

A large, abstract, light gray graphic on the left side of the page, consisting of several overlapping, curved shapes that create a sense of depth and movement.

SUN JAVA™ SYSTEM MESSAGING SERVER 6.3

White Paper
April 2007

Table of Contents

Overview	1
A Design Based on Internet Standards	1
Dependability	2
High Availability	2
Security	3
Lowered Total Cost of Ownership	4
Performance and Scalability	5
Support for a Wide Range of Clients	6
Manageability	6
Extensibility	6
Architecture Details	7
Java System Messaging Server Structure	7
Provisioning and Administration	8
Message Store	9
Message Transfer Agent (MTA)	13
Directory Services	18
Message Archiving	20
Communications Express Web Client	20
Configuration and Deployment Flexibility	22
Web Access to e-Mail	22
Anti-UBE (Antispamming)	23
Throttling Incoming Connections Using MeterMaid	29
The Message Multiplexor	30
Messaging Server Secure Communications	33
Java System Messaging Server High Availability	34
Unified Messaging	37
APIs and Programmable Extensibility	38
Appendices	41
Appendix A — System Requirements	41
Appendix B — Standards Support	41
Appendix C — UM Vendors and Features	45
Appendix D — Additional Resources	46

Chapter 1

Overview

The Sun Java™ System Messaging Server is a flexible, sophisticated product designed to meet the communication needs of both service providers and enterprises. This paper explores the product's architecture, design, performance, and deployment features.

The Java System Messaging Server offers enterprises and Internet service providers (ISPs) a reliable, manageable, and cost-effective messaging solution based on Internet standards. With its wide-ranging, end-user access solutions, the Java System Messaging Server is readily available to anyone with a compliant Web browser, and also supports the popular Post Office Protocol 3 (POP3) and Internet Message Access Protocol 4 (IMAP4) clients. When employed in conjunction with the Java System Portal Server Mobile Access or third-party software, users can access their e-mail via a diverse selection of mobile devices.

The Java System Messaging Server is a component of the Sun Java™ Communications Suite, an open and integrated infrastructure software system that delivers industry-leading email, calendaring, and real-time collaboration functionality for service providers and large organizations worldwide. The Java System Messaging Server is designed to enable the rapid delivery of communication and collaboration services over converged voice, wire, and wireless networks. It integrates with complementary products in the Java Enterprise System and exposes a broad range of open, extensible interfaces that enable service providers, telecommunication companies, and enterprises to create innovative services that take advantage of today's most advanced communication technologies.

Organizations today require a dependable, cost-effective messaging system. The Java System Messaging Server fills that need with a highly scalable and complete messaging server solution that can be used effectively by both enterprises and service providers.

A Design Based on Internet Standards

The Java System Messaging Server is built on native Internet technology, so enterprises can maintain a single architecture — even when collaborating with customers and partners. Organizations are not locked into a proprietary system. All key components of the Java System Messaging Server are based on proven, open Internet standards such as:

- **Lightweight Directory Access Protocol (LDAP)** — Provides access to enterprise directory information, enabling an accurate, secure messaging system
- **Extended Messaging Transport Protocol (ESMTP)** — Affords faster delivery and message status information

- **Multipurpose Internet Mail Extensions (MIME)** — Defines a message format that enables seamless exchange of messages among a variety of messaging systems
- **Hypertext Markup Language (HTML)** — Provides Web browser access through a standard formatting language
- **Internet Message Access Protocol 4 (IMAP4)** — Delivers superior disconnected or remote user functionality
- **Post Office Protocol 3 (POP3)** — Facilitates compatibility with popular e-mail applications through an established client protocol
- **Transmission Control Protocol/Internet Protocol (TCP/IP)** — Offers universality with a worldwide networking protocol
- **Simple Authentication and Security Layer (SASL)** — Permits use of several different authentication mechanisms through an extensible, modular security interface
- **Notary** — Defines message-status and delivery notification capabilities

For a more complete list of supported standards, see Appendix B – Standards Support.

Dependability

The features and capabilities of the Java System Messaging Server enable users to increase productivity while simultaneously reducing both administrative and operational costs. The product employs committed transactions at all interfaces to prevent lost or corrupted mail messages. For example, messages are not acknowledged as received until they are committed to disk. The Message Store is built around a custom-designed database that employs a write-once data store and a two-level index to achieve excellent performance and data integrity.

Some products from other vendors improve performance by not committing data to disk, risking the loss of messages.

High Availability

The Java System Messaging Server provides a high-availability feature that supports Sun™ Cluster and VERITAS clustering solutions. This feature enables users to be serviced by a secondary Java System Messaging Server system if the primary system is off-line for maintenance or other reasons. Even when Sun Cluster software is not used, the built-in monitoring capability of the Java System Messaging Server continuously monitors the health of server processes and service availability, and can be configured to automatically restart them if necessary. All failures and recovery operations are logged for later analysis and reporting. The Java System Messaging Server also supports system monitoring through the Simple Network Management Protocol (SNMP). An SNMP client can then be used to monitor certain aspects of the messaging server.

Security

The Java System Messaging Server offers secure connections for client and administrative sessions through its Transport Layer Security (TLS) support, which enables all communication between clients and servers to take place inside an encrypted session.

The Java System Messaging Server has always supported Secure/MIME (S/MIME) when using S/MIME capable-rich clients, and now the Communications Express Webmail client also supports S/MIME. When using a supported browser such as Microsoft Internet Explorer and a keystore in the browser or in a supported Java Card™ smart card, a user can sign, encrypt, and decrypt e-mail messages. The draft of a message to be encrypted can also be saved in encrypted form. S/MIME provides end-to-end security in addition to the benefit of having the messages encrypted on disk in the Message Store.

