the default value from `false` to `true` means that the MTA will always attempt delivery of a message before sending it to the Queue Server or queuing it in its own spool directory.
`alwaysQueue`	Setting the `alwaysQueue` option to `true` means that InterMail will queue every message for a domain and attempt to send them all during a single connection. The default setting is `false`.

7.5.2 ETRN (SMTP Queue Processing Requests)

Yet another mail queue processing option is the SMTP ETRN command, which can be used to instruct remote mail hosts to attempt delivery of queued messages This is useful if you have a PPP or a SL/IP connection, or have a similarly intermittent connection to the Internet. Although ETRN is designed to be used by connecting mail servers, you can also execute it manually to request queue processing.

Usage

To request manual queue processing, log in to port 25 of the SMTP server and enter ETRN followed by "@" and the domain of the queue that should be processed. This will cause your queued mail to be delivered immediately, regardless of when the next `deferProcessInterval` is scheduled to occur.

Example

The following console session shows the result of an ETRN command requesting immediate delivery of mail that has queued for `accordance.com`:

```
220 venus.software.com ESMTP server (InterMail v4.0 212) ready Tue, 12 May
1998 09:20:30 -0700
HELO
250 venus.software.com
ETRN @accordance.com
250 Ok
QUIT
```

The `250 ok` response acknowledges that that the request has been carried out.

Note: *ETRN is an open protocol standard, and is defined in RFC 1985.*

7.6 Throttling Mail in Delivery

Mail in delivery is a subset of mail in process. It represents those messages that are being actively processed, and does not include those items that have been explicitly deferred (for attempted re-delivery at the configured interval).

InterMail offers a series of controls to assist in managing mail in delivery. Most will be used infrequently, if ever, nevertheless they provide a valuable safety net to protect against excessive system load.

The three configuration keys involved in throttling mail flow are as follows:

- `inDeliveryDeferKb` sets a limit at which incoming mail will automatically be deferred (allowing all processing efforts to be focused on existing mail in delivery)
- `inDeliveryRejectKb` sets a limit at which incoming mail will automatically be rejected (with notice that the server is busy and re-delivery should be attempted at a later time)
- `inDeliveryStopDeferKb` sets a limit at which deferred mail processing will cease (to allow the MTA time to catch up)

Note: *Mail that is deferred via throttling is treated as any other deferred mail, and is reprocessed automatically at regular intervals.*

The value of each key is measured in kilobytes. A value of zero means unlimited. The default values are as follows:

```
/*/mta/inDeliveryDeferKb: [1000000]
/*/mta/inDeliveryRejectKb: [0]
/*/mta/inDeliveryStopDeferProcessKb: [10000]
```

The mail in delivery management keys are designed to be used in a complimentary manner. Therefore, the relative values are significant.

For example, if the value of the `inDeliveryDeferKb` key is set to 100000 (the default), the value of the `inDeliveryRejectKb` key must be set higher than 100000 in order to reserve the more drastic measure for last. The `inDeliverDeferKb` limit, which is reached first, merely defers mail, rather than rejecting it outright.

The value of the `inDeliveryStopDeferProcessKb` key is unique in that it is checked at the start of each defer 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 at the next defer processing interval.

Note: *Log entries are written each time one of the mail throttling controls is activated. Log files should be reviewed regularly for* `MtaTooBusyDefer`, `MtaTooBusyReject`, *and* `MtaTooBusyStopDefer` *entries, evidence of excessive system load.*

8

Logging Overview

This chapter discusses InterMail logging operations. It describes the various types of log files, explains how to read log data, and discusses log file management.

Topics covered in this chapter are:

- Types of log files
- Log file formats
- Reading log files

8.1 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. You can configure how much information gets logged.

Log files are very useful for debugging since you can retrace the steps that led to an event or problem. You can also use information in log files to monitor system performance.

8.1.1 Types of Log Files

InterMail generates three kinds of log files as shown in the following table.

Log File Type	Contents	Suffix
Event log files	Error, warning and informational messages recorded as the events occur.	`.log`
Statistics files	Statistical information for a period of time.	`.stat`
Trace files	Diagnostic information recorded as events occur. This is typically turned on at the direction of technical support.	`.trace`

All of the above log files are explained in detail in the following subsections. For troubleshooting purposes, `.log` files are the most significant of these three types. You should regularly review the `.log` files as part of routine system monitoring.

8.1.2 Specifying the Log Directory

All log files are written 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 Chapter 11 in the *InterMail Reference Guide*.

8.1.3 Managing Log Files

Rollover is a means of controlling the size of log files, and provides a means of conveniently archiving log files to secondary storage without disrupting running applications. When a log file rolls over, the old file is closed, a new log file created, and logging continues to the new file.

To specify how many times per day the log files will "roll over" use the configuration key `rolloversPerDay`. For example:

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

In this example, log files will roll over two times a day. If the value of this key is 0, then rollover is disabled altogether.

Rollover is always calculated in terms of GMT. By default, the first rollover period of a day starts at midnight (00:00:00 GMT). However, you can offset this by setting the configuration key `rolloverTimeZero` (expressed in seconds) which defines offset from GMT. For example, if the following is specified:

```
/venus/common/rolloverTimeZero: [300]
```

then rollover would begin at 00:05:00 hours GMT.

Tips to Manage Log Files

The following guidelines will help you maintain/organize log files more efficiently.

- Set the rollover time for log files to a reasonably quiet period. You can configure the rollover start time and the interval between rollovers using the `rolloverTimeZero` and `rolloversPerDay` configuration keys.
- Compress and/or archive log files regularly. This requires a customer written script that would be run as a cron job.

It is convenient to leave the event log files for the last few days uncompressed; you will probably be accessing them frequently. It is recommended that you archive log files older than a week which will save significant space.

8.2 Event Log Files

All InterMail server processes (e.g., `mss`, `mta`, `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. A new log file is then started.

Current log files are referenced in files that are identified by a server name and the file type (for example, `popserv.log` contains a reference to the current log file for popserver). Archived log files are similarly referenced in files that are identified by 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 maintained in local time or GMT. If time stamps are required to be in local time, the configuration key `gmtLogTimes` should be set to false (the value is false by default). To maintain the time in GMT, set the value of the `gmtLogTimes` key to true.

If the time is in local time, the log file name will have the timezone offset.

By default, event logs are not printed to `stderr`. To turn this capability on, set the value of the configuration key `servWarnToStderr` to true. For example,

```
/*/common/servWarnToStderr: [true]
```

This will print all event logs of severity level Warning and higher to `stderr`. See section 8.2.2 later in this chapter for a description of severity levels.

8.2.1 Event Log File Format

Inside each event log file, numerous log “events” are reported. The format for entries in a event log file is:

```
<logEntry>::= <date> <time> <host> <program> <pid> <lwp> <thread>
<severity> ";" <event>
```

Example:

The following example shows a typical event log in a event log file.

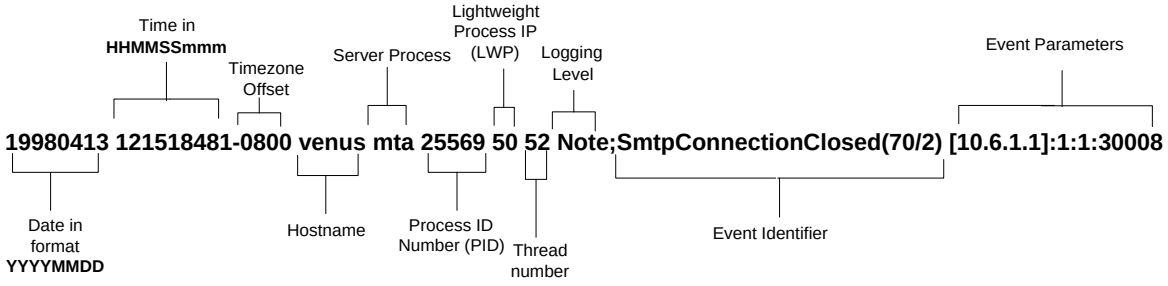

Figure 17 Event Log Format

Event logs are comprised of the following fields:

Date Stamp: in YYYYMMDD format

Time Stamp: in local time (HHMMSSmmm) or GMT, where mmm is in milliseconds. If the time stamp is in local time, the timezone offset is also displayed.

Hostname: the machine name where the server resides

Server Process: the name of the process which logged an event (e.g. mta, popserv, imdircacheserv)

Process ID number (PID): standard PID of the operating system

Light Weight Process ID (LWP): 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. These levels 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). See section 8.2.2 for additional information.

Event Identifier: Each type of logged event has a specific name containing: a prefix denoting the general event category, an abbreviated description of the event, and a Message Set ID and Message ID (a “non-human readable” code that refers to a catalog of message types used by other commands to produce a more readable format).

Event Parameters: Details about the logged event. Includes information to help trace data through a system, trace parent-child relationships, diagnose OS dependent problems, etc.

There are a few guidelines for the format of this field:

- Event parameters are colon delimited.
- Events associated with a network connection will contain the network address of the connection.
- Message events will include a message ID as their first parameter.
- Events associated with a specific client identity will provide the user name when available.
- Process IDs should be supplied to establish parent-child relationships when created/reaped.

In the example shown in Figure 17 Event Log Format, a notification is given that on 13 April 1998, at 12:15 PM local time, the MTA had an SMTP connection closed on the host named `venus` from the IP address 10.6.1.1. one second after connection time. The client sent 1 message of length 30008 bytes.

An event log may also include, at the end, any of the following colon-separated additional information:

`"user=%s"`	mailuserString
`"mss=%s"`	mailhostString
`"port=%d"`	portString
`"mbox=%s"`	mailboxString
`"from=%s"`	fromString
`"fldr=%s"`	folderString
`"msgid=%s"`	messageIdString
`"origMsgid=%s"`	origMessageIdString
`"size=%d"`	sizeString
`"cmd=%s"`	cmdString

`"rme=%s"`	rmeOpString
`"fromhost=%s"`	fromHostString
`"desthost=%s"`	destHostString

For example,

```
19980507 133423832-0700 venus 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
```

8.2.2 Troubleshooting Event Logs

When troubleshooting log events, there are two strategies to keep in mind. First, there are events that should be monitored due to their severity level; for example, Error, Urgent, and Fatal events all are indicative of an "error condition" that should be immediately diagnosed and remedied. In some cases, however, the frequency of events (regardless of the severity level), is indicative of a problem which can affect system performance. For example, ten bad password attempts in an hour may be expected behavior, whereas 1000 attempts might indicate a bid to break into the system.

Troubleshooting Event Logs by Severity

Log events each have a severity level that indicates how serious a particular event is and to what extent InterMail has been affected by an event. However, it should be noted that all similar errors may not have the same impact on the system. For example, failing to open a configuration file could be fatal, while failing to open a message file, while certainly an error, does not prevent the program from operating further.

The following table shows the five severity levels that are reported in log files.

Event Type	Meaning
Fatal (Fatl)	An unrecoverable event has occurred that prohibits further operation of a process. The process should exit and, if possible, restart or reschedule the work that it was performing. Critical errors require some action on the part of an operator even if only to verify that a process has restarted.
Urgent (Urgt)	An error has occurred that requires action by an operator.
Error (Erro)	A minor error has occurred. If the error persists, i.e., the error occurs more than a certain (threshold) number of times, operator intervention is required. You will need to develop your own threshold values based on event type, number of messages, number of users, etc.
Warning (Warn)	Events have occurred that do not affect the operation of the software. These events describe something that ideally should not happen (e.g., an error between an SMTP client and the MTA). Warnings should be monitored, although these events do not affect overall InterMail performance.
Notification (Note)	Notifications inform the user of typical events that are used to trace data-flow and server processes. This is expected behavior.

Troubleshooting Event Logs by Frequency

In certain situations, the frequency of notification and warning log events may indicate a persistent problem that can affect messaging throughput and system performance. The following table shows the most common of these types of log events and what type of problem is indicated if the event occurs frequently.[6]

Log Event(s)	Type of Problem
`AcctBadPswd`	A possible POP password attack; an unauthorized user is attempting to enter the system or a user or multiple users are attempting to launch a "denial of service attack" by flooding the POP server.
`ProcExitCode`	InterMail server processes have exited with error codes; this may be indicative of other system errors.
`SmtpConnectionDropped`	Connections have been dropped due to a violation of system policy; this may indicate an attempt to distribute unsolicited e-mail.
`SmtpRelayPrevented`	Relay attempts have been denied due to a violation in system policy; this may indicate an attempt to distribute unsolicited e-mail.
`SmtpSenderBlocked`	A message sent had recipients rejected because the sender was blocked; this may indicate an attempt to distribute unsolicited e-mail.

See Chapter 13 of the *InterMail Reference Guide* for descriptions of all event logs that appear in InterMail.

8.3 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 are generated by the server. You can specify how frequently you want the `.stat` file generated by using the `reportParamsInterval` configuration key. The default value for this key is 180 seconds. After the specified interval, the server checks if any of the reportable parameters have changed and reports only the changed reportable parameters.

Current statistics files are referenced in files that are identified by a server name and the file type (for example, `popserv.stat` contains a reference to the current log file for popserver). Archived statistics files are similarly referenced in files that are identified by a server name, a host name, a date/time stamp, and the log type (for example, `popserv.venus.199804210000-0700.stat`).

The sections that follow describe the statistics logged by each InterMail server.

[6] What constitutes "frequently" will depend on a variety of factors, including your specific hardware configuration and account database.

8.3.1 Statistical File Format

The format for entries to a statistical log file is:

```
<rplLogEntry>::= <date> <time> <host> <program> <pid> ";" <event>
```

Example:

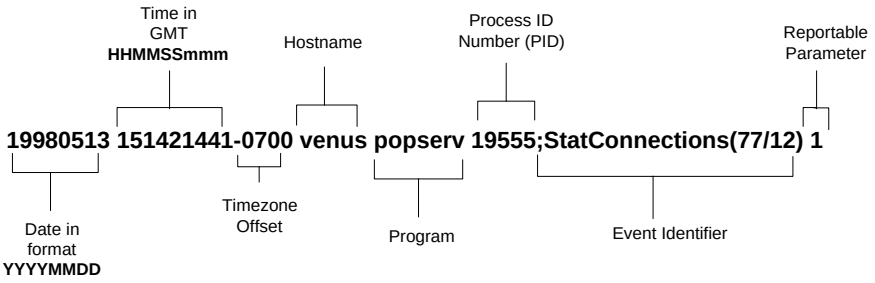

Figure 18 Statistical Log File

Statistics file entries are comprised of the following fields:

Date Stamp: in YYYYMMDD format

Time Stamp: in local time or GMT (HHMMSSmmm), where mmm is in milliseconds. If the time stamp is in local time, the timezone offset is also displayed.

Hostname: the machine name where the program resides.

Program: the name of the program which logged the statistic (e.g. mta, popserv, imdircacheserv).

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

Event Identifier: Each type of reportable parameter has a specific name (an abbreviated description of the statistic being reported). It also contains a general event category number, and a Message ID (a “non-human readable” code that refers to a catalog of message types used by other commands to produce a more readable format).

Reportable Parameters: Statistics being reported.

8.3.2 POP Server Statistics

The POP Server handles requests for message retrieval from any client that supports the POP3 protocol.

Event	Description
`StatStartUp`	POP Server startup time
`StatConnections`	Total number of current connections
`StatAccumulatedConnections`	Number of total connections accumulated
`StatFailedConnections`	Number of failed connections (connections fail when an attempt to create a socket for the client fails, typically because the maximum number of connections was reached.)
`StatRejectedConnections`	Number of rejected connections (connections are rejected if authentication fails.)
`StatTimedOutConnections`	Number of times that a connection was “timed” out (connections will time out if a client is inactive for a configurable amount of time.)
`StatRetrievedMessages`	Number of messages retrieved
`StatRetrievedVolume`	Total MIME size of all retrieved messages

8.3.3 Message Store Server Statistics

The Message Store Server (MSS) is responsible for persistent storage of messages.

Event	Description
`StatStartUp`	MSS startup time
`StatConnections`	Total number of current connections
`StatAccumulatedConnections`	Number of total connections accumulated
`StatFailedConnections`	Number of failed connections
`StatStoredMessages`	Number of stored messages. (This value differs from the number of received messages by considering messages that are duplicates of messages already received. Duplicate messages are not saved to the database.)
`StatStoredVolume`	Stored volume (sum of the MIME size of stored messages)
`StatReceivedMessages`	Number of received messages
`StatReceivedVolume`	Received volume of mail (the sum of the MIME size of received messages)

8.3.4 MTA Statistics

The Message Transport Agent (MTA) is responsible for delivering incoming and outgoing mail.

Event	Description
`StatStartUp`	MTA startup time
`StatConnections`	Total number of current connections
`StatAccumulatedConnections`	Number of total connections accumulated
`StatDeferredMessages`	Number of deferred messages
`StatProcessedMessages`	Number of processed messages
`StatReceivedMessages`	Number of received messages
`StatReceivedVolume`	Received volume of mail (the sum of the messages' sizes)
`StatReceivedRecipients`	Number of delivered messages
`StatMemoryUsed`	The amount of memory used

8.3.5 Queue Server Statistics

The Queue Server is responsible for the temporary storage of messages that cannot be processed immediately by the MTA.

Event	Description
`StatStartUp`	Queue Server startup time
`StatConnections`	Total number of current connections
`StatFailedConnections`	Number of failed connections
`StatAccumulatedConnections`	Number of total connections accumulated
`StatLocks`	Total number of locks
`StatAccumulatedLocks`	Number of total locks accumulated

8.3.6 Directory Cache Server Statistics

The Directory Cache Server responds to requests for account information. It has access to the Integrated Services Directory, but services requests from its cache, which contains a local copy of the required account information.