Today, a primary threat to messaging systems is an unsolicited bulk e-mail (UBE) attack, commonly referred to as spam. UBE consumes network and computer bandwidth, as well as time that employees or users would otherwise spend on productive work. The Java System Messaging Server delivers outstanding capabilities for dealing with UBE, including out-of-the-box antirelay features. Relaying through another server is the primary method used by UBE attackers to target a site under a false identity. The Java System Messaging Server enables administrators to set up anti-UBE rules by designating source address, destination address, source IP address, and desired action. The anti-UBE capability protects an organization from UBE attacks and from unwittingly participating in UBE attacks on other sites.

In addition, the Java System Messaging Server comes ready to support Symantec Brightmail AntiSpam. Customers subscribe to Symantec's LiveUpdate of antispam and antivirus rules. This information can be delivered to the Symantec Brightmail server using several means; for example, as a mail message to a special account on the Java System Messaging Server. Because the Symantec Brightmail server software resides on the customer premises and is used by the Java System Messaging Server for spam and virus filtering, the incoming messages being filtered and scanned do not leave the customer site. Outgoing messages can also be scanned prior to being released outside the company e-mail system. Messages determined to contain spam or a virus can be discarded or sent to an administrator.

For customers using SpamAssassin software, the Java System Messaging Server can be configured to use the SpamAssassin daemon to filter out unwanted mail and remove it at a system level, or identify it so that users can take further individualized actions.

Another security problem faced by today's e-mail administrators and users is computer viruses carried by e-mail. An antivirus solution deployed on the server prevents a virus from reaching users. The Java System Messaging Server, along with antivirus software packages from third-party vendors, can protect e-mail from becoming infected. It offers levels of integration and efficiency, from the simplest command line interfaces or antivirus software packages to third-party integration with channels in the Message Transfer Agent (MTA) to the new pre-integrated support for Symantec's AntiVirus Scan Engine software.

Sun does not recommend placing a third-party Simple Mail Transfer Protocol (SMTP) server in front of the Java System Messaging Server to face Internet traffic. Instead, a third-party antivirus gateway can be invoked after the Java System Messaging Server MTA has received the message, but before it is delivered to the user. In this way, important SMTP functions are handled by the Java System Messaging Server MTA without loss of features such as Notary.

- With Symantec's AntiVirus Scan Engine, customers can filter their incoming and outgoing messages for viruses (using the latest Symantec technology) at multiple points in the messaging system. Messages can also be rejected or flagged if they fail the AntiVirus Scan Engine mail-blocking policy, which provides an alternative way to block messages based on subject, size, origin, attachment name, and attachment size.
- A messaging proxy server can also be implemented to augment data security. A proxy server placed on the firewall with the actual Java System Messaging Server behind it prevents attacks on the valuable information contained on the server.

It should also be noted that Java System Messaging Server integrates with other antivirus/antispam solutions as well as those mentioned above. For a list of pre-integrated solutions, visit the partner page at http://www.sun.com/software/products/communications/partner_library/index.xml.

Lowered Total Cost of Ownership

The Java System Messaging Server solution offers efficiencies in virtually all aspects of the messaging system to help lower total operating cost. Compared with traditional departmental mail servers such as Microsoft Exchange and Lotus Notes, the Java System Messaging Server reduces the hardware necessary per user, saving up-front costs and reducing maintenance requirements and service downtimes.

Because it is part of the Java Communications Suite, the Java System Messaging Server offers additional cost advantages. For a single price, organizations can take advantage of all the other products in the suite, such as the Java System Calendar Server, Java System Connector for Microsoft Outlook, Java System Instant Messaging, Java System Directory Server, Java System Access Manager, Java System Web Server, and more.

Performance and Scalability

The Java System Messaging Server provides both horizontal scalability, achieved through a two-tiered deployment, and vertical scalability, achieved by increasing the number of CPUs, disks, and amount of memory per system. The product can support thousands of concurrently active POP3 and IMAP4 users. In well-documented laboratory tests geared to reproducible results, the Java System Messaging Server successfully supported 125,000 concurrent, active POP3 users on a single Sun Fire™ T2000 server, which means that an ISP provisioned for a maximum of 10-percent active users can host a total of more than one million mailboxes on a single server (assuming a 10-percent concurrency rate).

The unsurpassed vertical scalability of the Java System Messaging Server is complemented by the product's horizontal scalability, which is accomplished by means of a messaging multiplexor. The implementation of a simple messaging multiplexor enables organizations to expand the capacity of a Java System Messaging Server environment by adding more servers. With multiplexors doing the work of routing the protocol traffic to and from the appropriate Message Store server, clients only need to point to a single host name that provides access to their mail. This allows capacity to be added without any burden or reconfiguration on the clients. When a user is moved from one server to another by the administrator, the user does not have to know or reconfigure the client.

The Java System Messaging Server can also take advantage of the Local Mail Transfer Protocol (LMTP) feature to provide additional scalability. In a two-tiered deployment, when LMTP is used between the tiers, the load on the Java System Directory Server is reduced by 50 percent and the disk I/O is reduced by 40 percent. The reduction in disk usage in both the back-end MTA and Message Store means that the back-end system can now serve more users with the same hardware.

By supporting distributed shared folders, the Java System Messaging Server enables users on different Message Stores to share folders with each other. Multiple Message Stores share the same directory for user information, so the user does not have to know or care where another user is in order to share a folder. The user can employ Webmail to set privileges and access for folder sharing or an IMAP client such as Mozilla™, Thunderbird, or Mulberry.

Support for a Wide Range of Clients

The Java System Messaging Server features a single Message Store for POP3 and IMAP4 environments, so organizations can have a single mail server for PC, UNIX®, and Macintosh environments. The product has been successfully tested with some of the most popular Internet mail clients, including Mozilla, Thunderbird, Microsoft Outlook, Evolution, and Qualcomm Eudora Pro.

The product also features Communications Express, by which any standard JavaScript™ technology-enabled Web browser (for example, Internet Explorer, Mozilla, Firefox, Netscape™, and Safari) can access server-based e-mail, calendar, and directory information. Communications Express can also be customized for a site-specific look and functionality.

Manageability

The Java System Messaging Server features a Java technology-based administration interface that facilitates remote administration and decreases the need for on-site operators in remote locations. User and domain administration tools are available not only through the administrative client, but also through command line utilities — a key requirement for ISPs. For example, Message Store quotas can be set for users and domains in the administrative interface. To complement that, the administrator can use the quota notification utility to inform users that their mailboxes have exceeded a set percentage of their mailbox quotas.

Extensive logging in the Java System Messaging Server offers configurable levels of detail for debugging use, error reporting, and routine actions on process start or shutdown, user login or logout, and message traffic.

Extensibility

Despite the power and flexibility built into Java System Messaging Server, there will be times when a site needs to add its own functionality to the system. For this reason, extensibility options are built in at several levels.

Arbitrary processing can be performed on any or all message parts through the use of the conversion channel. This is the route often taken for virus scanning, document conversion, content analysis, and filtering. This method typically requires no programming beyond a simple script to launch a virus scanner or other application.

In scenarios where the conversion channel does not provide adequate functionality or performance, an organization may want to write its own message processing channels. To accommodate this need, the application programming interface (API) built into the MTA provides powerful, sophisticated routines. If a full-blown channel is not required, code can be integrated into the message processing at several levels that allows for custom actions, such as address resolution.

Chapter 2 Architecture Details

Java System Messaging Server Structure

The Java System Messaging Server is an extensible framework of cooperative modules that creates an enterprise-wide, open standards-based, scalable electronic message-handling system. This system is the combination of message user and transfer agents, Message Stores, and access units that together provide electronic messaging. Figure 1 shows the high-level modular architecture of the Java System Messaging Server system.

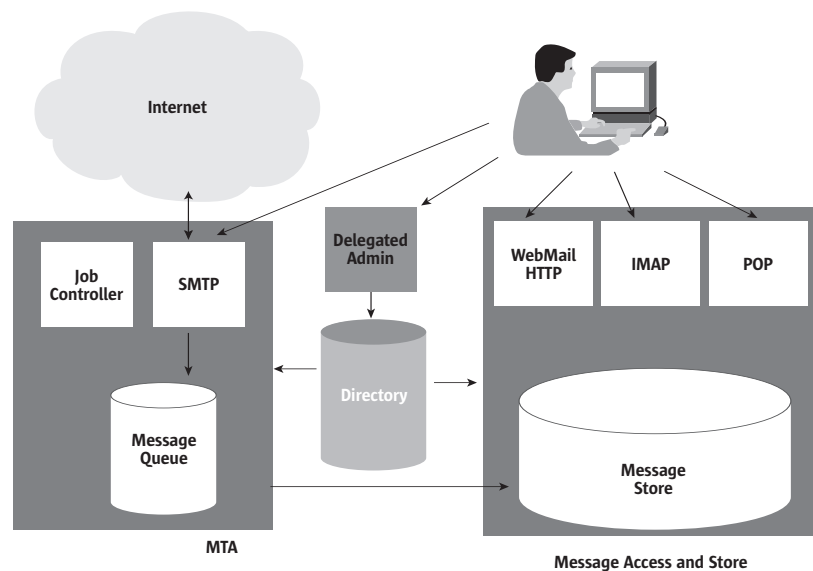


Figure 1. Java System Messaging Server Components

The components of this message-handling system include:

- **Delegated Administration Services.** Provides graphical user interface (GUI)-based and command line based user and domain administration
- **Message Access and Store.** Provides a repository of user messages; POP, IMAP, and Webmail servers retrieve and process those messages
- **Message Transfer Agent.** Responsible for routing, transfer, and delivery of Internet mail messages. The Java System Messaging Server includes a fast, scalable, and flexible MTA that replaces the Sendmail utility bundled with most UNIX systems
- **Directory Services.** The Java System Messaging Server is bundled with Java System Directory Server, the central repository for meta information including user profiles, distribution lists, and other system resources; and Java System Access Manager, which provides secure access across Intranets and Extranets.

The modular nature of the system allows for deployments that separate functions and run them on various types of hardware.

Provisioning and Administration

There are two means of provisioning users, groups, and domains for the Java System Messaging Server. The Delegated Administrator provides a GUI and a command line interface (CLI) through which administrators can provision organizations, users, groups and more. Additionally, a set of command line utilities allows full configuration and management of the server, giving system administrators the ability to script common tasks and fine-tune default configurations of various components and their interactions within the mail server.