Event	Description
`StatStartUp`	Directory Cache Server startup time
`StatQueriesProcessed`	Total number of queries processed
`StatQueriesSucceeded`	Total number of queries that succeeded
`StatQueriesFailed`	Total number of queries that failed
`StatUnknownUserReturns`	Number of user validation requests that returned UNKNOWN USER
`StatBadPasswordReturns`	Number of validation requests that returned BAD PASSWORD
`StatOKPasswordReturns`	Number of validation requests that returned PASSWORD OK
`StatConnections`	Total number of current connections
`StatFailedConnections`	Number of failed connections (connections fail when an attempt to create a socket for the client fails, typically because the maximum number of connections has been reached.)
`StatAccumulatedConnections`	Number of total connections accumulated
`StatUpdateTime`	Number of seconds spent in processing (`imdircacheserv`) cache updates
`StatRecentUpdates`	Number of updates from the Integrated Services Directory

8.3.7 Configuration Server Statistics

The Configuration Server distributes the latest version of the master configuration database to all the other InterMail hosts.

Event	Description
`StatStartUp`	Configuration Server startup time
`StatQueriesProcessed`	Total number of queries processed
`StatQueriesSucceeded`	Total number of queries that succeeded
`StatQueriesFailed`	Total number of queries that failed
`StatConnections`	Number of current connections
`StatTotalAssessments`	Total number of times the Configuration Server was requested to assess configuration parameters
`StatAssessesFailed`	Number of assessments that failed
`StatTotalInstalls`	Total number of installs
`StatInstallsFailed`	Number of failed installs
`StatAccumulatedConnections`	Number of total connections accumulated
`StatFailedConnections`	Number of failed connections (Connections fail when an attempt to create a socket for the client fails, typically because the maximum number of connections has been reached.)
`StatRejectedConnections`	Number of rejected connections (Connections are rejected if authentication fails.)

8.3.8 IMAP Server Statistics

The IMAP Server handles requests for message retrieval from mail clients that support the IMAP protocol.

Event	Description
`StatConnections`	Number of current IMAP client connections
`StatAccumulatedConnections`	Accumulated number of client connections established during this invocation
`StatFailedConnections`	Number of IMAP client connections that have failed (Connections fail when an attempt to create a socket for the client fails, typically because the maximum number of connections has been reached.)
`StatRejectedConnections`	Number of IMAP client connections that have been rejected (Connections are rejected if authentication fails.)
`StatTimedOutConnections`	Number of IMAP client connections that have timed out (Connections time out if a client is inactive for a configurable amount of time.)
`StatFetchRequests`	Number of fetch commands that have been serviced
`StatCreateRequests`	Number of create commands that have been serviced
`StatDeleteRequests`	Number of delete commands that have been serviced
`StatAppendRequests`	Number of append commands that have been serviced
`StatCopyRequests`	Number of copy commands that have been serviced
`StatSearchRequests`	Number of search commands that have been serviced
`StatFetchedVolume`	Volume of mail fetched by IMAP clients, in kilobytes
`StatAppendedVolume`	Volume of mail appended by IMAP clients, in kilobytes
`StatExpungedMessages`	Number of messages expunged by IMAP clients

8.4 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. This capability is typically turned on at the direction of Software.com technical support.

To set message tracing, use the configuration key `messageTracing`. This can be set independently for any server.

Trace files are referenced in files that are identified by a server name and the file type (for example, `popserv.trace` contains a reference to the trace file for the POP Server).

The level of detail or verbosity that is required in the diagnostic output to the `.trace` file can be set using the configuration key `traceOutputLevel`. For example,

```
/venus/mta/traceOutputLevel: [rme=1,sock=2]
```

The environment variable `IMDIAG` can be used to affect this at program startup.

By default, output to the `.trace` file is not printed to `stderr`. To turn this capability on for any InterMail application, set the value of the configuration key `traceToStderr` to `true`. For example,

```
/*/common/ traceToStderr: [true]
```

If either this key is set to `true`, or the program is launched with a `-traceToConsole` option, the trace output will go to `stderr`.

8.4.1 Trace File Format

If message tracing capability is turned on for a server, it adds an entry to its log file whenever processing a message. These “message tracing” entries allow messages to be traced 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>
```

Example

```
19980511 122236824-0700 venus popserv 2089 5 8;wait=1 entering
waitAll(timeout={5,0}), manager is ef16bb78([-1])
```

8.5 Reading Log Files

Log files can contain hundreds or thousands of lines and while you can view these “raw” log files, InterMail also allows you to parse a log file (or multiple files) to produce more readable format. You can view a more readable log format using the following two commands:

- `imlogsum` (used only for `.log` files)
- `imlogprint(`used for `.log`, `.stat`, and `.trace` files)

Using imlogsum

You can use the imlogsum administrative command only for reading `.log` files.

By issuing the `imlogsum` command with one or more log files specified, you can produce a summary of output that is easy to read and allows you to see useful system information while ignoring less important information.

The `imlogsum` command will report the following information:

- A list of message types that have occurred in this log
- Totals of each type of message
- The number of messages that were handled by the MTA while the file was logging activity in the system

You can specify the output that you want to view by using the `-l <severity>` argument. By using this argument, you report on only those messages that are of a specified severity level.

The illustration that follows displays the result of an `imlogsum` command.

```
venus% /vol1/imail/lib/imlogsum -v mta.venus.199708250000.log
File ‘mta.venus.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-venus-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
************************************************************
```

Figure 19: Sample Log File Output with imlogsum

See the Chapter 12 in the *InterMail Reference Guide* for more information about the `imlogsum` command.

Using imlogprint

You can use the `imlogprint` command to present `.log`, `.stat`, and `.trace` file information in a format that is less terse and more readily understandable. It is usually used after the logs have been pre-filtered to remove unwanted entries. Log entry fields, (specified by a single number, series of numbers, or range of numbers indicating the field(s)) can be included or excluded from the output. The output is printed on one line by default, but can be made multi-line (`-m`), which makes it easier to read. In addition, the fields can be tagged to make the meaning of each field clear.

If you specify a log file that displays local time, then `imlogprint` outputs time in the same timezone as the input file, even if `-l[ocaltime]` is not specified. If `-l` is specified, then it takes the time in the log file and converts it to local time (if it is not already in local time). This is useful if the log file is in GMT or if it is in a timezone other than the current one.

While the default behavior for `imlogprint` is to enter a log entry from standard input, you may use `imlogprint` on an entire log file by directing a named log file into `imlogprint` in the following manner:

```
imlogprint -t -m < mta.log
```

If you choose this option, you will probably want to redirect this output to a second file (e.g., `mta.log.printout`) in the following manner:

```
imlogprint -t -m < mta.log > mta.log.printout
```

Example

In the two examples below, `imlogprint` is invoked from standard input with sample arguments. First the command is issued, then the log entry is entered.

```
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 example `imlogprint` printed multiline reformatted output that indicated the date, time, host, server process, and other information in an easy to read format. It also used 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
```

This example shows how to exclude certain fields; in this case, the first two fields of the log entry (the date and time) are omitted (by using `-x 1-2`) .

See Chapter 12 of the *InterMail Reference Guide* for more information about the `imlogprint` command.

Accessing Log Data via Named Pipes

Logs are archived records of InterMail transactions; however, on some occasions, it is more helpful to see log entries in real-time, as they happen. In InterMail, all events that are reported in log files can also be accessed via a named pipe. The pipe can be directed to standard output and viewed 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 in addition to the file.

The named pipes exist in the same directory as the log files (the directory specified using the `logDir` configuration key) and are identified by the extension `.pipe.<filetype>`; file type being log, stat, or trace.

For example, if you have set the following value for the configuration key `logNamedPipeMode`

```
/*/common/logNamedPipeMode: [1]
```

the log output for the POP Server will be written to `popserv.log` and to the pipe `popserv.pipe.log` both of which reside in the same directory.

9

System Maintenance and Monitoring

The InterMail messaging system is designed to operate 24 hours a day, seven days a week. Like all mission-critical applications, it should be consistently monitored, with individual components adjusted based on performance requirements, observed performance, physical memory usage, virtual memory usage, and network usage. This chapter provides a description of the monitoring tools that can be used to analyze and maintain an optimally performing InterMail system

Also included in this chapter is a discussion of typical system "hotspots" (physical directories that should be regularly observed), and some strategies to fine-tune and expand the InterMail system. The tools described are either provided with InterMail or they are generic OS system tools that can be applied for InterMail-specific tasks.

InterMail system maintenance encompasses the following topics:

- monitoring server availability
- monitoring InterMail via SNMP
- monitoring physical disk usage
- monitoring Oracle
- monitoring system health via imsysmon
- monitoring system performance
- tuning the performance of InterMail
- expanding the Message File System
- adding Directory Cache Servers

Note: *Although every attempt is made in this chapter to provide information that is known about the InterMail system, inevitably maintaining a production InterMail system varies from site to site, based on hardware, network traffic, and what qualifies as acceptable performance.*

9.1 Key Maintenance Concepts

Before reading further in this chapter, it is important to understand several key terms used in relationship to InterMail maintenance:

- *Optimum Capacity*: an InterMail system, like any system, has a certain optimum capacity; that is, acceptable levels of CPU, network, and disk activity. A service running at optimum capacity levels means response times to customers and remote servers are within acceptable ranges. A system's optimum capacity level is pre-determined by hardware components and network bandwidth specified prior to software installation; this capacity level can by optimized by performance "tuning" during the pre-production and production process.
- *Response Time*: response time indicates how fast an external user will see a response to his initial input or connection request. When a system is over its capacity, server response times can be negatively affected and unacceptable to customers (e.g., taking two minutes to POP mail may be unacceptable and could also result in a network timeout).
- *Stress*: when the optimum capacity of InterMail has been exceeded in some fashion, stress is placed on the system. Stress is typically accompanied by system time outs, system lock outs, and unacceptable response times as well as the reduced ability to failover to a backup host because such a failover may cause the backup to have deteriorated performance.

9.2 Monitoring Server Availability

In a production InterMail environment, it is important to ensure that all servers are available and respond in a timely manner. While different levels of precision can be checked when monitoring response times, two quick methods of checking server availability are to telnet to the POP, IMAP, and SMTP ports on the appropriate InterMail hosts (the hosts that are running the POP, IMAP, and MTA servers) or to ping these hosts using `imservping`.

Telnetting to Server Ports

In the following example, a telnet session is started on port 110 (POP port):

```
venus% 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.
```

Figure 20. Telnetting to the POP Server port.

After invoking the telnet command, the POP (or SMTP or IMAP) banner (e.g. "+OK InterMail POP3 server ready") should immediately appear. If the banner does not immediately appear, there may be excessive stress on the POP server.

A telnet session can also be started on port 25 (SMTP port) to check to see if the MTA is available:

```
venus% telnet 0 25
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
220 venus.software.com ESMTP server (InterMail v4.0 205) ready Fri, 30 Jan 1998
14:18:27 -0800
quit
221 venus.software.com ESMTP server closing connection
Connection closed by foreign host.
```

Figure 21. Telnetting to the SMTP port.

As with checking the POP server, when the server does not immediately respond, there may be excessive stress on the MTA 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:

```
venus% imservping 1 5 mta

Fri Feb 28 10:03:18 EST 1997. imservping: (Info) mta responded
```

Figure 22. Pinging the MTA.

In this example, the MTA was pinged with a warning time of 1 second (to begin pinging) and maximum time of 5 seconds (within which to report). All InterMail servers can be checked in a similar fashion.

9.3 Monitoring InterMail via SNMP

Certain events and processes can be monitored via SNMP (the Simple Network Management Protocol). SNMP is a protocol that provides users the ability to monitor useful 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 was started, the total number of messages that are stored in the MTA, and the total number of messages stored in the MSS. This information is "sampled" and sent to output on the user-defined SNMP monitoring station.

This information is particularly useful in InterMail because it can be viewed real-time without any script development or parsing of logs, can be viewed with standard SNMP monitoring stations (e.g., HP OpenView), and provides information about the present state of a server (volumes or numbers of connections at a given time) as well as archived information (accumulated volumes or numbers of connections over a time period).

9.3.1 SNMP Support for InterMail

Each InterMail installation includes a single SNMP server. Each InterMail server that can be monitored via SNMP (the POP, IMAP, MTA, MSS, Directory Cache Server, and Queue Server) contains a client that "speaks" SNMP. An SNMP thread resides on each SNMP-monitored server and responds to requests from the SNMP Server. When requests are not pending, this thread dispatches whatever traps are pending. Any of the SNMP-enabled InterMail servers may raise a trap (a notification) through the SNMP Server. The SNMP Server listens for requests from SNMP management stations. By default, the SNMP Server takes requests on port 161 and transmit traps on port 162. These ports are configurable (via the `snmpRequestPort` and `snmpTrapPort` configuration keys); however, they are SNMP-standard ports defined by most UNIX operating systems and typically should not be changed.

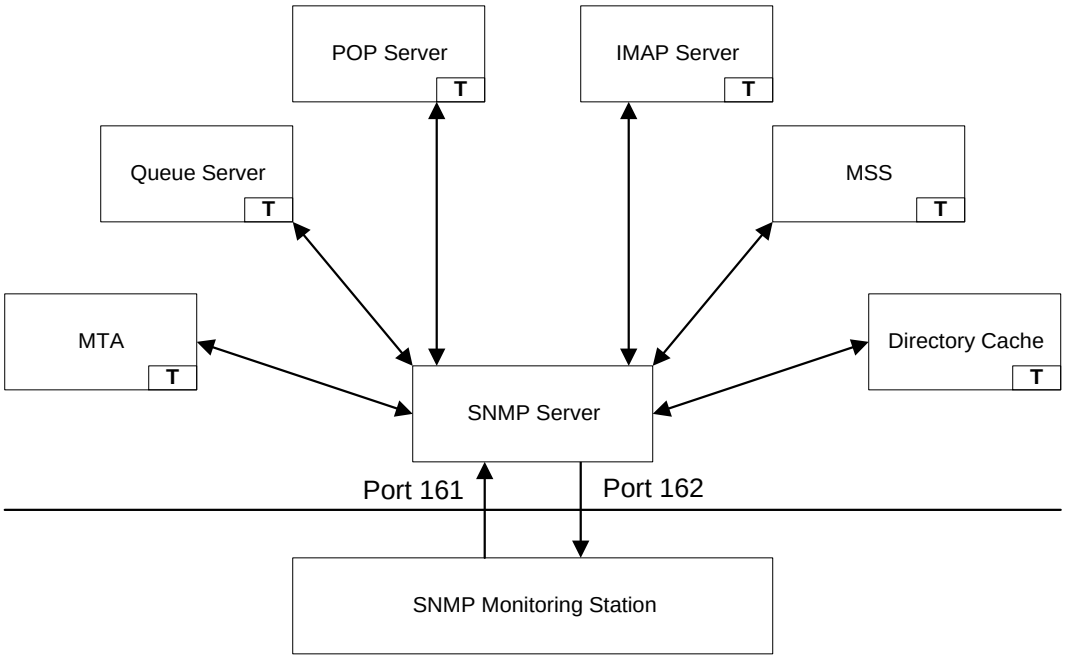

Figure 23. SNMP Client-Server Relationship in InterMail.

Figure 23 shows the relationship between an SNMP Monitoring station and the SNMP Server. Requests made from the SNMP monitoring station to the SNMP server are forwarded to the individual SNMP-enabled servers. The response to this request returns to the monitoring station via the SNMP Server. Communication between the SNMP server and monitoring stations is two-way; information can be initiated from the monitoring station and "pulled" from the InterMail sub-agents; or, the SNMP-enabled InterMail servers can "push" information to the monitoring station (via the SNMP Server). Pushed information (*traps*) is classified according to the level of severity. Pulled information (*polls*) is classified by the type of statistic that is reported.

Note: *In InterMail, SNMP traps correspond to the same events that are logged in InterMail log files; SNMP polls correspond to the information reported in InterMail stat files. SMTP traps can be filtered according to their severity level.*

For more information on SNMP and for a complete list of reportable parameters available for SNMP Monitoring, please see Chapter 10 of the *InterMail Reference Guide*.

9.3.2 Configuring an SNMP Server and Monitoring Station

In InterMail, any remote host can be used to view SNMP information. For example, although InterMail runs on UNIX platforms, an SNMP monitoring station can be run on a Windows NT machines that receives InterMail information via SNMP.

The process of configuring SNMP to run on a monitoring station involves several steps on the SNMP server side (the InterMail host running the SNMP server) and the SNMP monitoring station side (the machine running an SNMP monitoring station).

SNMP Server Configuration

When configuring the InterMail SNMP Server for operation, the following keys should be adjusted as necessary:

- The `/*/common/trapMask` configuration key, which lists the severity level of events that are reported by SNMP. Valid values for `trapMask` are "notification," "warning," "error," "urgent," and "fatal."

Note: *As with all configuration keys, the common/trapMask configuration key can be set on a server-by-server basis, by adding a configuration key for each server, e.g. mta/trapMask By setting severity levels and specifying them for all server, the amount of unwanted information displayed in an SNMP monitoring station can be reduced.*

- The `/*/snmp/masterAgentHost` configuration key, which specifies the name of the host which is running the SNMP Master agent.

The SNMP thread in each server maintains a queue of events (traps) that are dispatched periodically. This activity takes place when the thread is not busy processing requests. The following configuration keys are related to the queuing and dispatching activities of the SNMP thread:

- The `/*/snmp/trapQueueSize` key, which specifies the number of events tracked and stored in the queue. After the value specified by this key is reached, new events will be added to the "bottom" of the queue and events that are at the "top" of the queue will no longer be tracked. The default size is 1024 (this occupies only 1Kb of memory for the queue itself). If the queue were to fill up, some traps may not actually make it to the monitoring station but the event would still be logged to the appropriate server log. Valid values range from 512 to 4096.
- The `snmp/eventMaxWait` configuration key, which specifies the maximum amount of time (in milliseconds) that the SNMP thread will block waiting for a request from a management station can be established. Remember, whenever this thread is not servicing requests, it is actually dispatching queued events (if any). The default used is 3 seconds. And the servers will allow values between 3 and 10 seconds.

An example of the configuration keys used to configure the SNMP server is shown in Figure 24:

```
.....
/*/common/trapMask:
    [fatal]
    [urgent]
/*/common/trapQueueSize: [1024]
/*/snmpdm/eventMaxWait: [5000]
......
/venus/snmpdm/masterAgentHost: [venus]
......
```

Figure 24. SNMP Server Configuration Keys

SNMP Monitoring Station Configuration

The process of configuring an SNMP Monitoring Station varies based on the particular Monitoring software chosen; however, the following generic steps should be followed:

1. Find the Ovtb directory for the SNMP Monitoring Station.
2. Create a directory under the Ovtb directory (e.g. “swcom”).
3. Establish an FTP connection to the InterMail host running the SNMP server. Find the directory that contains the MIB (Management Information Base) files (these files will be named `*.my`). Copy these files to the Ovtb/swcom directory.
4. Once the files are copied to the machine running the SNMP Monitoring Station, compile these files into `*.mib` files. Refer to the documentation provided with the SNMP Monitoring Station software.
5. Identify the host that is running the InterMail SNMP Server to the SNMP Monitoring Station.

9.4 Monitoring 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 Integrated Services Directory, contain information that should be checked on a regular basis. The devices or mount points where these databases are located should be checked regularly. Also, the InterMail Message File System (`/msgfiles`) and other parts of the InterMail file system which are not server-specific (i.e., the `/journal` and `/logs` directory and the Oracle redo logs) tend to grow in disk space and without an effective archiving and backup strategy in place, large amounts of hard drive space can be consumed. The directories where disk “growth” is likely to create problems are the `/logs`, `/spool`, and `/journal` directories. In some cases (depending upon if and how a sidelining policy has been implemented), the growth of the sidelined directory, located at `/queue/sidelined` (or `/spool/deferred` in rare occasions) can also become problematic.

Using `df` or a similar utility, the amount of space *used* in a system, by mount point, can be observed. *Total disk space*, by mount point, can be viewed with the `df` command and file usage can be checked with the `du` command.

After determining the relevant mount point, run `du` on the directories at that mount point, and determine which file sizes are large enough to warrant inspection. Typically, files that will cause a drain on disk space and the strategy for dealing with these files are as follows:

Type of File	Action
Journal Files (`/journal` directory)	Determine or revisit the backup strategy for old journal files.
Log Files (`/log` directory)	Set log rollover to occur more frequently; physically move old log files to a separate file system or media source.
Message Files (`/msgfiles` directory)	If space becomes an issue for Message files, allocate more space; message files are deleted when no longer used through garbage collection and Message Aging policies.
Deferred mail and undeliverable (`/queue` directory)	Fix any mail that can not be delivered and reprocess this mail.
Deferred mail (`/spool` directory)	Fix any mail that can not be delivered and reprocess this mail.
Oracle Redo logs	Similar to the journal file strategy—remove expired redo logs and place them onto backup medium.
Sidelined Mail (`/queue/sidelined` or `/spool/sidelined`)	Manually inspect messages and reprocess using the immsgprocess command. See Chapter 4 in the *InterMail Reference Manual* for more details on Sidelining Messages.

9.5 Monitoring Oracle

InterMail employs Oracle databases in the Message Store database and Integrated Services Directory. These databases require regular monitoring and maintenance in order to function at peak capacity.

The following sections show how to perform some routine monitoring functions, as well as how to check for problems that can adversely affect performance of InterMail, such as fragmented Oracle indexes or lack of available disk space for tablespaces. For these problems, InterMail provides tools to reorganize indexes and allocate additional space for tablespaces.

9.5.1 Indexes and Tables

There are two types of "objects" in any InterMail database—tables and indexes. It is important to understand these objects in order to perform monitoring and maintenance functions in InterMail.

- A *table* contains zero or more rows. Each row is made up of a number of columns. Each column is of a specific data type.
- An *index* refers to one or more columns inside a table. Each row in the table has a corresponding entry in the index. An index entry for a row consists of a copy of the values of the indexed column(s) in the row, plus a pointer to the row, called a RowID.

Indexes are directly related to tables and are used to optimize the querying process. Rows in tables are completely unordered; in order to find a row with a particular value in a given column without an index, the table would have to be scanned row by row starting from the beginning of the table. Indexes, on the other hand, are organized as balanced trees; the relevant index entry for a particular column value can be fetched relatively quickly. The RowID in the index entry will point to the desired row in the indexed table. The relationship of indexes to tables is shown in Figure 25.

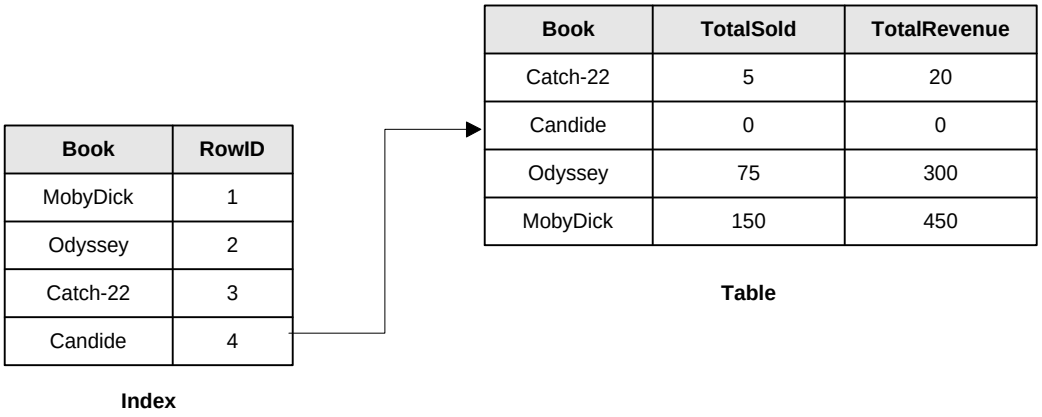

Figure 25. Indexes and Tables

Fragmented Indexes

In the course of InterMail operations, various creation and deletion processes such as adding and deleting messages will affect indexes. Over time, an index will develop "holes" from deleted entries and the database will then consist of more and more space formerly occupied by entries for updated or deleted rows. This space hurts InterMail performance for three reasons:

- extra space is wasted in the database storing the "holes"
- extra time is used to read and write these "holes" to disk, and
- extra cache space is wasted because the "holes" are also stored.

For example, if "Odyssey" were updated to "Illiad" in the table, the entry for the "Odyssey" in the index would be "dropped" and no longer exist as an entry in the index. However the space "Odyssey" occupied in the index would not be reclaimed and would form a "hole" in the index. Because Oracle does not efficiently reuse the space vacated in an index when rows in tables are deleted or updated, fragmentation results.

Note: *Tables do not suffer from this problem; the empty space created when a row is deleted will subsequently be used when a new row is inserted.*

With a fragmented index, searches take longer. When Oracle needs an index entry from the disk, it doesn't just read in the single index entry, it reads in the entire block of the index containing the desired index entry. For example, if the database needs to look up the “Candide” row in the table above, it reads in the entire block from the index that contains the entry for “Candide.” In our example, this block contains other index entries, as well as a “hole” for the now departed “Odyssey.”

When Oracle reads in an index block from the disk, it stores it in an in-memory cache. If some subsequent query needs an entry and that entry is present in the same block, Oracle avoids having to do a disk read, and can use the index entry straight from the in-memory cache.

The cache stores a fixed number of blocks (with the actual number based on the size of the cache). In order to make room for a new block when the cache is full, Oracle flushes the least recently used block from the cache.

The less fragmented the index, the more index entries per block, which implies more index entries per block stored in the in-memory cache, and a greater chance that a needed entry will be in the cache and not require a disk read to fetch. Therefore the less fragmented the index, the greater the performance.

9.5.2 Reorganizing Database Indexes

Index reorganization improves performance by eliminating the fragmented free space that builds up in Oracle indexes over time. As mentioned previously, eliminating index fragmentation also increases the performance of the database. The `imdbindexreorg` accomplishes index reorganization.

To run `imdbindexreorg`, use the following example as a guide.

```
imdbindexreorg -tablespaces /tmp/table.backup -bloat 70
```

In this example, the command is run and produces an output file to `/tmp/table.backup` (or any other user-defined file). This file lists those tablespaces that have been affected by the index reorganization and therefore require a hot backup.

The command must be run on the host where the Message Store database runs. The `-bloat <percentage>` switch allows the user to fine-tune the sensitivity of an index reorganization by setting a high-water mark for the percentage of index space used. When this high-water mark is reached, the index is considered bloated and `imdbindexreorg` should be run in order to reorganize the indexes to an acceptable percentage of the total available space.

Warning! The `imdbindexreorg` command must be run during a maintenance window when all services that attempt to modify the database have been shut down.

When an index has a bloat percentage greater than the specified bloat size, a reorganization will be performed on that index. In this way, a critically bloated index can be reorganized and the other indexes can be left alone.

Next, `imdbindexreorg` reads the value of `db/indexReorganizationTimeLimitMinutes`, which specifies the amount of time (in minutes) that the `imdbindexreorg` command is allowed to run.

If the `db/indexReorganizationTimeLimitMinutes` configuration key is set too high, index reorganization may take longer than the time defined by a maintenance window. If this key is set too low and the `imdbindexreorg` command run too infrequently, performance will be hurt and space wasted because the command will not have enough time to defragment the indexes. This key can be overridden by specifying the `-timelimit` minutes argument.

For more details about `imdbindexreorg`, please see Chapter 12 of *the InterMail Reference Guide*.

9.5.3 Monitoring Oracle Tablespaces

Physical space in a database is partitioned into tablespaces. Each tablespace is composed of one or more files. All the storage for a particular index or table resides in one particular tablespace, although the storage might be split up amongst the several files that compose the tablespace.

Figure 26 represents the total space in a database. The space is separated into five tablespaces: SYSTEM, TEMP, RBS, POX01, and POD01.

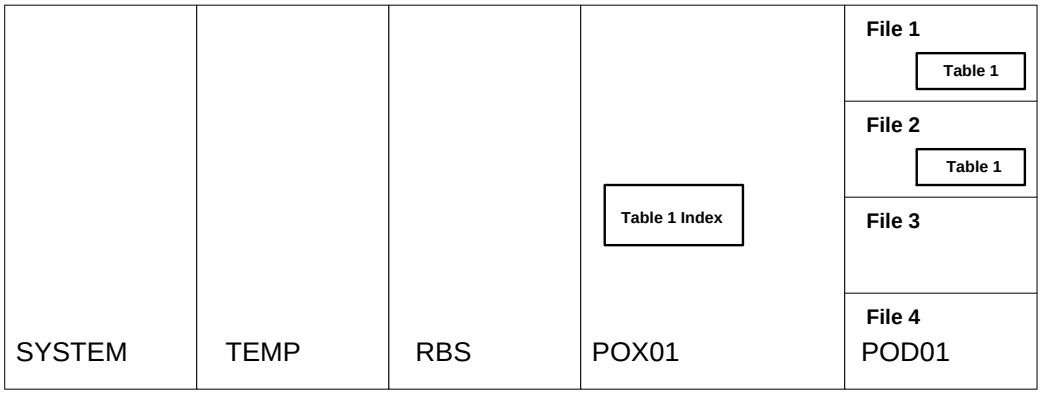

Figure 26. Tablespaces in a database

If we look at one logical partition of an Oracle database in InterMail, we can clearly see what sort of difficulties arise in the management of available space. When a table is first created in an InterMail Message Store database, a certain fixed amount of space is allocated for the storage of its rows.

This space is called the table's initial extent. When this space is exhausted, an additional increment of space called an extent is allocated. An extent is a multiple of the Oracle block size, so, for example, if the database is constructed with 1k blocks, table extents (as well as indexes) are created in multiples of 10 k.[7] Every table has a particular preassigned extent size.

[7] In InterMail, the Oracle block size is typically 8 k.

For example, if more space is needed for a given table that has an extent size of 80 k, an additional 80 k extent will be added. In order to add this additional extent, Oracle will scan the tablespace until it finds a free extent of contiguous blocks (unallocated space in the tablespace) of 80 k or greater. The space for the new extent does not need to be contiguous to any of the table's existing extents, and can be added anywhere in the tablespace. To further illustrate the way the Message Store database allocates space, observe Figure 27.

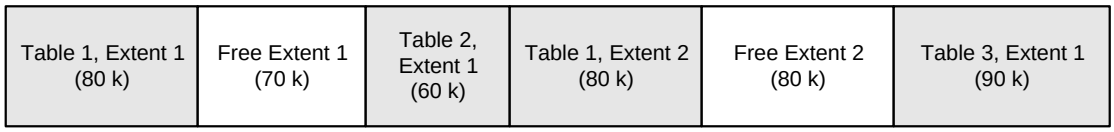

Figure 27. Space Allocation in a Tablespace

In this example, a tablespace exists with three tables and two extents of available space. Remember, as a rule, extents are only added in available free extents and these free extents must contain enough blocks to accommodate an additional extent. So, if Table 1 which has two 80 k extents (Table 1, Extent 1 and Table 1, Extent 2) needs to add an additional extent, it will search the tablespace for a free extent of sufficient size (80 k). After discovering that Free Extent 2 contains 80 k, it will allocate that space, creating a third extent (Table 1, Extent 3).

This works well while there is a free extent to accommodate an additional extent, but when there is no available space, problems can occur. While Table 2 can add an extent (a 60 k portion of the 80 k block in Free Extent 2), if Table 3 needs to add an extent, it will search the tablespace for a 90 k free extent and finds that only a 70 k free extent exists in Free Extent 1; therefore, the table cannot add an extent. In this case, the tablespace needs to be expanded.

Another potential problem exists. Since database tables grow at different rates, it is possible that Table 2 may grow before Table 1 grows. Suppose Table 2 was the first to require a new extent. Table 2 will search the database and discover that the next available extent is Free Extent 2 and it will allocate 60 k of this 80 k block.[8] This creates a problem for Table 1, because now there is only a 70 k block (Free Extent 1) and a 20 k block (the remainder of Free Extent 2) remaining. Again, the tablespace will need to be expanded.

The challenge in managing space in databases is to determine when a tablespace needs to be expanded (by growing the size). In order to monitor the growth of Oracle databases, InterMail provides two tools to check on database usage and specifically estimate when the available free space will need to be expanded: `imdbspacecheck` and `imdbspacequickcheck`.

9.5.4 Checking Free Space in Oracle

The `imdbspacecheck` command monitors the remaining free space in an Oracle database, and estimates when more space will be required. It warns when available space is at a point where it is advisable to add more. It calculates this by checking all possible permutations in the Message Store database and also the historical growth of tables, which it will write to a file called `imdbspacecheck.<DB_instance>.hst`. Each time the `imdbspacecheck` is executed, the `imdbspacecheck.<DB_instance>.hst` file is updated. In addition, `imdbspacecheck` can be run while InterMail is in full operation and is intended to be run regularly.

When executed, the command will log an Urgent message when available space is at the point where more should be added. Two conditions will cause `imdbspacecheck` to log an "Urgent" message:

- when there is an object (table or index) that contains an extent that is equal to or greater than the percentage set by the `db/extentFullWarningThresholdPercent` configuration key. This would indicate the possibility that this object would not be able to add an extent.
- if there is an object whose last extent is predicted to fill within the time specified by the `db/extentGrowthAllowanceDays` configuration key and there is a possibility that the object will not be able to add an extent.

To run `imdbspacecheck`, use the following example as a guide:

```
imdbspacecheck -thresholdpercent 70 -thresholddays 60 -dir MSS
```

In this example, the command is executed with very conservative parameters. It will report back based on any combination of database growth that will prohibit any extent that is currently at 70% (set by `-thresholdpercent`) to add an additional extent within 60 days (set by `-thresholddays`). If these parameters are used in the `imdbspacecheck` invocation, they will replace the values specified in the `db/extentFullWarningThresholdPercent` and `db/extentGrowthAllowanceDays` configuration keys for that invocation.

While `imdbspacecheck` is memory-intensive and can take a long time to run, InterMail provides a second command, `imdbspacequickcheck`, to quickly check free space in a database. The `imdbspacecheck` command checks every possible combination of table growth, calculating all permutations and referring to stored historical data, to determine when additional space will be needed. `imdbspacequickcheck` performs a much more cursory check, only checking on current conditions instead of providing an in-depth analysis of future projections, as `imdbspacecheck` does.

The `imdbspacequickcheck` command searches the database to determine the result if any of the existing tables in the database need an additional extent in isolation. If any of the existing tables will have a problem growing, `imdbspacequickcheck` will report this information. This differs from `imdbspacecheck`, which projects all the growth of all extents in the future, based on historical data and database growth patterns. Basically, `imdbspacequickcheck` can warn of an impending space crisis and serves as a last-chance safety net to catch space problems before a crisis erupts.

Like `imdbspacecheck`, `imdbspacequickcheck` logs an "Urgent" message if there is an object (table or index) whose last extent is over a configurable high-water mark (system default setting is 90% full), and the object cannot grow. "Static" objects, that is table and indexes known not to grow, are exempt from this check.

In typical system operation, `imdbspacequickcheck` should be scheduled as a frequently run cron job, executed approximately once every thirty minutes. Running `imdbspacequickcheck` imposes an insignificant load on the monitored system. Therefore it can be run much more frequently than `imdbspacecheck`.

---

[8] The database uses a "first fit" algorithm when allocating a new extent for a table. What this means is that, when searching for available space, the next available block—not the first block in the entire tablespace—that can accommodate an extent for a given Table is used regardless of the growth needs of other Tables.

To run `imdbspacequickcheck`, use the following example:

```
imdbspacequickcheck -thresholdpercent 70
```

As in the example for `imdbspacecheck`, a conservative strategy is followed where `imdbspacequickcheck` overrides the `db/extentFullWarningThresholdPercent` configuration key with `-thresholdpercent 70`.

9.5.5 Expanding Oracle Tablespaces

Once it has been determined that a particular database needs to grow and there is available hard drive space to accommodate this increase, run the `imdbspacegrow` command to expand the available tablespace. The log entries created by `imdbspacecheck` and `imdbspacequickcheck` give warning of an impeding space crisis, determining when `imdbspacegrow` should be run.

The following example illustrates how to run `imdbspacecheck`:

```
imdbspacegrow -dryrun
```

In this example, `imdbspacegrow` is executed with the `-dryrun` argument. When this argument is set, the command will not actually perform the database expansion and instead logs a description of what *would* happen if the database was enlarged. It is generally a good idea to perform a dry run of `imdbspacegrow` and review the results before actually adding the space.

After the dryrun of `imdbspacegrow`, status information is displayed to standard output:

```
Database name:       IMM1
Database access:     local
ORACLE_HOME:         /disk2/oracle/7.3.3
Ready to proceed?    (Y/N) [yes]
```

Figure 28. Setting Environment Variables for imdbspacegrow.

If the information presented is correct, type yes and press enter; if not, type no and press enter.

Once status information is confirmed, a list of tablespace names is displayed, and the name of the tablespace to be expanded can be chosen. The following is an example of the output that can be displayed; however, the exact display will differ depending upon the actual names of tablespaces in the system.