The provisioning and administration services provided with the Java System Messaging Server consist of (see Figure 2):

- Java System Web Server¹
- Java System Access Manager
- Delegated Administrator Console
- Communication User Management (commadmin) Command Line Interface (CLI)
- Java System Directory Server²

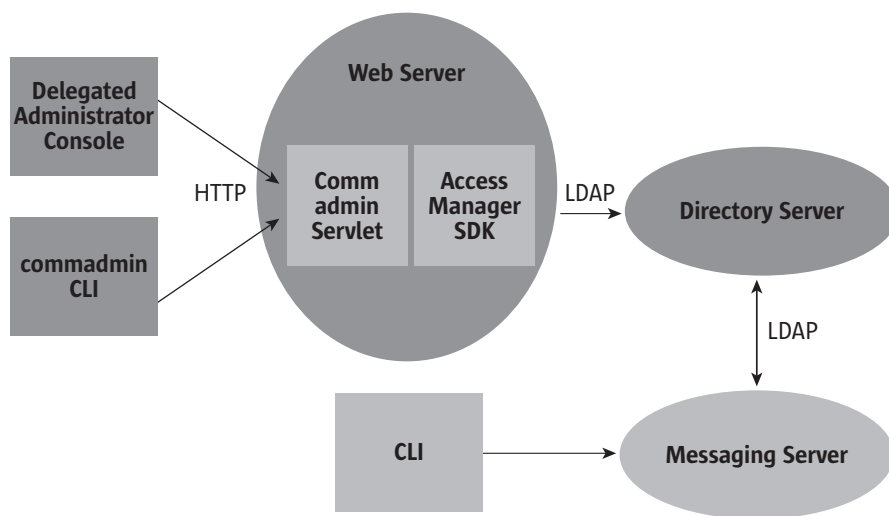


Figure 2. Java System Messaging Server Provisioning and Administration Architecture

1. The license provided with the Java System Messaging Server for the Java System Web Server is limited. The license grants the right to run the Java System Web Server to enable the administrative capabilities of the Java System Messaging Server that require the Java System Web Server, but does not grant general use.

2. The license provided with the Java System Messaging Server for the Java System Directory Server is limited. The license grants the right to run directory services to support users of the Java System Messaging Server, but does not grant general use.

The Java System Messaging Server administration and configuration is performed through a command line interface. The command line utilities enable administrators to tailor the Messaging Server configuration to environmental limitations and requirements. With the command line utilities, the administrator can configure the ports, security, alarm attributes, logging attributes, timeouts, and more.

The Delegated Administrator console provides a GUI within a Web browser to provision domains, organizations, and users. This utility is useful to service providers, but can also be employed in large enterprises where there is a need to delegate administration of users to suborganizations. The Delegated Administrator utility (commadmin) provides a CLI to the Java System Access Manager to provision users, groups, and domains for messaging. Class of services are used to manage services for users and domains. The Delegated Administrator is an administrative (not end-user) utility that can be used by several levels of administrators: top-level, service provider, and organizational unit.

Message Store

The Java System Messaging Server message store is a dedicated data store for the delivery, retrieval, and manipulation of Internet mail messages. It works with the POP3 and IMAP4 client access servers to provide flexible and easy access to messaging, as well as through the Webmail server to provide messaging capabilities to Sun Java System Communications Express (also part of the Java Communications Suite) in a Web browser.

Message Store Architecture

The Message Store is organized as a set of folders or user mailboxes; each user has an inbox where new mail arrives and may have one or more folders where mail can be stored. Folders can contain other folders arranged in a hierarchical tree. Mailboxes owned by an individual user are private folders, but can be shared at the owner's discretion with other users. Figure 3 shows the overall architecture of the Message Store.

There are two general areas in the Java System Messaging Server Message Store: one for user files and another for system files. In the user area, the location of each user's inbox is determined with a two-level hashing algorithm. Each mailbox or folder is represented by another directory in its parent folder. Each message is stored as a plain text file in the MIME format. When there are many messages in a folder, hash directories are created for that folder so that the number of message files does not place a burden on the underlying file system. In addition to the messages themselves, the Message

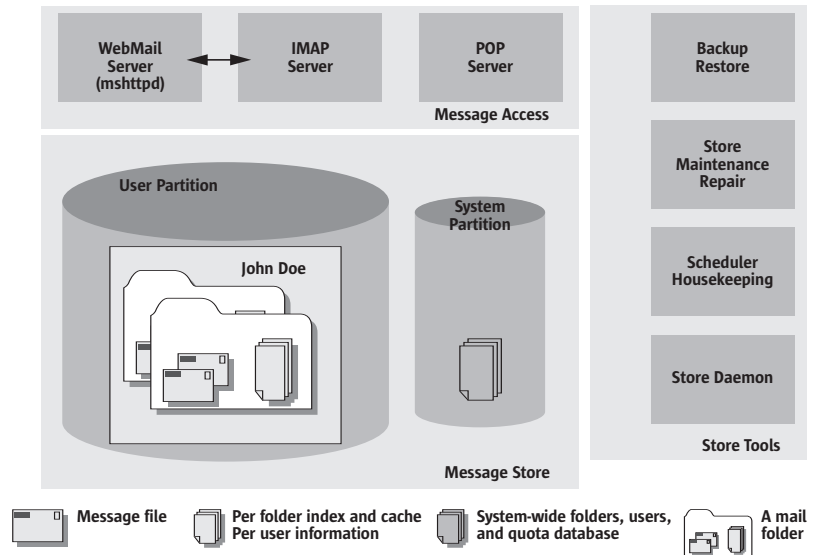


Figure 3. Message Store Architecture

Store maintains an index, a cache of message header information, and other frequently used data to enable rapid retrieval of mailbox information by clients. The Message Store can contain many partitions that are in turn contained by file systems. As a file system becomes full, the administrator can create additional file systems and Message Store partitions.

Hybrid Message Store Design

The design of the Java System Messaging Server Message Store provides unique performance and scalability improvements over other messaging systems. Older mail systems utilized a single file store for each user or a single file per message. In the single-file-per-user case, the entire mailbox must be rewritten to make changes such as deleting a single piece of mail or revising a message's status. The traditional UNIX /var/mail system works in this manner.

Other Message Store designs employ a model of one directory for all users, and consider every piece of mail as a file. Although this is an improvement, there still remains a number of limitations. File system performance goes down when there are too many entries in one directory. This problem can be remedied by creating multiple partitions and limiting the number of users to 32,000 per partition (this is not a hard limit, but merely a recommendation).

A pure database approach is also not suitable for large message stores. Mail messages vary wildly in size, making them a bad fit for storage in a database. Once received in the Message Store, a particular message is not modified; only the status of the message changes over time — for example, read, answered, or deleted.

The design employed in the Java System Messaging Server introduces the use of a hash table and structure with no more than 200 users per directory. The Message Store uses a hybrid design that combines an indexed database for storage of message header information, and flat text files for storage of message content. The use of hashed directories and a file-naming algorithm avoids problems caused by too many files in one directory.

The system area contains information on the entire Message Store in Berkeley database format for faster access. The Java System Messaging Server also has database snapshot capability, so when needed, the database can quickly be recovered to a known state.

Automatic Fast Recovery

By default, snapshots of the database are taken every 24 hours. Use of snapshots reduces the database recovery time to only minutes.

Upon starting the Java System Messaging Server Message Store, databases are checked for integrity. Most problems are automatically corrected, and logs provide comprehensive analysis and instructions to the administrator if further actions are necessary. The snapshots are used automatically when necessary, reducing the time to recover from database corruptions. That means it will take no more than a few minutes to start the service or inform an administrator. When the database is repaired, it is usually up-to-date and no further action is needed.

In rare cases, however, when it is necessary to synchronize a database from redundant data, the Java System Messaging Server offers fast recovery capability. The administrator can shut down the Message Store and bring it back immediately without having to wait for a lengthy database reconstruction. On-demand repair is performed on user folders so that mail can still be accessed without waiting for a system-wide repair to be completed.

Single-Copy Message

A notable feature of the Java System Messaging Server Message Store is that it maintains only one copy of each message per partition, sometimes referred to as a single-copy Message Store. If the Message Store receives a message addressed to multiple users, a group, or a distribution list, it adds a reference to the message rather than copying the actual message in each user's inbox, cutting down on storage of duplicate data. Individual message status (seen and deleted) is maintained per folder for each user.

Distributed Shared Folders

The Java System Messaging Server supports IMAP4 Access Control List (ACL) extensions so that the user can use any IMAP client that supports these extensions — such as Mulberry, Netscape Messenger, and Microsoft Outlook — to set access privileges for the user's folders. The Webmail client can be used for the same purpose. A user can choose to share a folder with other users on another Message Store, as long as these systems utilize the same directory server and the systems are configured as peers.

By employing a database to store the shared folders list, performance of shared folder lookup is improved — even on a large Message Store. Shared folders across multiple Message Stores are achieved via proxy from the local IMAP server to the appropriate peer IMAP server.

Public folders are owned by a special account, *public*, that is administered by the system administrator to facilitate sharing by the public. There are no owners for public folders.

Flexible Message Aging and Purging

The Java System Messaging Server Message Store supports flexible rules for aging of messages. Expired messages can be purged from the system so that they do not take up space. Rules can be defined to apply per user, per partition, per folder, or globally to the entire message store. Different rules can apply based on message age, message header, status of messages (such as read or deleted), number of messages in the folder, message type (voice, email, fax, video), size of the folder, and so on. Regular expressions can be used to specify the folders. Aging and purging operations can be invoked independent of each other. The Java System Messaging Server Message Store also supports multiple expire actions that determine what to do with the message such as delete, archive, or file into a folder.

Quota Enforcement

The Message Store supports the IMAP quota extension (Request for Comment or RFC 2087). Enforcement of the quota can be turned on or off. Administrators can configure a user's quota by the number of bytes or messages. They can also set quotas for specific folders and message types, and can set quotas for users or domains. A threshold can be set so that if it is reached, a warning message is sent to the user. When the user is over quota, new messages can be held up for retry during a grace period. After the grace period, messages sent to the over-quota user are returned to the sender with a nondelivery notification. For special applications where a quota is used but messages must be delivered regardless of the user's quota status, a guaranteed message delivery channel can deliver all messages.

Special utilities report quota usage and send over-quota warnings. When a quota is enforced, it is possible to temporarily reject incoming mail for the user or domain, providing better control of system resources.

Message Transfer Agent (MTA)

At its most basic level, the MTA is a message router. It accepts messages from other servers, reads the address, and routes it to the next server on the way to its final destination, typically a user's mailbox or message store. The MTA is also a message relay. It can relay messages to other domains and can look up the user and domain information directly from the LDAP server. This means the LDAP server becomes a critical link in the mail delivery process. The results of LDAP queries are cached in the process, with configurable size and time, so performance is tunable.

Channels

The MTA routes, transfers, and delivers Internet mail messages for the Java System Messaging Server. Mail flows through interfaces known as channels, which consist of a pair of channel programs and configuration information. Figure 4 illustrates this idea. Channels can be configured individually and mail can be directed to specific channels based on the address.