```
The database is partitioned into 5 tablespaces.
TABLESPACE LIST
1) SYSTEM
2) TEMP
3) TB1
4) TB2
5) TB3
Please select a tablespace from the list by number: [5]
You selected tablespace TB3.
After you make a selection, InterMail will report the current status of the tablespace
that you chose.
For your information, the tablespace TB3 is currently composed of the following files:

FILE NAME                                    SIZE IN BYTES
/disk2/oracle/admin/D1ALE733/TB3.dbf         32522240 bytes
Ready to continue? (Y/N) [yes]
```

Figure 29. Specifying a Tablespace to Expand.

The command prints information about the recommended sizing for the additional file and then prompts the user to specify the size of the new tablespace file:

```
Enter the size of the file to be added to the tablespace: [0] 32522240
------------------------------------------------------------
```

Figure 30. Choosing the Size of a New Tablespace File.

Next, the full pathname for the new file is entered. Typically, the .dbf extension is used for Oracle database files.

```
Enter full path name of new file:
/disk2/oracle/admin/D1ALE733/TB3_2.dbf
A file named /disk2/oracle/admin/D1ALE733/TB3_2.dbf of size 32522240 bytes will be
added to tablespace TB3.

Proceed? (Y/N) [yes]
------------------------------------------------------------
```

Figure 31. Confirming imdbspacegrow Operation.

At this point, InterMail will typically add the new file to the tablespace TB3. In this example, since `-dryrun` was specified, the results of the operation are displayed, but the file will not be written to disk.

```
Oracle added a new file named "/disk2/oracle/admin/D1ALE733/TB_3.dbf" to tablespace
"TB3". The actual size of the file is 0 bytes.

NOTE: this differs from the size you requested (32522240).

The tablespace TB3 should be hot backed up now, since its structure has been altered
in a significant way.
```

Figure 32. Completing the imdbspacegrow Process.

While `imdbspacegrow` does not interfere with the normal operation of InterMail, it is highly suggested that InterMail be backed up (performing a hot backup) immediately after creating more tablespace. Therefore, it is advisable to run `imdbspacegrow` during a maintenance window or other non-peak time.

9.5.6 Viewing Database Information

To see an overview of all database information, use the `imdbspacereport` utility, which prints out information about the database, including the exact Oracle version used, the value of all database parameters, and information about space usage in the database.[9]

To run `imdbspacereport`, type the following:

```
imdbspacereport -dir MSS
```

where `-dir MSS` indicates the name of the Message Store database instance for which to produce a report.

[9] In addition to providing an overview of all database information, when `imdbspacecheck` or `imdbspacequickcheck` warn of low tablespace, `imdbspacereport` can be run to obtain a snapshot of space allocation in the database in order to better understand the depth of the impending crisis.

9.6 Monitoring System Health with imsysmon

InterMail servers can be monitored by telnetting to ports and via pinging; however, to provide time-sensitive monitoring of servers, the `imsysmon` command can be employed. The `imsysmon` command checks the overall health of the InterMail system and reports status to a log file. When `imsysmon` discovers events of a user-definable severity level (see Chapter 8 for more information on severity levels), it will optionally send an e-mail and/or page to a list of specified addresses. In this way, real-time monitoring can be accomplished 24x7.

9.6.1 The imsysmon.ini file

Before running the `imsysmon` command, `imsysmon` should be configured to determine where system files are located, which servers will be monitored, where log files will be written, and who will receive e-mail and/or pager alerts. The `imsysmon` command is configured by modifying the `imsysmon.ini` file.

The `imsysmon.ini` file contains four sections:

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

The following four sections describe the variables and meaning of values found in the `imsysmon.ini` file:

IMSYSMON: Setting the Running Environment for imsysmon

The IMSYSMON section of the `imsysmon.ini` file controls how `imsysmon` is expected to 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`	This variable set the number of seconds to wait for a banner to appear.
`MailHosts`	A list of mail hosts that can be used to deliver e-mail and pager notifications. It is important to specify at least one mail host that can be used but is not contained inside your InterMail system in the event that InterMail MTAs are not available.
`MailRcptTo`	The email address(-es) that will receive a message when a Fatal or Urgent event occurs.
`MaxConnAttemptNote`	The maximum number of attempts in connecting to a port.
`MinErrDiskUsage`	The minimum amount of disk usage before reporting an “Error” event.
`MinFatlDiskUsage`	The minimum amount of disk usage before reporting a Fatal event.
`MinUrgtDiskUsage`	The minimum amount of disk usage before reporting an Urgent event.
`MaxErroFreeExtents`	The minimum percentage of free extents available before reporting an Error event.
`MaxFatlFreeExtents`	The minimum percentage of free extents available before reporting a Fatal event.
`MaxUrgtFreeExtents`	The minimum percentage of free extents available before reporting an Urgent event.
`PageRcptTo`	The e-mail address(-es) that will receive a message when an Urgent event occurs.
`TimeOut`	The period of 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 imsysmon’s interaction with InterMail. The variables in the INTERMAIL section and their meaning are as follows:

`UserName`	The name of the InterMail user. This will typically be set to the same value as the `common/commonUser` configuration key.

ORACLE: Setting the Running Environment for Oracle

The ORACLE section of the `imsysmon.ini` file consists of one variable used to set the Oracle instances that the `imsysmon` command will monitor.

`OracleSID`	The list of Oracle instances that will be monitored. Typically, there will be at least two instances specified—one for the Message Store database and one for the Integrated Services Directory.

DEFAULT: Setting the Threshold Limits for imsysmon

The DEFAULT section of the `imsysmon.ini` file determined which events will be monitored and what severity level should `imsysmon` consider a particular error when it reports it to a log file, standard error, and/or sends a mail or page (if the event is Urgent or Fatal severity level). Whereas the IMSYSMON section sets thresholds in percentages or numbers, the DEFAULT sections sets the severity level that `imsysmon` will report when the thresholded percentage or number is reached.

The following values are used to 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, the user can set a value to reflect which severity level will be reported. If the user does not wish to have a particular event monitored and reported by `imsysmon`, the event can be "commented out" (see section 9.6.2 for examples on modifying the `imsysmon.ini` file).

`FatlDiskSpace`	The percentage of disk usage is at least the value specified by `MinFatlDiskUsage`.
`FatlFreeExtents`	The number of free extents is less than the value specified by `MaxFatlFreeExtents`.
`OraInstPingFail`	An Oracle instance (as defined by the `OracleSID` variable) is not responding.
`ProcNotFound`	An InterMail server is not responding to a ping.
`FatalMsgsFoundInLog`	Fatal messages found in server logs.
`AlarmTimeout`	A timeout has occurred while waiting for a task.
`ArchivePathNotFound`	The Oracle archive path for a specific instance was not found.
`DiskSpaceCheckFail`	An error has occurred while trying to perform a disk space check.
`LogFileOpenFail`	The `imsysmon` log file could not be opened.

`UrgtDiskSpace`	The percentage of disk usage is at least equal to `MinUrgtDiskUsage`.
`OraTabOpenFail`	The oratab file cannot be opened or read.
`PortBannerFail`	A failure to connect to server port expecting a banner has occurred.
`UrgtFreeExtents`	The number of free Oracle extents is less than or equal to `MaxUrgtFreeExtents`.
`UrgtMsgsFoundInLog`	Urgent messages found in server logs.
`CheckFreeExtentsFail`	There has been a failure to check for free Oracle extents.
`ErroFreeExtents`	The segment for an Oracle instance is low.
`IMAPQuitError`	The IMAP server rejected a “QUIT” command.
`IMDbSpaceChecker`	There has been a failure to run `imdbspacechecker`.
`IMDbLogParsingFailure`	There has been a failure to parse an InterMail log file.
`IMServPingFail`	There has been a failure to run `imservping`.
`LogSegmentCreationFailure`	There has been a failure to create a log segment.
`LogSegmentReadFailure`	There has been a failure to read a previously saved Log segment.
`POPQuitError`	The POP server rejected a “QUIT” command.
`SendMailAttemptFail`	There has been a failure to notify an event via sendmail.
`SMTPDataError`	SMTP server rejected our “DATA” address.
`SMTPFromError`	SMTP server rejected our “MAIL FROM” address.
`SMTPRcptToError`	SMTP server rejected our “RCPT TO” address.
`SMTPMessageBodyError`	SMTP server rejected the body of our message.
`SMTPNotifyFail`	Failure to notify of event via e-mail.
`SMTPQuitError`	SMTP server rejected our “QUIT” command.
`SocketSMTPMailFail`	Failure to notify of event via e-mail.
`SockOpenFail`	Unable to create a socket to communicate with one of the servers.
`UnhdlSignal`	A signal that has not been accounted for has 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.
`OraSidNotFound`	Oracle instance (`Oracle_SID`) not found.

`OraHomeNotFound`	Oracle home directory for specific instance not found.
`ProcFound`	An information message that a process is found.
`PortBanner`	An information message that a banner was found for a server port.

9.6.2 Modifying the imsysmon.ini file

As shown in section 9.6.1, 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]
......
......

# 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
venus.software.com
earth.software.com
accordance.com
EOT

# MailRcptTo - The email address(-es) that will receive a message when
#              an urgent event happens.
MailRcptTo = <<EOT
administrator@software.com
tech_support@software.com
EOT

# MinErrDiskUsage - The minimum amount of disk usage before raising
#                   an "error" message.
MinErrDiskUsage = 80
......
......

[INTERMAIL]

# UserName - The name the the InterMail user.
UserName = imail

[ORACLE]

# OracleSID - The list of Oracle instances that will be monitored.
OracleSID = <<EOT
IMM1
IMD1
EOT

[DEFAULT]

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

Figure 33. Sample imsysmon.ini File.

9.6.3 Running imsysmon

After the `imsysmon.ini` file has been configured, the `imsysmon` command can be invoked on a command-line by typing `imsysmon`. However, in order to run `imsysmon`, you must be root.

Note: *For a complete description of available syntax for imsysmon, refer to Chapter 12 of the InterMail Reference Manual.*

The following example shows the execution of `imsysmon` and some possible problems that may be reported.

```
venus% ./imsysmon
imsysmon:  Monitor procedure started for host venus on Fri May 08 14:46:15 PDT 1998
           Initializing......

imsysmon:  Initialization complete. Running with these values

        HostName               venus
        OpSys                  SunOS
        ImailUName             imail
        OracleUName            oracle
        InterMailDir           /vol1/imail
        VerboseMode            0
        LogMode                1
        LogFile                /vol1/imail/log/imsysmon.venus.19980508144615.log
        MailDeliveryHosts      earth:7131
        WaitTime               2
        Environment: imail
        Environment: oracle
imsysmon:  monitoring InterMail modules:  imdircacheserv mss mta popserv
--
Checking module imdircacheserv...
--
imsysmon:  FATAL!  19980508144617:  imdircacheserv does NOT respond to connection
requests.
--
Checking module mss...
--
imsysmon:  FATAL!  19980508144624:  mss does NOT respond to connection requests.
--
Checking module mta...
--
imsysmon:  FATAL!  19980508144628:  mta does NOT respond to connection requests.
--
Checking module popserv...
--
imsysmon:  FATAL!  19980508144633:  popserv does NOT respond to connection requests.
imsysmon:  Monitor procedure ended for host venus on Fri May 08 14:46:35 PDT 1998

venus%
```

Figure 34. Running imsysmon.

Based on the output shown in Figure 34, a log file will be created (`imsysmon.venus.19980508144615.log`) and an e-mail will be sent to the addresses specified in the `MailRcptTo` variable.

9.7 Monitoring System Performance

During the course of InterMail operations, certain system resources should be checked to make sure that the system is not being overloaded. The areas that typically require monitoring are as follows:

- RAM Usage
- Throughput Speed and Volume
- Disk Performance
- Timeouts and Dropped Connections
- Networked Resources

The sections that follow describe each of these performance areas and suggest a tool that can be used for monitoring. The actual tool used will depend on the UNIX platform on which the InterMail installation is running.

Note: *The following examples and suggestions are meant only as guidelines; specific hardware configuration, system usage, additional third-party software, and network connectivity can greatly affect an InterMail system and must be taken into account when estimating system performance for a specific site.*

9.7.1 Checking RAM 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, the amount of RAM devoted as swap space can be monitored in a more objective manner and should follow these basic guidelines. Use `vmstat` or an equivalent utility on a regular basis to determine available swap space. An example of using `vmstat` to determine RAM usage follows:

```
venus% 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
```

Figure 35. Checking RAM Usage.

The important numbers to track are the swap space used and the available free swap space (swap-17504 and free-11760, in this example). In this example, 60% of available swap space is used (`swap` column / (`swap` + `free` columns)) and this is considered “green” condition in which operation is within reasonable parameters. If the amount of total used swap space was at 75%, this would be considered a “yellow” condition that should be closely observed and, if it continued, would indicate a possible problem. If this percentage were at 95%, this would be a “red” condition and definitely indicates a problem.

Warning! The swap partition should be analyzed to determine if it is large enough to handle the transaction volume. If not, more space may need to be added to the swap partition, available virtual memory may need to be expanded, or the number of available connections and processes in InterMail may need to be decreased.

9.7.2 Checking Throughput Speed and Volume

It is important for an InterMail system to be within an acceptable range of throughput times for the amount of transactions through the system.

```
venus% 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
```

Figure 36. Checking Throughput Volume.

With this output, the number of reads per second (`r/s` column), writes per second (`w/s` column), and average service time can be observed. These numbers will vary; however, as a general rule, if the service time is high and/or the reads or writes per second is low, then problems may exist in the system.

9.7.3 Checking Disk Performance

Another potential problem area is disk performance, specifically access times to the disk in question. For instance, if it takes too long for a connection to the MSS to access the hard disk, problems may ensue.

Access Times

To check access times for a given host, issue a `sar` command. The following output will be produced.

```
venus% sar -d -o /var/tmp/disk.sar 10 100000

SunOS venus 5.5.1 Generic_103640-12 sun4u    12/03/97

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
```

Figure 37. Checking Access Times.

With this output, problems or bottlenecks can be observed on a particular hard drive and CPU performance can be determined based on the average number of requests that are outstanding (queued) and the average time to process a request. Depending upon how a disk array is configured, the `avwait` column can provide useful data and, in general, a high `avque` time indicates problems and possible file contention where the system is exhibiting poor disk performance.

CPU Load

To analyze the actual CPU load, issue the following command:

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

SunOS venus 5.5.1 Generic_103640-06 sun4u    12/03/97

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
```

Figure 38. Checking CPU Load.

Generally, load should be calculated from system (kernel) processes and user processes and this behavior should be observed in a console window when CPU load is evaluated. 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 thirty 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.

9.7.4 Checking for Timeouts and Dropped Connections

In InterMail, connections may time out and be dropped. In order to check for timeouts and dropped connections, it is suggested that the `mta.log`, `popserv.log`, and `imap.log` be checked for `NioMaxConnections` log events.

9.7.5 Checking Networked Resources

Basically, issues dealing with networked resources are similar to those of disk space, but with more difficult solutions. Most of the time, network I/O capacity is not a problem. If network problems are observed, first make sure that the network is clean (no unexpected traffic from other devices) and all segments are working correctly.

For example, `netstat -i 5` displays network traffic on all NICs (Network Interface Cards) for the past 5 seconds.

```
venus% 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
```

Figure 39. Checking Networked Resources.

In general, if the number reported in the `colls` column is high, then there is a problem with network resources.

9.8 Tuning the Performance of InterMail

In InterMail, performance tuning is an interactive process that involves observation by the administrator of response times, available memory, and other system activity and *then* modifying certain InterMail parameters. The process of tuning an InterMail system is iterative; however, this process can be specified into certain scenarios based on bottlenecks, problems, or events that have been logged by InterMail.