Each channel consists of up to two channel programs. Some channels have an inbound and outbound program, like the SMTP server and client channels, but most are unidirectional, like local delivery or Message Store delivery. An outgoing message queue stores messages that are destined to be sent to one or more of the interfaces associated with the channel. Channel programs perform one of two functions:

- Slave channels (marked 1 on Figure 4) accept messages from other interfaces and (2) place them in the queue for further processing by the MTA or reject them so they are not accepted onto the system.
- Master channels (3) process messages from the queue area. Then they (4) put them into the queue on the same system for further processing by another channel, (5) transmit them off the system to other interfaces and delete them from the queue after they are sent, or (6) deliver them to a final destination on the system, such as the Message Store.

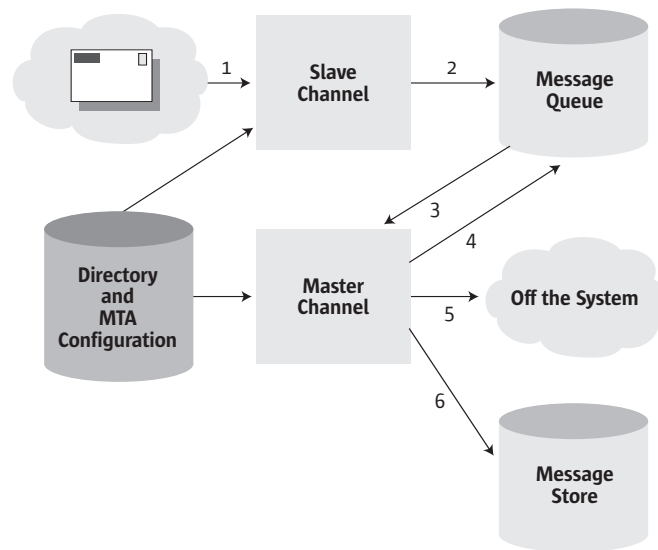


Figure 4. Channel Architecture

Using a configuration text file, administrators can configure channels with a variety of parameters to control how messages are handled. This handling includes performance tuning, as well as reporting aspects of the system. For example, multiple channels can be defined to segment traffic by groups or departments, message size limits can be defined to limit traffic, and delivery status notification rules can be defined according to the needs of the business. Diagnostic attributes are also configurable on a per-channel basis. The number of configuration parameters that can be set on a channel basis is large. For detailed information, see the *Sun Java System Messaging Server 6.3 Administration Reference* at docs.sun.com/app/docs/coll/1312.2.

Some of the default channels provided with the Java System Messaging Server include:

- **SMTP:** Used for TCP/IP-based message delivery and receipt; provides both master and slave channels
- **LMTP:** Used between the front-end MTA and the back-end Message Store to reduce processing and disk I/O for better throughput and performance; an alternative to using SMTP between them
- **Pipe:** Used for alternative message delivery programs; allows delivery of messages to programs, such as a mail sorter, rather than directly to a user's inbox
- **Local:** Delivers mail to /var/mail; provides for compatibility with older UNIX mail clients

- **Reprocessing:** Useful for messages that are resubmitted
- **Defragmentation:** Reassembles partial messages into the original, complete message to support MIME message/partial content type
- **Conversion:** Performs body-part-by-body-part conversion on messages; useful for rewriting addresses or reformatting messages
- **Message Store:** Provides for local delivery to the message store

The Java System Messaging Server runs destination addresses through domain rewriting rules, or rewrite rules for short. These Rewrite rules convert addresses into true domain addresses and determine their corresponding channels. Addresses appearing in both the transport layer and message header are rewritten according to these rules. The transport layer is the message's *envelope*, which contains routing information and is invisible to the user. It is the mechanism that actually delivers the message to the appropriate recipient.

Rewrite rules and the table of channels cooperate to determine the disposition of each address. The result of the rewrite process is a rewritten address and a *routing system*, that is, the system to which the message is sent. Depending on the network's topology, the routing system may be just the first step along the path the message takes to reach its destination, or it may be the final destination system.

After the rewrite process is finished, the channel portions of the configuration file are searched to locate the routing system. Each channel has one or more host names associated with it. The routing system name is compared with each of these names to determine which channel should receive the message. For example, a simple rewrite rule is `thor.innosoft.com $U@$D`. This rule matches addresses for the domain `thor.innosoft.com` only. Such matching addresses are rewritten with the template `$U@$D`, where `$U` indicates the user portion or left side of the address (before the `@`), and `$D` indicates the domain portion or right side of the address (after the `@`). The message is placed in the channel tagged with the `thor.innosoft.com` domain name.

Rewrite rules are very powerful and can perform complex substitutions based on mapping tables, LDAP directory lookups, and database references. While occasionally cryptic, they are useful because they operate at a low level and impose little direct overhead on the message processing cycle. For full information on these and other features available in the rewrite process, see the *Sun Java System Messaging Server 6.3 Administration Reference* at docs.sun.com/app/docs/coll/1312.2.

Job Controller

The Java System Messaging Server job controller program controls message queues and executes programs to do the actual message delivery. The job controller runs as a multithreaded process and is one of the few processes always present in the Java System Messaging Server system. The channel processing jobs are also created by the job controller, but are transient and not present when there is no work for them to do.