9.8.1 Tuning based on observed system performance

Access times to the MSS are slow

If the average access time to the MSS is observed to be at an unacceptable speed, then the value set in the `mta/timeoutServerDelivery` configuration key should be decreased.

Access times to the MTA are slow

When the MTA appears to be running slow, try to adjust the following:

1. Decrease the value in the `mta/cacheLimitInKB` configuration key.
2. Increase the value in the `mta/maxDirectDelivery` configuration key
3. Increase the value in the `mta/maxDirectKB` configuration key.

Note: *Keep in mind that the relationship of access time versus number of available connections is very inelastic; that is, in order to improve access times, it may be necessary to decreasing the number of available connections or adding new hardware.*

9.8.2 Tuning based on common log events

Connections are being timed out in the POP server

When connections are being timed out in the POP server (which can be readily identified by observing the `popserv.log` and searching for `NioMaxConnection` events), try the following:

1. Increase the value in the `popserv/maxSessions` configuration key.
2. Increase the number of Files Descriptors in a 3:1 ratio to the value specified in the `popserv/maxSessions` configuration key.

Connections are being timed out in the MTA

When connections are being timed out in the MTA (which can be readily identified by observing the `mta.log` and searching for NioMaxConnection events), the following steps are suggested:

1. Increase the value in the `mta/smtpDeliverNumThreads` configuration key.
2. Increase the value in the `mta/smtpAcceptNumThreads` configuration key.
3. Increase the value in the `mta/fileDescriptors` configuration key.

9.9 Expanding the Message File System

When an InterMail system is running low on space in the Message File System, the Message File System can be expanded by adding additional subdirectories. To expand the Message File system, follow these steps:

1. Either mount an additional external storage device, symbolically link an existing partition, or add a subdirectory under the directory defined in the `mss/messageFilesDir` configuration key.
2. Run `imbucketscreate` on the new subdirectory, designating the hierarchy and number of leaf directories (see Chapter 12 in the *InterMail Reference Guide* for more information on `imbucketscreate`).
3. Run `imbucketscreate` to load balance the existing directory(-ies) in the Message File System with the newly created subdirectory.

Expanding the Message File System Scenario

In the following example, the Message File System will be expanded to include an additional subdirectory.

```
venus% cd msgfiles
venus% ls
messages        buckets      buckets.dir
venus% mkdir Messages_2
venus% pwd
/vol1/imail/msgfiles
venus% ../lib/imbucketscreate messages_2 10 10
imbucketscreate: creating 100 directories under msgfiles_2.
imbucketscreate: finished creating 100 directories under msgfiles_2.
venus% ls
messages        messages_2      buckets      buckets.dir
venus% more buckets.dir
messages
messages_2
venus% ../lib/imbucketscreate
```

Figure 40. Expanding the Message File System.

In this example, an additional subdirectory called “messages_2” is created and the Message File System is created to have ten leaf directories and ten leaf subdirectories. After creating the Message File System, an entry will be written to the buckets.dir file, identifying the additional subdirectory. Finally, `imbucketscreate` is run again, without arguments, in order to distribute the message files in the newly expanded directory structure.

Note: *It is also possible to add an external mounted device on an additional host, although for performance, it may be desired to add the additional device on the host that currently contains the Message File System.*

9.10 Adding Directory Cache Servers

In order to perform account information queries in a more efficient manner, additional Directory Cache Server can be added to the InterMail system. By adding these additional servers, with a maximum of one server per host, the InterMail system can query a local cache file instead of querying a remote host for this information. This can provide faster lookups for account information.

In order to accomplish this, follow these steps:

1. Enable additional Directory Cache Servers by adding a `/<hostname>/dircachehost` configuration key for each host, and setting the `/<hostname>/sysadmin/imdircacheserv_run` configuration key to "on." This will enable the Directory Cache Server on a particular host.

   For each additional Directory Cache Server, add the order of failover for all Directory Cache Servers in the InterMail system. For example, if an InterMail system originally had a Directory Cache Server for "hostA" and then added one for "hostB," the following keys would be present:

   ```
   /hostA/dircachehost: [hostA:hostB]
   ```

   ```
   /hostB/dircachehost: [hostB:hostA]
   ```

   This would indicate that if the Directory Cache Server on hostA is unavailable, then hostA should use the Directory Cache Server on hostB. Additionally, the Directory Cache Server on hostB is unavailable, then it should use the Directory Cache Server on hostA.

2. Each host that will run a Directory Cache Server must also be able to recognize the Directory database connection and login. Therefore, configuration keys for the Directory database for the additional Directory Cache Servers will need to be added:

   ```
   /<hostname>/imdircacheserv/primaryDBconnection and
   /<hostname>/imdircacheserv/primaryDBuserInfo:
   ```

   These keys will contain the same values as the original host containing the Directory Cache Server, but apply to new hostnames in the system

3. Each additional host that will run a Directory Cache Server must be able to identify an additional

   - port that the Directory Cache server runs on (`/<hostname>/imdircacheserv/dirCachePort`)
   - port that SNMP listens on (`/<hostname>/imdircacheserv/adminPort`)
   - path/filename for the Directory Cache file (`/<hostname>/imdircacheserv/DBFilePath`)
   - directory where the server executable is located (`/<hostname>/imdircacheserv/runDir`)

9.10.1 Additional Directory Cache Server Scenario

To add an additional Directory Cache Server for an InterMail system that contains two hosts, hostA and hostB, and currrently runs one Directory Cache Server (on hostA), perform the following steps:

1. Run `imconfedit`.
2. Modify and/or add the following configuration keys:

```
...
/hostA/dircachehost: [hostA:hostB]
/hostB/dircachehost: [hostB:hostA]
/*/sysadmin/imdircacheserv_run: [on]
/*/imdircacheserv/primaryDBconnection: [IMD1]
/*/imdircacheserv/primaryDBuserInfo: [imail/imail]
/*/imdircacheserv/dirCachePort: [5889]
/*/imdircacheserv/adminPort: [7003]
/*/imdircacheserv/DBFilePath: [/vol1/imail/dirCacheDB]
/*/imdircacheserv/runDir: [/vol1/imail/tmp/imdircacheserv]
...
```

Figure 41. Adding an Additional Directory Cache Server.

Note: *Several keys are defined as system-wide; however, these may be modified if a path/filename or port needs to be changed on a per-host basis.*

3. Run `imctrl restart` on all affected hosts (see Chapter 3 of the *InterMail Operations Guide* or Chapter 12 of the *InterMail Reference Guide* for more information on `imctrl`).
4. Next, create the directory cache file (`dirCacheDB`) for each host by either:
   - Running `imdirsync` on the host to create a cache file on; this will create a new `dirCacheDB` file from the Integrated Services Directory.
   - Physically copy the `dirCacheDB` file from hostA to hostB and hostC (either by UNIX `cp` if hosts are cross-mounted, or through ftp if hosts are separate).

10

Backup and Recovery

The primary function of this material is to familiarize you with the important concepts behind backup and recovery. You can then apply these concepts in your specific backup and recovery environment. This is not a prescriptive, step-by-step method. Each site will have specific issues concerning their provisioning system, the size of their database, available hardware, and available person hours, that must be addressed on a case-by-case basis. A comprehensive backup strategy combined with good, tested recovery techniques can allow you to design your system to meet a required level of service.

The topics covered in this chapter include:

- An overview of backup and recovery in the context of InterMail
- Instructions to guide you in backing up and recovering message data and configuration information.
- Backup and recovery test procedures.

This chapter assumes you have thorough knowledge of your operating system, system configuration, backup software, and a good working knowledge of InterMail and its operations. General understanding of backups and backup strategy options is expected. It is also assumed that you have considered the requirements for recovery and the expected level of service your chosen backup strategy will protect. Oracle knowledge is helpful but not expected.

10.1 Introduction

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 upon a variety of factors including available hardware, message traffic, and any number of variables that are unique to your own InterMail installation.

Backup efforts in InterMail should focus primarily on:

- Account information
- Configuration information
- Persistent mail storage
- Temporary mail storage

***Note**: Backing up and restoring account information is not covered in this chapter. It is explained in the Integrated Services Directory Reference Guide.*

Regular system or environment backups should be performed in addition to the procedures discussed here. Environment backups should include regular incremental backups of non-InterMail files and occasional backups of InterMail binaries and configuration files.

10.1.1 Backup Options and Strategies

Unforeseen disaster, hardware failure, data corruption, and operator error can severely impact your InterMail system. A comprehensive backup strategy combined with an understanding of recovery techniques can allow you to design your system to meet the level of service that you need, ranging from high availability to a specific level of confidence of prompt recovery in the event of a failure or disaster.

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

- **Hardware failures** — CPU failure, memory errors, disk errors, disk controller, device drivers, or total system failure
- **Software failures** — errors that can occur from the operating system or related, third-party software applications
- **Operator errors** — errors that result from various user behaviors that might be called "unexpected," such as erroneously deleting files or logs, accidentally shutting down the system, or other problems

---

***Note**: No backup and recovery strategy can be considered effective until it has been fully tested under field conditions – we strongly urge you to conduct regular "fire drills" in a lab environment to ensure that your recovery strategy works.*

10.1.2 Recommended Backup Hardware

One of the first steps in developing a backup and recovery plan is to decide upon the hardware to be used for backup and recovery. A comprehensive review of the hardware available to implement your backup and recovery strategy is outside the scope of this document, but here are some general recommendations:

Disk Arrays

Disk arrays are fast and will take images of the data quickly without much overhead on the production system. Disk arrays are also expensive and have limited storage capacities; so, if cost is an issue, a spare disk array may be less desirable than tape. If performance is the prevailing issue, clearly the disk array option is the best choice. Spare disk arrays need not be RAID or mirrored.

Tape Devices

Tape devices are relatively slow and will require some overhead for a production system because an image copy will take considerable time to complete. However, tape devices are inexpensive and have unlimited storage potential. For long term storage, storing images to tape is the only practical solution. DLT (Digital Linear Tape) is much faster and higher capacity than DAT (Digital Audio Tape) but is also more expensive, especially when DLT jukeboxes are considered. Tape devices also tend to have higher error-rates relative to disk writes. Having multiple backups of any given file helps reduce the risk of file write errors.

Ideally, files that are backed up would first be copied to a separate fast disk array to minimize impact on production systems. The files could then be copied to tape or other backup media. Copying files directly to tape is cheaper but increases the overall load on production systems. Obviously, the choice is driven by hardware costs and desired performance.

High Availability Solutions

Another measure that may increase system reliability is the use of High Availability (HA) solutions. HA is usually implemented as a software failover mechanism that runs as a daemon process. HA requires at least one redundant machine with an identical configuration and shared disks to the production machine(s) for which it is meant to recover. During failover, the spare machine essentially assumes the identity of the failed production machine and begins operation as soon as the disk farm has synchronized and been mounted to the spare. Software.com does not provide this functionality and relies upon third party tools that have been certified for use with InterMail.

***Note**: For MTA hosts, regular online backups are not possible due to the transient nature of the data. The most reliable strategy for MTAs is full hardware mirroring. If this strategy is adopted, it is strongly suggested that the user employ full three-way mirrors with redundant controllers.*

10.2 Complete Image Backup

The first step after a new or upgraded installation of InterMail should be to make a complete image backup of both the InterMail-specific file systems and operating system file systems affected by the installation. This type of backup is taken only infrequently but is important for recovery. This backup will be used during a complete restoration to provide the basis for applying the current working system. Complete image backups should be done whenever the system is brought down in a quiescent state—the more recent the complete backup, the better.

***Note**: Whenever new software is installed, it should be backed up immediately. This type of backup should be included in 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 back them up accordingly.*

Immediately after an InterMail installation, no services are started. If you have started them manually or if this is an existing InterMail system, stop them on all MTA, MSS, and Integrated Services Directory machines prior to the backup.

The backup method employed to perform this operation is system and configuration dependent. You should at least 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 install, plus the system configuration files and user files.

Following this process, you should have a complete image of the working system and the operating environment. At this point, InterMail should be installed, running and, if this is a fresh installation, ready to be provisioned with new accounts.

10.3 Configuration Information Backup and Recovery

One host in the InterMail system is designated as the master configuration host. The Configuration Server is installed on this host. The Configuration Server maintains the master configuration database and is responsible for distributing the content of that database to all other InterMail hosts. It is extremely important that you back up the configuration information in the master Configuration Database.

You should make a backup of the master Configuration Database:

- after a major installation
- each time the configuration information is changed

You can set the `extraCopyConfigDBPath` configuration key to create an extra copy of the configuration database file. Specify a path (either absolute or relative to InterMail) for the extra copy, and the Configuration Server writes an extra copy of every new master configuration database file it writes to this path. If no path is set with this key, the Configuration Server will not write an extra copy of the configuration database.

To revert to an *earlier* version of the master configuration database, should that ever be necessary, use the `versionConfigDB` configuration key

Using versionConfigDB

You can use the configuration key `versionConfigDB` to make a backup copy of the current Configuration Database, so that in case of a hardware disk failure, you can restore the backup copy. If the key is set to `true`, then each time the Configuration Server writes a new master Configuration Database, the old one will be renamed to `config.db.YYYYMMDDHHMMSS`. This makes it easier to revert to an earlier Configuration Database should that later prove necessary. It is recommended that you periodically examine the size of the directory containing the backed up Configuration Database files and keep it in check.

10.4 Backing Up Mail in Process

Messages in transit may be stored temporarily. Under ordinary circumstances this mail in process is managed by the Queue Server and stored in the `queue` directory on the host on which the Queue Server resides. If mail needs to be stored temporarily and no Queue Server is available, the MTA assumes this role and stores messages in its local `spool` directory.

The directory structure used to organize mail in process is shown in Figure 42 Structure of the queue Directory. The structure of the directory is identical for the Queue Server and MTA; however, the root for the entire structure is called `queue` on the Queue Server and `spool` on the MTA.

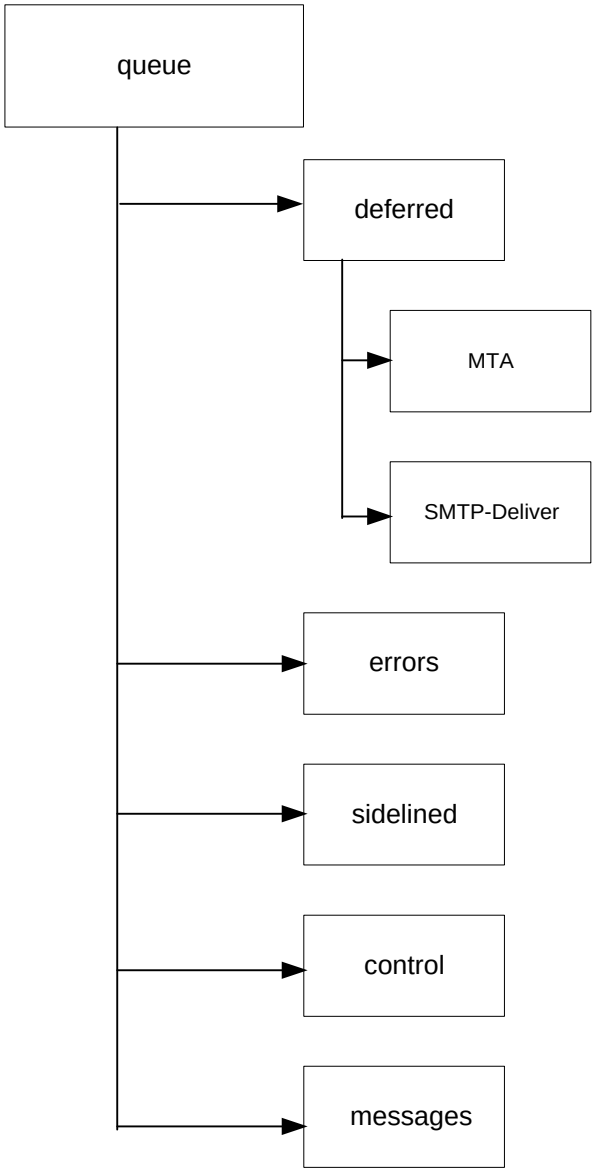

Figure 42 Structure of the queue Directory

10.4.1 Backing up the Queue Directory

Deferred mail is generally stored on the Queue Server, in a directory structure whose root is the queue directory. The queue directory contains all mail that is deferred for various reasons, in error, or sidelined. Therefore, it is very critical to back up this file system.

All queuing activity is journaled. Therefore, when used in conjunction with full backups of the queue directory it is possible to recover all messages in the queue directory to the very second that the directory was lost.

Backup should, therefore, be a two-step operation:

1. Back up the contents of the queue directory.
2. Back up the queue journal files and the garbage collector files.

   The Queue Server is typically co-located with MSSes. Therefore, this step can be performed as part of the backup required for the MSS.

10.4.2 Backing Up the Spool Directory

The one important difference between the Spool directory and the queue directory is that the spooling activity on the MTA is not journaled.

The Spool directory should be backed up periodically. It is important to note here that since there are no journal files for the Spool directory, some loss of data is possible. To reduce the chances of such a loss, the MTA server should have disk or file system redundancy.

10.5 Backing up Mailboxes and Messages

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. These tables also keep track of the status of message files (e.g., which users have read and deleted them). The actual message files (which contain the content of all messages) are present in the Message File System.

The Message File System and its associated Message Store Database should always be considered as a pair. If one is lost, the contents of the other would be meaningless. For every MSS host on your system, there is a Message File System and an associated Message Store Database. All of them should be backed up.

Figure 43 illustrates the structure and content of the Message File System and Message Store Database.