Figure 5 illustrates the job controller architecture. Slave channels, which respond to external stimuli, notify the job controller of a newly created message file. The job controller enters this information into its internal data structure and if necessary, creates a master channel job to process the message. This job creation may not be necessary if the job controller determines that an existing channel job can process the newly created message file. When the master channel job starts, it receives a message assignment from the job controller. When the master channel is finished with the message, it updates the job controller about the status; either the message is successfully dequeued or the message should be scheduled for retrying. The job controller maintains information about message priority and previous delivery attempts that failed, allowing for advantageous scheduling of channel jobs. The job controller also keeps track of each job's state — whether it is idle, how long it has been idle, or whether it is busy — to maintain an optimal pool of channel jobs.

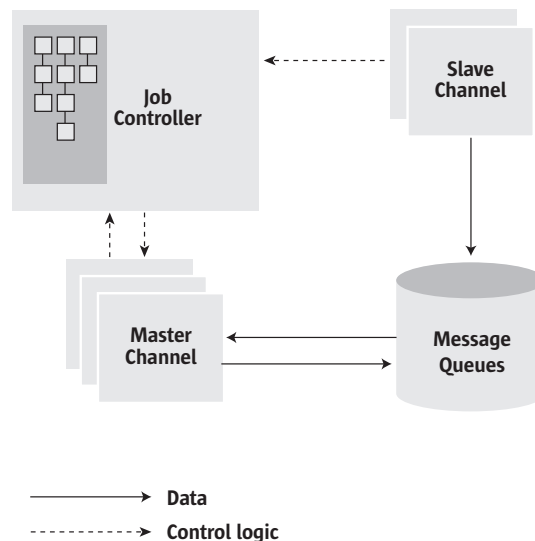


Figure 5. Job Controller Architecture

Dispatcher

The dispatcher is another process that is always present on a messaging server system. The Java System Messaging Server features a multithreaded traffic dispatcher that dispatches incoming SMTP or LMTP connections to the pool of SMTP or LMTP server threads for protocol-specific processing. The SMTP and LMTP server programs provide a pool of worker threads at the disposal of the dispatcher. After the dispatcher processes a message by either rejecting the message or enqueueing it into its destination channel, the worker thread is ready to accept more work from the dispatcher.

The dispatcher can block incoming traffic based on IP address and can throttle traffic to prevent Denial of Service (DoS) attacks. It also creates and shuts down SMTP or LMTP server processes based on load and configuration. This means the SMTP or LMTP slave channel programs are under the control of the dispatcher, not the job controller.

LMTP

In the Java System Messaging Server environment, LMTP can be configured and used in a two-tiered deployment, with MTA relays on separate systems from the message store back ends. Note this is not a general-purpose LMTP implementation and can only be used between the Java System Messaging Server MTA and the Message Store back end. This means it is useful only in a two-tiered deployment. In a small deployment with only one machine, LMTP cannot be used.

When employing SMTP and a two-tiered deployment, a simple message goes through the following steps before being delivered to the user's message store.

1. A message comes in from the Internet and is received by the SMTP server (after being dispatched by the Dispatcher)
2. The SMTP server queries the Directory Server for routing and user information and determines that the message needs to go to the Message Store back-end
3. The message is placed in the proper queue, and the job controller is notified
4. The job controller asks the SMTP client channel program to pick up the message from the message queue
5. The SMTP client sends the message to the Message Store system, where the SMTP server on that system receives it
6. The SMTP server on the Message Store system looks up the routing and user information in LDAP and decides the user is on this Message Store
7. The SMTP server queues the message for the Message Store delivery program and tells the job controller about it
8. The Message Store delivery channel program picks up the message from the queue and delivers it into the Message Store

When LMTP is used between the MTA and the Message Store on the MTA system, the SMTP client is replaced with the LMTP client. On the back end system, steps 6, 7, and 8 are eliminated, and the LMTP server replaces both the SMTP server and the Message Store channel program. The message queue is also eliminated.

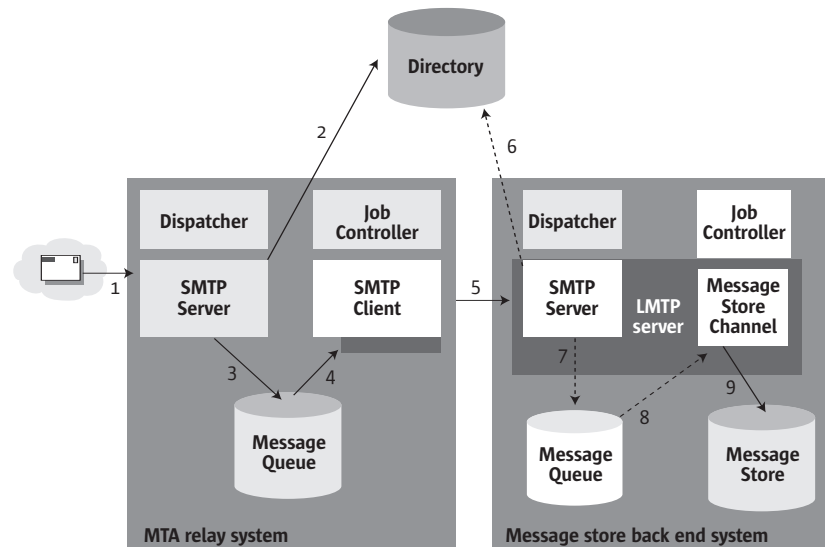


Figure 6. LMTP Architecture

By switching from SMTP to LMTP, you can eliminate 50 percent of the Directory lookup and 40 percent of the disk I/O (because there is no read/write to the queue on the Message Store system). When the message cannot be accepted by the Message Store back end, the message is queued on the MTA relay system. Message forwarding and Sieve processing are switched to be processed by the MTA relay. This does mean that the workload and disk requirement on the MTA relay is increased somewhat.