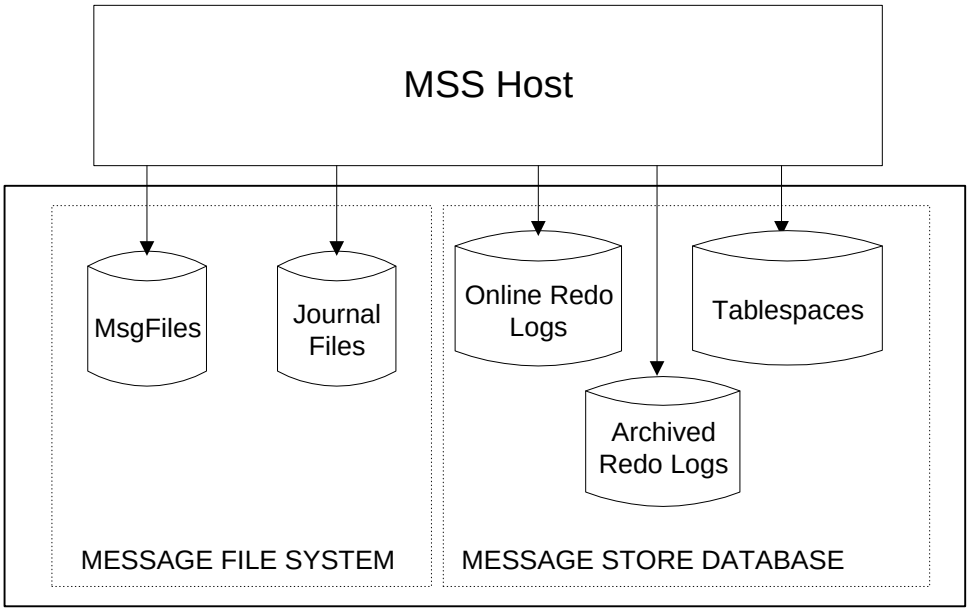

Figure 43. Message File System and Message Store Database

For a successful recovery of message data it is critical that all of the following be backed up:

- **MsgFiles —** All message files in InterMail are stored in the `MsgFiles` directory.
- **Journal Files —** InterMail creates a transaction record for every message file creation and deletion and stores these records in journal files.
- **Online Redo Logs —** These files are similar to the InterMail journal files as they store transactions; however online redo logs record transactions in the Message Store Database.
- **Archived Redo Logs —** These are online redo logs that have "filled up" with transactions and are archived.
- **Tablespaces —** This includes the system tablespace, index tablespaces, and data tablespaces.

There are two types of unsynchronized states between the tables in the Message Store Database and the message files:

- the database tables refer to missing message files (widows)
- there are message files with no corresponding pointers in the database tables (orphans)

In the event of a host or site disaster it is necessary to bring both of these storage areas up to an operational and synchronized state.

The design for InterMail assumes that in the event of an unsynchronized state (e.g., a pointer exists with no associated file or an incorrect file), the information contained in the Message Store Database is the reliable view of the overall message store.

10.5.1 Message File System Backup

The Message File System contains the content of all messages for all users on a given MSS host. The size of the Message File System can be formidable. Because of the large size, taking a full system backup nightly can be very inconvenient. Therefore, the strategy should be to take incremental backups. An incremental backup adds the new messages that are in the online message file system to the backup message file system copy. So the full backup is constantly updated by the incremental backup.

To keep the backup copy from growing uncontrollably you must prune those messages from the backup copy that have been deleted from the actual Message File System. To do this, you need to play the garbage collector journal files against the backup. Garbage collector journal files contain the pathnames of deleted message files, and playing them against the backup will delete these message files from the backup.

To back up and prune the Message File System, you should establish two utilities as cron jobs:

- Run the `imdbmsgbackup` utility to take incremental backups of the Message File System.
- Run the `imdbplaygcjrn` utility to play old garbage collector journal files against the backup and remove unnecessary messages.

Making Incremental Backups of the Message File System

The administration command `imdbmsgbackup` queries the MSS database to determine the list of message files to back up. This is more efficient than doing "stat" on a million message files to determine which ones are new.

The syntax for this command is:

```
imdbmsgbackup <backupDirectory> [-cronjob] [-tempdir <directory>] [-
batchsize <size>] [-messagebackuplimit <n>] [-sleeptime <m>][-threads
<n>] [-id <user>/<pass>] [-dbname $ORACLE_SID]
```

Where:

`<backupDirectory>`	The root of the backup Message File System copy. Message files are copied into a subdirectory of this directory named BACKUP_MESSAGES. ***Note:*** *if any subdirectory is missing,* `imdbmsgbackup` *will automatically create it*.
`-cronjob`	Write information to log entries in a log file instead of writing information to stdout.
`-tempdir <directory>`	Use the indicated directory for storage of temporary files. The utility creates a temporary file that contains the pathname of each message file that will be copied to the backup. If not specified, `imdbmsgbackup` uses the `tmp` directory.
`-batchsize <n>`	If specified along with `-cronjob`, `imdbmsgbackup` logs a message concerning its progress after every `<n>` message files are copied.

`-messagebackuplimit <n>`	When specified, `imdbmsgbackup` will not copy more than `<n>` messages. If not specified, `imdbmsgbackup` will copy all the messages it has not previously backed up. The actual number of messages backed up will tend to be less than `<n>`; the actual number of messages copied will be `<n>` less garbage messages.
`-sleeptime <n>`	If specified along with `-batchsize`, `imdbmsgbackup` sleeps for `<n>` seconds after copying each batch of messages.
`-threads <n>`	Instead of processing all work in one single thread, `imdbmsgbackup` will divide the messages needing backup into `<n>` sets of the same size before processing, and then will run a separate "thread" (actually a process) to handle each individual set.
`-id <user><pass>`	Oracle username and password to utilize. If not specified, `imdbmsgbackup` uses the mss configuration parameters to determine this value.
`-dbname $ORACLE_SID`	The SID of the database. If not specified, `imdbmsgbackup` uses the configuration parameters for the MSS to determine this value.

For details about the arguments used with the `imdbmsgbackup` command, see Chapter 12 of the *InterMail Reference Guide*.

The `imdbmsgbackup` utility does not backup garbage message files, that is message files for messages that are in no message store. Therefore, it is not critical if the utility is run before or after the garbage collector runs. However, it is advisable to avoid running `imdbmsgbackup` concurrently with the garbage collector to avoid contention for the Message File System and to avoid the situation where the garbage collector deletes a message that `imdbmsgbackup` is about to back up, resulting in `imdbmsgbackup` issuing a warning message that a message file is missing.

It is recommended that you run the `imdbmsgbackup` command nightly.

Pruning Unnecessary Messages from the Backup

The administration command `imdbplaygcjrn` plays garbage collector journals that it has not played before but are at least 24 hours old against a backup copy of a Message File System.

The `imdbplaygcjrn` command should be run nightly as a cron job. It is suggested that it be directed to play garbage collector journals from the InterMail journal directory, tar the journals it has played, and store the resulting compressed tar file.

The syntax for the `imdbplaygcjrn` command is:

```
imdbplaygcjrn [-cronjob] -journaldir <directory> [-journallimit
<n>][-playedjournalnames <file>] -messagebackupdir <directory>
[-logdir <directory>] [-logdir <directory>] [-logfile <file>]
[-tarandremove <directory>] [-compression <command>]
```

Where:

`-cronjob`	Writes log entries in a log file in the InterMail log directory instead of writing information to stdout.
`-journaldir <directory>`	The directory where the garbage collector journal files to play can be found.
`-journallimit <n>`	Restrict `imdbplaygcjrn` to playing no more than N garbage collector journal files.
`-playedjournalnames <file>`	If specified, `imdbplaygcjrn` records the name (just the name, not the path name) of each garbage collector journal file in the given file, one per line.
`-messagebackupdir <directory>`	The root of the backup copy of the Message File System to play garbage collector journal files against. The directory pathname should be the same as the directory pathname passed as the first argument to `imdbmsgbackup`.
`-logdir <directory>`	Ignored unless `–cronjob` is specified. The path name of the directory where the log file will be located. If not specified, the InterMail log directory will be utilized.
`-logfile <file>`	Ignored unless `–cronjob` is specified. The path name of the file to record log entries in. If not specified, the log file name will be of the format `PlayGCJournals.<time>.log`. If specified, `-logdir` is ignored.
`-tarandremove <directory>`	If specified, `imdbjrnplay` tars up all the garbage collector files it played against the backup Message File System, puts the resulting tar file in the specified directory, and deletes the journal files. The tar file is given a name of the format `jrnmgr.<time>.immssgcjrn.tar`.
`-compression <command>`	Compresses the tar file using the indicated command. Ignored unless `–tarandremove` is specified.

For details about the arguments used with the `imdbplaygcjrn` command, see Chapter 12 of the *InterMail Reference Guide*.

10.5.2 Message Store Database Backup

The Message Store Database is an Oracle database. It contains state information about the messages stored in the Message File System.

To fully recover the Message Store Database you must back up the following:

- Critical data files
- Archived redo logs
- Online redo logs

If any one of them is missing, the results of some (if not all) committed transactions will be lost. In InterMail terms, this translates into mail lost from mailboxes.

Two InterMail utilities, `imdbhotbackup` and `imoracopyarchredo`, are run as cron jobs to make hot backups of critical data files and copy archived redo logs.

Backing Up Critical Data Files

The critical data files contain the *database tables*, the *rollback segments*, and the *system tablespace*. The database tables contain data that records which messages are in which mailboxes. The rollback segments are vital to being able to successfully complete recovery. The system tablespace basically contains the "map" of the database that describes the schema and organization of tables, blocks, and files. All these are critical pieces of information.

Not all data files are critical; in particular, the index data files and the temporary tablespace data files. Indexes can readily be recreated from the tables. Temporary tablespaces are merely "scratch" areas that can be quickly recreated.

You need to run the administration command `imdbhotbackup` as a cron job to make backup copies of critical data files.

The syntax for this script is:

```
imdbhotbackup [-dryrun][-cronjob][-sleeptime <n>][-backupindexes]
[-othertemporarytablespaces <Temp1> [<Temp2>...][-backuptemp]
[-backupdestdir <directory>] [-totalbackupskept <n>]
[-backupcopydestdir <directory>][-totalcopybackupskept <n>]
[-backupdirexception <file> [<directory>]][-logdir <directory>]
[-backupcommand <command>] -dbname <$ORACLE_SID> -id
<USER/PASS>[@ORACLE_SID] [-dir]
```

Where:

`-dryrun`	Show what would happen when the script is run without copying any files or actually putting any tablespaces in backup mode. Writes the text of commands and sql statements it would have run in a "real" run to stdout
`-cronjob`	Suppress output to stdout; instead write log entries to a log file. If the `-logdir` flag is not specified, the directory where the log file will be created is determined by the `logDir` configuration key. If the `-dryrun` flag is used with `-cronjob`, commands and sql statements will still be written to stdout.
`-sleeptime <n>`	Sleep for <n> seconds between backing up each data file.
`-backupindexes`	Back up index tablespaces. By default, `imdbhotbackup` does not backup any index tablespaces, since these can be recreated from data tablespaces. `imdbhotbackup` creates scripts to recreate index tablespaces from scratch.
`-othertemporarytablespaces <Temp1> [<Temp>...]`	For a standard InterMail 3.1 or 3.2 install, use `othertemporarytablespaces TEMP`.
`-backuptemp`	Back up temporary tablespaces. By default, `imdbhotbackup` does not back up temporary tablespaces, since these can be easily and quickly recreated from scratch. `imdbhotbackup` generates scripts for recreating temporary tablespaces from scratch.
`-backupdestdir <directory>`	Specifies where to place the backup copies of data files. See following note.
`-backupcopydestdir <directory>`	Instead of overwriting the last backup in the backup directory given by `-backupdestdir`, `imdbhotbackup` will keep the <n>-1 previous hot backups in subdirectories named PREVIOUS.1, PREVIOUS.2, . . . , PREVIOUS.N-1. The current hot backup will be kept in a subdirectory called CURRENT if <n> > 1.
`-totalcopybackupskept <n>`	Analogous to the `-totalbackupskept` flag, except it applies to the `-backupcopydestdir` directory.
`-logdir <directory>`	Ignored unless the `-cronjob` flag is used.
`-backupcommand <command>`	When backing up a data file, instead of `imdbbackup` using its own mechanism to copy the data file, it invokes the specified command supplying a single argument, the complete path name of the data file. The command should return "0" if it succeeds.

`-dbname <$ORACLE_SID>`	The database instance name, as identified by the $ORACLE_SID environment variable.
`-id <USER/PASS> [$ORACLE_SID]`	Oracle name and password; this argument can be specified with an alternate database instance name.
`-dir`	Back up the Integrated Services Directory (Message Store Database is the default)

For a detailed explanation of the arguments for this command, see Chapter 12, *InterMail Reference Guide*.

The `imdbhotbackup` utility can, by default, only handle backups to disk. To handle tape, you need to write a customized command to backup a file to tape, and pass that command into `imdbhotbackup` by using the `–backupcommand` flag.

Here is a sample crontab entry for the `oracle` UNIX user for backup to tape. The crontab entry belongs to Oracle so that there will be no permission problems accessing data files.

```
30 3 * * * ORACLE_HOME=/volX/oracle/7.3.2 ORACLE_SID=IMMX
LD_LIBRARY_PATH=/volX/imail/lib:/volX/oracle/7.3.2/lib /volX/oracle/imdbhotbackup
–cronjob –sleeptime 120 –othertemporarytablespaces TEMP –backupdestdir /volY/backup –
totalbackupskept 1 –logdir /volY/backup/LOGS –backupcommand
/usr/local/bin/backuptotape –dbname IMMX –id imail/imail
```

The Oracle user (`oracle`, by default), is not a member of the InterMail group and will not be able to access the Configuration Database. Therefore, the `–logdir` flag is used to prevent the command from trying to look up `logDir` in the Configuration Database and failing.

Backing Up to Tape

Even if you are backing up all data files to tape, you still need to specify the `–backupdestdir` argument. This directory stores the following:

- Scripts to recreate indexes and temporary tablespaces
- A backup history file maintained by `imdbhotbackup`
- A backup copy of the database control file

The following is an example of a customized `backuptotape` command:

```
#! /bin/sh
if echo “$1” | cpio –ocBv > /dev/tapexas
   then
     exit 0
   else
     exit 1
fi
```

Backing Up to Disk

In the next example, a disk array is dedicated to storing backups. It is desirable that two backup copies be kept on the array: the latest backup and the previous one. The disk array is mounted at `/volB`.

Here is a sample crontab entry for the `oracle` UNIX user. The crontab entry belongs to the `oracle` user so that there will be no permission problems accessing data files.

```
30 3 * * * ORACLE_HOME=/volX/oracle/7.3.2 ORACLE_SID=IMM1
LD_LIBRARY_PATH=/volX/imail/lib:/volX/oracle/7.3.2/lib /volX/imail/bin/imdbhotbackup
–cronjob –sleeptime 120 –othertemporarytablespaces TEMP –backupdestdir /volB
–totalbackupskept 2 –logdir /volB/BACKUP_LOGS –dbname IMM1 –id imail/imail
```

Crashing during a Hot Backup

If a crash occurs while `imdbhotbackup` is running, there is a good chance the database will not come back up cleanly when the machine is rebooted. However, all the Oracle background processes will be successfully launched, so doing a "`ps`" will reveal that database processes are running, but the database will not be accepting any connections. The MSS processes will not come up.

The problem is that a crash during a hot backup of a data file leaves its file header in an inconsistent state. Oracle detects this at startup, issues an error saying "`ORA-01113: file X needs media recovery`," and refuses to fully open the database.

To get the database fully operational, log in as the `oracle` UNIX user, and run Oracle's `svrmgrl` utility as follows. Make sure the `ORACLE_HOME` and `ORACLE_SID` environment variables are set properly first.

```
% echo $ORACLE_HOME $ORACLE_SID
% $ORACLE_HOME/bin/svrmgrl
>connect internal
>startup
ORA-01113: file N needs media recovery
ORA-01110: datafile N: '/foo/bar/snafu.dbf'
>recover datafile '/foo/bar/snafu.dbf'
>alter database open;
```

If another ORA-01113 error is issued, just repeat the `recover datafile` and `alter database` steps. Use an `exit` command to terminate `svrmgrl`.

Online Redo Logs

The online redo logs contain the most recent set of changes to the database. One particular online redo log will be the "current" log where Oracle is recording all new changes. If the online redo logs are lost, in particular the "current" log, the most recent set of changes to the database will be lost.

The online redo logs are protected by Oracle's software mirroring feature.

Copying Archived Redo Logs

Oracle records all changes to the database in the current online redo log. At some point this log, which is a fixed size, becomes full. Oracle then switches to a different redo log and begins recording changes there. The previous current log is then copied by Oracle to the `archive_dest` directory. The copy in `archive_dest` is known as an *archived* redo log.

An archived redo log is a copy of a former *online redo log* that contains all the changes to the database during a particular period of time. This period is typically about five minutes. Archived redo logs are critical to restoration when a critical data file has been lost. The archived redo logs are needed to "roll forward" a backup copy of the data file. If one of the needed archived redo logs is missing, it is analogous to a link missing from a chain. A backup copy can only be rolled forward to the point in time where the missing log started recording changes.

Archived redo logs should be protected by backing them up shortly after they are created. This ensures two copies always exist: one in Oracle's `archive_dest` directory and one in the backup directory, or one in Oracle's `archive_dest` and one among the online redo logs.

There are two problems with archived redo logs:

- Archived redo logs are a single point of failure that can prevent full recovery
- Archived redo logs, if left unchecked, will eventually fill up the `archive_dest` directory.

Losing one archived redo log is enough to make full recovery impossible. The `imdbcopyarchredo` utility addresses this by copying archived redo logs soon after they are created so that two copies exist on independent devices.

When the second problem occurs, the database literally freezes up. The `imdbcopyarchredo` utility, therefore, regularly prunes old archived redo logs from the `archive_dest` directory.

The `imdbcopyarchredo` utility is intended to be run as a cron job at an interval that is a small multiple of the average time between log switches. The idea is to ensure that `imdbcopyarchredo` runs frequently enough so that an archived redo log is copied before its online redo log original gets overwritten.

10.6 Restoring the Queue and Spool Directories

To recover mail in process you need to restore the `queue` and `spool` directories.

10.6.1 Restoring the Queue Directory

To restore the `queue` directory, do the following:

1. Restore the last backup of the `queue` directory.
2. Replay the journal files from the time of the last backup using the `imjrnrecover` utility. To recover queue data only, confine yourself to the journal files with the .queue extension.

***Note**: If you have lost both the Message File System and the* `queue` *directory, you should postpone running the* `imjrnrecover` *utility until after restoring the last backup of the* `queue` *directory and the Message File System.*

10.6.2 Restoring the Spool Directory

To restore the `spool` directory, restore the last backup of the `spool` directory.

The MTA does not support message journaling so any mail in process that was not included in the last backup may be lost.

10.7 Restoring the Message File System

To enable recovery of the Message File System, you require the following.

- Hot backups of the Message File System
- Journals maintained by Message Store Server processes

The journals record all transactions (writes and deletes) to the Message File System. Journaling provides protection for messages received between regularly scheduled backups.

Journals, when combined with backups, allow you to recover your system to the exact point it was at before disaster struck. It is therefore important to take frequent (at least hourly) backups of these files to protect against media failures or other file loss.

***Note**: It is also important to keep the journal files on a disk system which is separate from the Message File System.*

A hot backup is performed as a cron job. Between hot backups, journal entries are written and, depending upon the number of messages moving through the system, one or more journal files are created. Until the next hot backup, journal files will be the last record of transactions to the MSS, with the journal files backed up into the next hot backup.

The basic strategy for recovering the Message File System is simple: copy the backup Message File System, and roll it forward using journal files.

1. Recursively copy the archived message file directory to `$MessageFileDir`.

   Instead of doing this in one step using "`cp -r`", it is suggested that you copy each subdirectory separately using "`cp -r`". In this way, if there is some problem doing a copy, you will not have to start over.

2. After the Message File System has been completely copied, play the journal files against it to bring it up to date. Use the `imjrnrecover` utility to do this.

There are three flavors of journal files: MSS journals, Queue journals (discussed in section 10.4.1) and GC (garbage collector) journals. After the system is back up, GC journals can be played at your convenience. The only drawback to delaying the replay of GC journals is that the recovered Message File System may initially contain garbage messages that results in wasted space.

To determine what journal files need to be played against the backup, look at the content of the file `message_file_system_backup/message_backup_state.txt`. In this file you will see a line of the form "LAST_BACKUP_SNAPSHOT_TIME=YYYYMMDD.HHMMSS." This line records the time the last incremental backup began. You will need to play all journal files with a time in their name newer than this time, plus the journal files just older than this time. The time in journal file names indicates when the journal file was first opened, and indicates the start of the period where it records changes to the Message File System.

***Note**: Keep in mind the incremental backup time, and make sure you play enough journal files.*

Activating the Stand-In

Since restoring the Message File System will take some time, you may want to bring up an initially-empty "stand in" Message File System so that customers can at least read their new mail while the restore is in progress. Mailboxes will be created in the stand-in automatically as messages arrive or users attempt to retrieve mail if the `createsMboxes` configuration key is set to `true`. After restoration is completed, all mail contained in the "stand in" must be transferred into the restored Message File System.

Deactivating the Stand-In

After the Message File System has been copied and all the relevant MSS journals have been played against it, you can either choose to have the restored system take over and deactivate the stand-in, or leave the stand-in active and keep the restored system off line. In either case, it will be necessary to move all the mail from the inactive system to the active one.

The inactive system's MSS database records which messages on the inactive system are in what user's mailbox. This data, together with the message body in the Message File System, can be used to redeliver each message to the proper mailboxes on the active MSS.

10.8 Restoring the Message Store Database

In this section we'll go over different scenarios that require restoration of all of the Message Store Database or specific portions of it.

Loss of the Oracle Home Directory

In this section we discuss what to do if the "`oracle`" Unix user's home directory is lost. If installation defaults were taken, all the Oracle binaries and libraries, plus database parameter files, have been lost.

The Oracle binaries, located under the `$ORACLE_HOME` directory, are not backed up nightly. Recovering in this situation will be painful if another copy doesn't exist within easy reach. Try copying Oracle binaries from another machine. If that isn't a possibility, the Oracle binaries --- absolutely everything under `$ORACLE_HOME` should be committed to tape.

Recovery Procedure

Recreate the "oracle" home directory, including `.cshrc` and `.profile` files with `TNS_ADMIN`, `ORACLE_SID`, and `ORACLE_HOME` set properly. `LD_LIBRARY_PATH` must include `$ORACLE_HOME/lib`.

In the following instructions, assume that HOME is set to the `oracle` user's home directory path name.

1. Restore `$ORACLE_HOME` from a copy.
2. Make directories `$HOME/admin/$ORACLE_SID/dump_dest/bdump`, `…/cdump`, and `…/udump`. Make `$HOME/admin/$ORACLE_SID/pfile`.
3. Copy the hot backup file `BAK.init$ORACLE_SID.ora` to `$HOME/admin/$ORACLE_SID/pfile/init$ORACLE_SID.ora`.
4. Create a symbolic link:

   ```
   ln –s $HOME/admin/$ORACLE_SID/pfile/init$ORACLE_SID.ora
   $ORACLE_HOME/dbs/init$ORACLE_SID.ora
   ```

5. Copy the hot backup file `BAK.configIMDB.ora` to `$HOME/admin/$ORACLE_SID/pfile/configIMDB.ora`.
6. Copy the hot backup files `BAK.tnsnames.ora` and `BAK.listener.ora` into `$ORACLE_HOME/network/admin/tnsnames.ora` and `…/listener.ora`.

Loss of System Tablespace

The system tablespace file is typically named `SystemTablespace.dbf`. The system tablespace records the schema and structure of the database, and Oracle cannot start the database without it.

Use the following procedure to recover from Oracle system tablespace file failures.

Warning: Use this procedure only if you have a thorough understanding of the process and if you are certain that only the data files you are restoring are the ones affected and that you are operating on the correct tablespace.

This type of restoration is only effective when a single tablespace has been damaged due to media failure. The advantage of restoring only data files is that restoration time is reduced in comparison to a full Oracle recovery.

It is possible to run the data file recovery over a live system recovering only the specific tablespace. However, if any InterMail tablespaces (including TEMPORARY, SYSTEM, or RBS) are involved, the system should not be mounted and open for InterMail operation since the database objects required will be affected. It is important to note that *only* complete recovery may be used for data file recovery. If you do not have a complete set of redo logs available for Oracle complete recovery, you are required to perform a full Oracle recovery.

1. Copy `BAK.SystemTablespace.dbf` from your last hot backup to `SystemTablespace.dbf` in the proper directory.
2. Run Oracle Server Manager and start the Oracle instance in restricted mount mode.

   ```
   SVRMGR>  CONNECT INTERNAL
   SVRMGR>  STARTUP RESTRICT MOUNT;
   ```

***Note**: If you need to restore the database to a different path or file system you will need to issue an ALTER DATABASE command with the CREATE DATAFILE option on a mounted (but closed) database to redirect the file paths for the tablespace data files. A detailed discussion of this process and when and how to use it can be found in the Oracle Backup and Recovery Documentation.*

---

3. You must have all redo log files (including the online redo logs) without any gaps in the directory specified in the `init$SID.ora` file. You may run the following in server manager (where datafile1… are the file system names for the data files including the path as listed in `SYS.DBA_DATA_FILES`). This step will take some time.

```
SVRMGR>  RECOVER AUTOMATIC DATAFILE ‘datafile1’, ‘datafile2’,… ;
SVRMGR>  ALTER DATABASE OPEN ;
```

***Note**: If your redo log files are not in the original path/file system, you must also specify the path using the FROM path clause for the RECOVER command. Also, other recover options are available if the above commands cannot recover the database. A detailed discussion of the RECOVER command can be found in the Oracle Backup and Recovery documentation.*

---

4. When the process completes successfully, you can then shut down the instance using Server Manager.

```
SVRMGR>  SHUTDOWN IMMEDIATE
```

   It is advisable to restart the machine to ensure consistency and stability.

5. Start the affected MSS Oracle instance on the recovered MSS machine and start InterMail on the MSS.

Loss of Temporary Tablespace that is not Backed Up

By default, `imdbhotbackup` does not backup temporary tablespaces. However, it does include scripts with the nightly hot backup that can be used to quickly and easily recreate each one. For every temporary tablespace that is not backed up, there will be scripts `FilesOffline`*`TEMPTBS`*`.sql` and `RecreateTempTbs`*`TEMPTBS`*`.sql` included in the hot backup.

1. Create all necessary directories.
2. Run Oracle Server Manager and start the Oracle instance.

```
Svrmgrl
>connect internal
>startup mount
>@/archive/store1/HOT_BACKUP_COPIES_IMD1/CURRENT/FilesOffline*TEMPTBS*.sql
>recover automatic database
>alter database open
>@/archive/store1/HOT_BACKUP_COPIES_IMD1/CURRENT/RecreateTempTbs*TEMPTBS*.sql
```

Loss of a Data Table Tablespace

Files making up data table tablespaces are included in the hot backup.

For each corrupted data table tablespace file, copy a hot backup copy to where the online copy should be.

1. Run Oracle Server Manager.

```
Svrmgrl
> connect internal
> startup mount
> recover  automatic database
> alter database open
```

Loss of Rollback Segment Tablespace

By default, there is one rollback segment tablespace in an InterMail database containing all the database's rollback segments. Rollback segments are vital to Oracle database recovery. Files making up rollback segment tablespaces are included in the hot backup.

For each corrupted rollback segment tablespace file (typically there will be just one, `RollbackTablespace.dbf)`, copy a hot backup copy to where the online copy should be.

1. Run Oracle Server Manager.

```
Svrmgrl
> connect internal
> startup mount
> recover  automatic database
> alter database open
```

Loss of Index Tablespaces

The index tablespaces are not included in the hot backup by default. However, the hot backup does include scripts for recreating all indexes from scratch, `IndexFilesOffline.sql` and `RecreateIndexTablespaces.sql`.

Make absolutely certain that `immssgc`, the garbage collector, is not running and cannot run until all the indexes are restored. Comment out "imail's" crontab entry for `immssgc`, and do a "`/bin/ps –ef | grep immssgc`" to make sure it is not running.

1. Run Oracle Server Manager.

```
Svrmgrl
> connect internal
> startup mount
> @IndexFilesOffline.sql
> alter database open
> @RecreateIndexTablespaces.sql
```


Loss of an Online Redo Log

The online redo logs should be mirrored in hardware or software. The goal is to make it unlikely both copies of an online redo log will be lost. If that happens, the database cannot be fully recovered.

When one mirror is lost, it can be restored by copying the other mirror. By default, redo log files have names like `RedoGrpNMemX.dbf`. Files that have equal values of N in their redo log names are mirrors of one another. The X is the mirror number, from 1 to the number of mirrors.

Loss of the archive_dest Directory

If all of the archived redo logs have been lost, take a hot backup as soon as possible.

Loss of a Combination of Files

If several files are lost belonging to several of the categories, combine the instructions from each section as follows:

1. Do all the steps from each section up to the point `svrmgrl` is run.
2. Run the Server Manager.
3. Do all the subsequent steps from each section up to the point the "`recover automatic database`" statement is issued.
4. Run "`recover automatic database`".
5. Do all the subsequent remaining steps from each section.

10.9 Testing Backup and Recovery Procedures

In order to ensure that backup and recovery procedures will be effective for your InterMail installation, it is important to test all procedures before using them in a production environment. Regular fire drills should also be conducted in a lab environment to verify the backup procedures and backup hardware.

The results from the following test procedures should help you tune your backup and recovery strategy to obtain optimum system performance and reliability. It is very important to establish a regular schedule of backups and system monitoring to ensure that the InterMail system can run smoothly with minimum intervention and service impact.

Establishing Test Procedures

The following four test procedures will model the potential failures that may occur, and the subsequent recovery strategies:

- **Deletion of total system**: This simulates a total system loss, caused by conditions including total hardware failure, disk array loss, operator mistakes that result in an unrecoverable system loss, and environmental conditions that may severely affect the system.
- **Removal of Oracle and all associated files**: This simulates total Oracle failure in which either the Oracle database, tablespace, or data files have been lost or totally corrupted.
- **Removal of Oracle control file**: This simulates the loss of an Oracle control file
- **Delete messages**: This simulates the loss of individual messages.

Testing the Procedures

The following steps are recommended to test the backup/recovery procedures:

1. Establish test platform. Before production rollout, it is desirable to construct a trial run and perform a backup and mock restore using the production configuration.
2. Document production environment. Produce detailed installation options and backup strategy documentation.
3. Install InterMail. Create fresh installation for testing
4. Perform initial backup. Create backup of configuration of test environment
5. Build simulated mailboxes and sample messages. Add users, mailboxes, sample messages and other data to simulate working system
6. Backup simulated work environment. Take incremental backup of working test system.

10.9.1 Full System Loss Test and Recovery

1. Shut down InterMail and remove installation to simulate catastrophic disaster.
2. Restore from initial backup and apply incremental backup to recover previous system.
3. Test the recovered environment.

Test Procedure

1. Create test accounts and send test messages.
2. Shutdown InterMail and Oracle.

```
% lib/imservctrl stop
% lsnrctl stop
% shutdownDB IMM1 // or MSS Oracle SID
% shutdownDB IMD1  // or DIR Oracle SID
```


3. Delete the file systems containing InterMail, InterMail journals, message text, Oracle, Oracle tablespaces, Oracle redo archive logs, and Oracle Control files.

   [Optional] Delete the Operating System, machine configuration, and tuning files and restore from the bare machine to the required OS and Patch levels. Re-configure system files and tuning parameters as required for InterMail.

4. Restore complete backup.
5. Restore last available hot backup.
6. Copy backup control file to all Oracle control file paths. The control file paths can be found in `$ORACLE_SID_home_directory/pfile/configIMDB.ora` file under the key "`CONTROL_FILES`". The backup control file needs to be copied to the same path/file name as each item in the list before the Oracle instance will start.
7. Start up the database and recover using the redo log files.
8. Replay the MSS journals to restore the latest state of the Message File System.

   ```
   imjrnrecover –a
   ```

9. Restart the InterMail and Oracle services.

   ```
   % startupDB IMM1 // or MSS Oracle SID
   % startupDB IMD1  // or ISD Oracle SID
   % lsnrctl start
   % lib/imservctrl start
   ```

10. Attempt to POP the test messages created in step 1.
11. Create new test mail accounts and messages to test the restored services.

10.9.2 Oracle Tablespace/Data File Failure and Recovery

1. Shut InterMail down and remove a specific tablespace data file such as POM01 to simulate an Oracle tablespace failure.
2. Restore the data file(s) from the latest backup and apply Oracle redo logs to restore previous system.
3. Test the restored environment.

Test Procedure

1. Create test accounts and send test messages.
2. Shutdown InterMail and Oracle.

   ```
   % lib/imservctrl stop
   % lsnrctl stop
   % shutdownDB IMM1 | IMD1 // or MSS Oracle SID
   ```

3. Delete the Oracle POM01 Tablespace datafile(s).
4. Restore data file set from last available hot backup.
5. Shut down the Oracle services.

6. Start up the database and recover using the redo log files (see the section on Oracle recovery for details).
7. Restart the InterMail and Oracle services.

```
% startupDB IMM1 // or MSS Oracle SID
% startupDB IMD1  // or ISD Oracle SID
% lsnrctl start
% lib/imservctrl start
```

8. Attempt to POP the test messages created in step 1.
9. Create new test mail accounts and messages to test the restored services.

10.9.3 Oracle Control File Recovery

Shut InterMail down and remove a specific Oracle control file such as `ctrl$ORACLE_SID01.ctl` to simulate an Oracle Control file failure. Restore the control file from the latest backup and apply Oracle redo logs to restore the previous system. Test the restored environment.

Test Procedure

1. Create test accounts and send test messages.
2. Shut down InterMail and Oracle.

```
% lib/imservctrl stop
% lsnrctl stop
% shutdownDB IMM1 | IMD1 // or MSS Oracle SID
```

3. Delete the Oracle control file `ctrl$ORACLE_SID01.ctl`.
4. Restore the backup control file from last available hot backup.
5. Start up the database and restore using the redo log files (see the section on Oracle recovery for details).
6. Shut down the Oracle services.
7. Restart the InterMail and Oracle services.

```
% startupDB IMM1 // or MSS Oracle SID
% startupDB IMD1  // or DIR Oracle SID
% lsnrctl start
% lib/imservctrl start
```

8. Attempt to POP the test messages created in step 1.
9. Create new test mail accounts and messages to test the restored services.

10.9.4 Message Deletion Recovery

1. Shut down InterMail and move specific message files from the Message File System to a temporary area to simulate accidental deletion.
2. Attempt to POP these messages and restore the message files.
3. Use `imbadmsglist` and `imbadmsgfix` to restore the original message connections.
4. Test the restored environment.

Test Procedure

1. Create test accounts and send test messages.
2. Move the specific test message files from the Message File System to a temporary location.
3. Attempt to POP these messages and receive the `-ERR` message: this places the message entry in the database to a `.ERROR` folder for later repair.
4. Move the message files back to the original locations.
5. Use `imbadmsglist` to produce a message list for `imbadmsgfix` to repair.
6. Re-attempt to POP these messages.
7. Create new test mail for the affected accounts to test the restored services.

11

Troubleshooting Scenarios

InterMail is a robust messaging system designed for continuous use seven days a week, 24 hours a day. To best prepare yourself for managing a mail server in this environment, it is important to understand the relationships between system components and the overall impact on mail service should one component become unavailable.

To assist you in obtaining the necessary understanding, we've created a model mail system to be used in discussing a variety of troubleshooting scenarios. Our intention is not to provide a definitive list of issues and solutions, but rather to equip you with an understanding of the entire system in relation to its individual parts.

For each scenario proposed, the following information is furnished:

- Immediate effect on the system
- Impact on service
- Possible actions to take to remedy the situation

11.1 Sample System Configuration

The troubleshooting scenarios that follow reference the sample InterMail system shown in Figure 44. This diagram is not intended to represent the typical InterMail installation. In fact, given the flexibility of InterMail, no such standard exists. This sample is one of many viable InterMail configurations. Your specific installation will undoubtedly vary, and your associated troubleshooting procedures should be modified accordingly.

Our sample system includes two hosts dedicated to message delivery, four hosts dedicated to message storage, and two hosts dedicated to servicing client requests for mail retrieval. In addition, our sample system includes a host dedicated to system management, and another host dedicated to the storage of account and domain information.

Each message delivery host runs an MTA and a Directory Cache Server. (See Hosts A and B in the diagram.) Each host dedicated to message storage runs an MSS and a Queue Server. (See Hosts C, D, E, and F in the diagram.) The hosts responsible for message retrieval run a POP Server and an IMAP Server. (See Hosts G and H in the diagram.) A Configuration Server and an SNMP server are installed on the host responsible for system management (Host G). Account and domain information are maintained in the Integrated Services Directory (ISD) installed on Host H.

***Note**: A Manager Server runs on each host.*

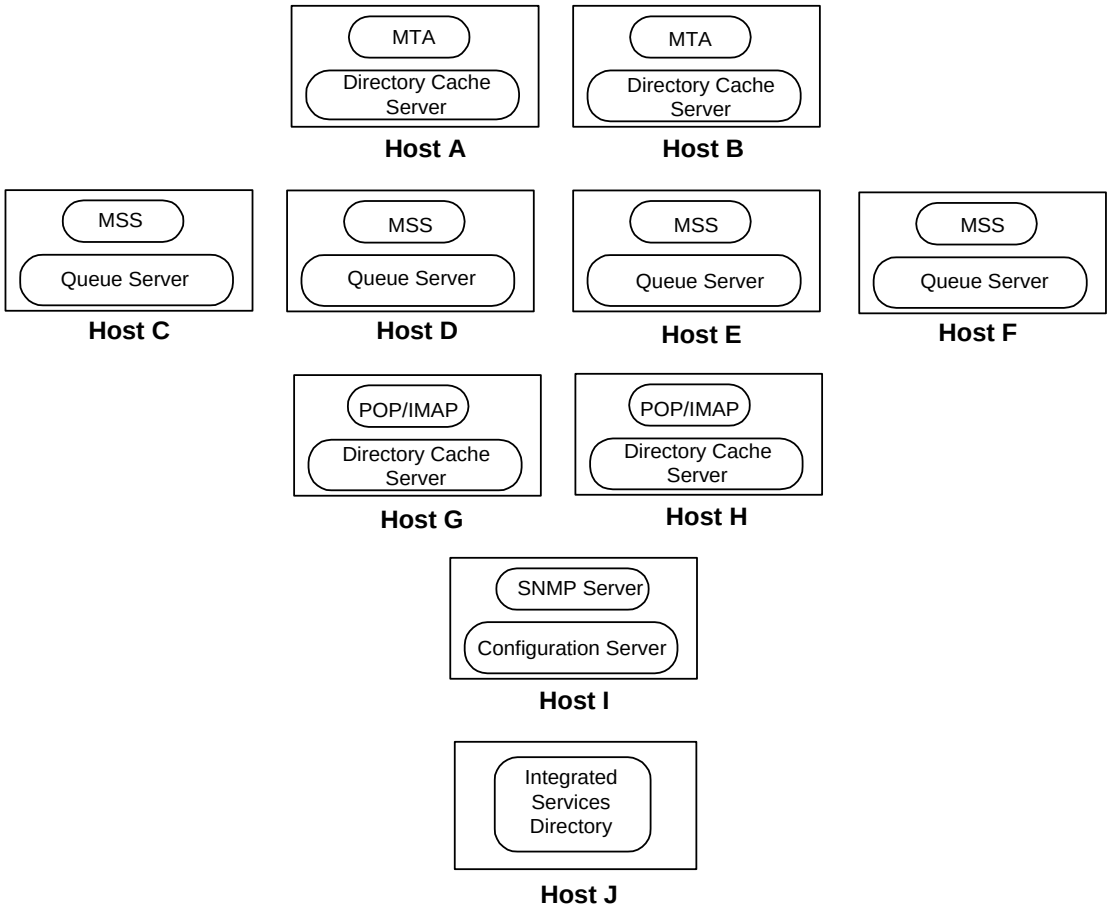

Figure 44. Sample System Configuration

To simplify the troubleshooting discussions and concentrate our comments on the most important issues, the following assumptions have been made:

- The DNS records for this site map the domain name associated with this mail server to multiple IP addresses. When the DNS server performs name resolution, it “round robins” through the records, allowing the load for the domain to be distributed among several machines.
- The `queueServerHosts` configuration key has been set to allow each MTA to locate multiple Queue Servers. The primary Queue Server (the one typically contacted) is the one located on the same host as the MTA requesting contact.
- The `dirCacheHosts` configuration key has been set to allow each MTA, each POP Server, and each IMAP Server to locate multiple Directory Cache Servers. The primary Directory Cache Server (the one typically contacted) is the one located on the same host as the server requesting contact.

- Host F is fully configured, but the servers installed on that machine are not used in standard production. This host is reserved as a “dark machine” that can substitute for another MSS host, if necessary.

11.2 Sample Scenarios

The scenarios described in the sections that follow are organized by server type. They are written in reference to particular incidents, but the information can be generalized to apply to other, similar circumstances.

11.2.1 Directory Cache Server is Unavailable

The Directory Cache Server responds to requests for account information. It interacts with the MTA, the POP Server, and the IMAP Server, supplying and verifying account information as required.

The design of the InterMail messaging system allows one Directory Cache Server to take over for another automatically, if required. Our sample system is configured to support this feature.

Scenario

The Directory Cache Server on Host A is unavailable

Effects

When the MTA on Host A is unable to contact the Directory Cache Server on Host A, it sends the request for account information to the Directory Cache Server on Host B. The MTA continues to send the next 99 requests for account information to the Directory Cache Server on Host B. For the 100th request, it re-attempts contact with the Directory Cache Server on Host A. If the Directory Cache Server on Host A responds, standard operation resumes. If the MTA still cannot reach the Directory Cache Server on Host A, it continues sending requests to the Directory Cache Server on Host B.

Impact on Service

There is no noticeable effect on mail delivery or retrieval.

Actions

1. Bring the server up again, if possible. Check the Directory Cache Server log files for cause of failure.
2. If the Directory Cache Server on Host A has been down for less time than the value specified in the `logAgeHours` configuration key, no further action is required; it will automatically synchronize the content of its local cache with the content of the ISD on the next update cycle (every 60 seconds, by default).
3. If the Directory Cache Server on Host A has been down longer than the time specified in the `logAgeHours` configuration key, it will be unable to synchronize its cache automatically. Instead, you must either copy a cache file from one of the other Directory Cache Servers (on Hosts B or C) or run the `imdirsync` command to create a new cache file directly from the ISD.

11.2.2 MTA is Unavailable

The MTA handles the receipt of all incoming messages. For mail addressed to local users, it obtains account information from the Directory Cache Server then hands the message to the appropriate Message Store Server. For recipients in other domains, it sends the message to the remote location over the Internet.

Scenario

The MTA on Host B is unavailable.

Effects

Round-robin DNS redirects mail to the MTA on Host A. Ordinarily, there are no issues involving spooled mail. However, if *no* Queue Servers are running, mail may have been spooled locally on the MTA host. Any mail stored temporarily on Host B continues to be deferred till the MTA on Host B returns to service.

Impact on Service

Mail delivery may be slow because the MTA on Host A is carrying the entire burden. There is no effect on mail retrieval.

Actions

1. Bring the server up again, if possible. Check the MTA log files for cause of failure.
2. If it appears the server will be down for more than a few minutes, you should start up an MTA on an alternate InterMail machine and roll the IP address for Host B to that machine.

11.2.3 MSS is Unavailable

The Message Store Server (MSS) is responsible for hosting mailboxes, storing incoming messages, and providing access to clients' mailboxes. Large InterMail systems may have several MSS machines.

Each MSS has its own database, which contains a unique set of mailboxes. Information about these mailboxes, including their users and their exact physical location, is maintained by the Integrated Service Directory (ISD).

Message storage is transaction-based and transactions are recorded in journals for full data integrity and recoverability.

Scenario

The MSS on Host C is unavailable.

Effects

Mail destined for users whose mailboxes are managed by this MSS is deferred internally. The MTAs pass the mail to the Queue Servers for temporary deferral and re-attempt delivery on a regular basis. See chapter 7 in this manual for more information about delivery of deferred mail.

Impact on Service

Users whose mailboxes are managed by this MSS are unable to retrieve their mail. Mail delivery and retrieval for users whose mailboxes reside on Hosts D and E is unaffected.

Actions

1. Restart the MSS and resume service; check the MSS log files for the cause of the failure.
2. Assuming the server cannot be restarted but the MSS database and the Message File System are fine, *and* that Host C is dual ported to Host F (the hot spare); start up an MSS on Host F, roll the IP address from Host C to Host F and allow it to take over.
3. If Host C and Host D are dual ported you could fail the MSS on Host C over to Host D. However, be aware that the combined burden cannot exceed 100% capacity of Host D. Also watch for conflicts in MSS ports.

11.2.4 Message Store Database is Unavailable

The MSS database contains data about message store content (folders and message file pointers) along with indexing information for message headers and Message Store Server statistics.

Scenario

The Message Store Database on Host C is unavailable.

Effects

Mail destined for users whose mailboxes reside on Host C is deferred internally. The MTAs pass the mail to the Queue Servers for temporary deferral and re-attempt delivery on a regular basis.

Impact on Service

Users whose mailboxes reside on Host C are unable to retrieve their mail. Mail delivery and retrieval for users whose mailboxes reside on Hosts D and E is unaffected.

Actions

1. Shut down the MSS pointing to this database.
2. Restore last good backup of the MSS database.
3. Restore the archived redo logs.
4. Restart the MSS. (At this point service to all users is restored.)

For more information on restoring the Message Store Database, see Chapter 10 in this manual.

11.2.5 Message File System is Unavailable

Message files contain all message data - header, body, and attachments. InterMail stores one message file per message on each MSS host, regardless of the number of recipients addressed in the message.

The effect and recommended response to message file loss varies based on the number of files in question. The scenarios described in the sections that follow illustrate the differences in impact and approach.

Scenario 1

A small portion of the Message File System, one leaf directory on Host C, is lost.

Impact on Service

Mail delivery continues uninterrupted. Some users with mailboxes on Host C are unable to retrieve a portion of their mail. Mail retrieval for users whose mailboxes reside on Hosts D and E is unaffected.

Actions

1. Leave the MSS up and running.
2. Copy the directory from your latest file system backup.
3. Run `imjrnrecover` to bring the content of the directory up-to-date.
4. Run `imbadmsglist`. It produces a list of messages that could not be retrieved via the POP server and were therefore moved from a user's INBOX to their .ERROR folder.
5. Run `imbadmsgfix` to move the restored messages back to the user's INBOX folder.

Scenario 2

The whole Message File System on Host C is lost.

Effect

Incoming mail destined for users with mailboxes on Host C is deferred internally.

Impact on Service

Users with mailboxes on Host C are unable to retrieve their mail. Mail retrieval for users whose mailboxes reside on Hosts D and E is unaffected.

Actions

1. Stop the MSS on Host C.
2. Copy the backup file system.
3. Play the journal files to bring it up-to-date.
4. Run the `imbadmsglist` and `imbadmsgfix` utilities.
5. Restart the MSS.

For more information on restoring the Message File System, see Chapter 10 in this manual.

11.2.6 Queue Server is Unavailable

If the MTA cannot deliver a message immediately (as when a remote host is temporarily off line) it passes the message to a Queue Server, which stores it in the `queue` directory. At regular intervals, the MTA requests stored messages from the Queue Server and attempts to deliver them again. The Queue Server also provides temporary storage of messages that exceed the MTA's limits for message size, number of recipients, or time required for delivery.

The design of InterMail allows one Queue Server to take over for another automatically, if required. Our sample system is configured to support this feature.

Scenario 1

The Queue Server on Host C is down.

Effects

When the MTA on Host A is unable to contact the Queue Server on Host C, it sends the request for temporary mail storage to the Queue Server on Host D. The Queue Server on Host D provides temporary storage in its `queue` directory. The MTA periodically re-attempts contact with the Queue Server on Host C. If the Queue Server on Host C responds, standard operation resumes. If the MTA still cannot reach the Queue Server on Host C, it continues sending requests to the Queue Server on Host D.

Impact on Service

There is no noticeable effect on mail delivery or retrieval.

Actions

1. Bring the server up again, if possible. Check the Queue Server log files for cause of failure.

Scenario 2

The Queue directory on Host C is lost.

Impact on Service

There's no effect on sending or receiving mail. Mail that had previously been deferred will remain undelivered until the content of the Queue directory is restored.

Action

1. Shut down the affected Queue Server.
2. Restore latest backup of the `queue` directory.
3. Run `imjrnrecover` to restore the lost files.
4. Restart the server.

11.2.7 POP Server is Unavailable

The POP Server handles requests for message retrieval from any client that supports the POP3 protocol. It communicates with the Directory Cache Server to validate the user's login name and password, and to obtain the name of the MSS host on which the user's mailbox resides. The POP Server also communicates with the Message Store Server to service message retrieval requests on behalf of the client.

Scenario

The POP Sever on Host G is unavailable.

Effects

Round-robin DNS will send client requests to the next POP server (on Host H).

Impact on Service

There is no impact on mail delivery. Mail retrieval may suffer slightly because there is extra load on the POP server on Host H (some client connections may be refused if the server is too busy).

Actions

1. Restart the POP server. Check the log files to see if you can find the cause of failure.
2. If it appears the server will be down for more than a few minutes, you should start up a POP Server on an alternate InterMail machine and roll the IP address for Host G to that machine.

11.2.8 IMAP Server is Unavailable

The IMAP Server handles requests for message retrieval from mail clients that support the IMAP protocol. It communicates with the Directory Cache Server to validate the user's login name and password, and to obtain the name of the MSS host on which the user's mailbox resides. The IMAP Server also communicates with the Message Store Server to service message retrieval requests on behalf of the client.

Scenario

The IMAP Sever on Host G is unavailable.

Effects

Clients connected to Host G at the time of the failure will be dropped. Round-robin DNS will send new requests for IMAP access to the next IMAP server (on Host H).

Impact on Service

There is no impact on mail delivery. Mail retrieval will suffer because there is extra load on the IMAP server on Host H (some client connections may be refused if the server is too busy).

Actions

1. Restart the IMAP server. Check the log files to see if you can find the cause of failure.
2. If it appears the server will be down for more than a few minutes, you should start up an IMAP Server on an alternate InterMail machine and roll the IP address for Host G to that machine.

11.2.9 Configuration Server is Unavailable

One host in the InterMail system is designated as the master configuration host. The Configuration Server is installed on this host. The Configuration Server maintains the master configuration database and is responsible for distributing the content of that database to all the other InterMail hosts.

Scenario

The Configuration Server is unavailable.

Effects

Servers regularly try to re-connect to the Configuration Server (every fifteen seconds). Because the Configuration Server is unavailable, these attempts fail with `ECONNREFUSED.` The servers attempting connection record this event with an entry in their log files.

The configuration editing utility, `imconfedit,` will run in local mode, meaning that it will not be able to synch up with the master configuration database; it will not be able to assess the impacts of changes, and it will not be able to propagate these changes to running servers.

Any attempted installations fail because they need the Configuration Server to be running.

Impact on Service

There is no impact on mail delivery or retrieval, however, it is more difficult to propagate configuration changes. If you could not get the Configuration Server started again and you *had* to make a configuration change, you would be required to copy the master Configuration Database by hand to each InterMail host, then re-start the servers on those hosts to force them to read the new configuration settings.

Actions

1. Go to the Configuration Server host, and as the InterMail user (imail, by default), start the Configuration Server (`lib/imservctrl start imconfserv`). Check the log file to verify that whatever brought the server down isn't preventing it from starting up again.

11.2.10 Integrated Services Directory is Unavailable

The Integrated Services Directory stores domain and account information.

The Directory Cache Server responds to requests for account information. It has access to the Integrated Services Directory, but services most requests directly from its cache, which contains a local copy of the required account information.

Scenario

The Integrated Services Directory is unavailable.

Effects

The Directory Cache Servers respond to all requests with information from their local cache files.

Impact on Service

There is no discernible impact on mail delivery or retrieval, however no account information can be added, deleted, or changed, until the database is restored to service.

Action

1. Try to bring the Integrated Services Directory up again.

Index