Directory Services

Java System Messaging Server is bundled with the Java System Directory Server, which is a dedicated LDAP directory service. Directory Server provides the central repository for information critical to the operation of the Messaging Server: user profiles, distribution lists, and other system resources. The Delegated Administrator can be used to provision users in the Directory Server, using the `commadmin` command line interface or the Delegated Administrator Console graphical user interface.

Directory Information Tree

The data is stored in the directory in the form of a tree, known as the directory information tree (DIT). The DIT is a hierarchical structure with one major branch at the top and many branches and subbranches below. The arrangement of the tree is flexible, so administrators can decide how best to deploy the service for their organization. For some, it may be best to arrange the tree according to the actual business organizational structure or geographic structure. For others, a one-to-one mapping to Domain Name System (DNS) layers may be best. Changing the DIT structure in a running deployment is not a trivial task, so care must be taken when designing the tree.

The DIT also provides the flexibility to support a wide range of administration scenarios. It can be administered in either a centralized or distributed manner. In centralized administration, one authority manages the entire DIT. This type of administration is usually employed in scenarios where the DIT resides on a single mail server. In distributed administration, multiple authorities manage the DIT. This type of administration is usually implemented when the DIT is divided into portions, or subtrees, residing on several mail servers.

When the DIT is large in size or when mail servers are geographically dispersed, it may be beneficial to delegate management of portions of the DIT. Typically, an authority is assigned to manage each subtree. The Java System Messaging Server allows for a single authority to manage multiple subtrees; for security reasons, however, an authority can make changes only to the DIT subtree that it owns.

When the Java System Access Manager is not used, the Java System Messaging Server uses a different default schema; the product is capable of supporting both schemas and allows for their transition and migration.

Directory Replication

The directory service supports replication, allowing for a variety of configurations to provide redundancy and efficiency. Enabling replication of all or part of the DIT from one Directory Server to one or more additional servers provides the most flexible configuration capabilities.

- Directory information is more accessible because it is replicated on multiple servers rather than on a single Directory Server.
- Directory information is cached on a local Directory Server, saving the effort of accessing information from a remote Directory Server and enhancing performance, especially in deployments with limited network bandwidth back to the central directory.
- Depending on the actual configuration, requests generated by mail clients can be processed faster by multiple Directory Servers rather than by a centralized Directory Server.

Directory replication, performance tuning, and DIT structure design are complex subjects and are well beyond the scope of this paper. For more information, see the *Sun Java System Directory Server Enterprise Edition 6.0* documentation and supporting material available online at docs.sun.com/app/docs/coll/1224.1.

Message Archiving

The Java System Messaging Server supports archiving through the Sun Compliance and Content Management Solution powered by the AXS-One Compliance Platform. Whether archiving is required for regulatory, compliance, or litigation purposes, needed to manage the growth of the message store, or used to reduce storage costs, the Sun Compliance and Content Management Solution can help achieve these goals individually or simultaneously. With the Sun Compliance and Content Management Solution, messages can be archived at different points during the e-mail's life, for example, at delivery time or at expire time. For more information on how to connect the Sun Compliance and Content Management Solution to the Java System Messaging Server, see the *Message Archiving Using the Sun Compliance and Content Management Solution* documentation at <http://docs.sun.com/app/docs/coll/1312.2>.

Communications Express Web Client

The Java System Messaging Server supports an integrated Web client also known as Java System Communications Express. This client is a component of the Java Communications Suite and integrates personal address book and calendar functionality with e-mail functionality. Communications Express replaces the Messenger Express client available in previous versions of Java System Messaging Server software. Figure 7 presents a high-level depiction of the Communications Express architecture.

- Communications Express is a single client that presents a unified user interface (UI) to the back-end Java System Messaging Server and Java System Calendar Server
- Communications Express provides access to a common address book server that is shared by the mail and calendar functions
- Communications Express uses Java Platform, Enterprise Edition (Java EE) technology such as JavaServer Pages™ (JSP™), the eXtensible Markup Language (XML), and the Java System Application Framework (JATO) for its calendar, address book, and global options interfaces
- Communications Express provides single sign-on (SSO) between e-mail, calendar, and address book, with or without Java System Access Manager
- Communications Express supports Sun LDAP schema 1, as well as schema 2
- Communications Express is deployed as a Web application using either the Java System Web Server or the Java System Application Server
- Communications Express is customizable and extensible

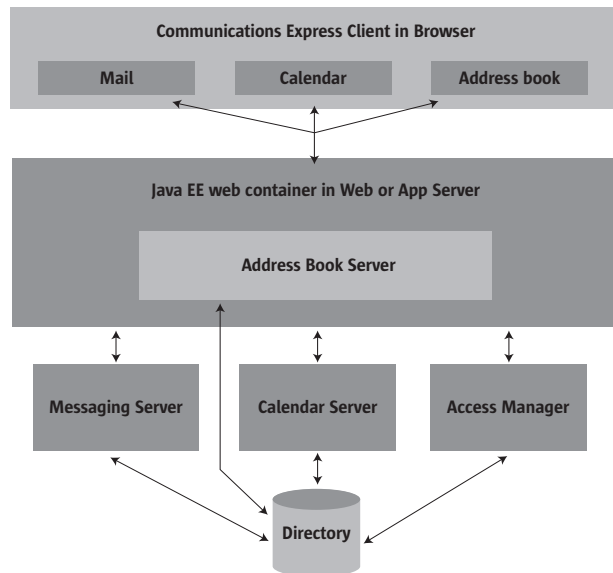


Figure 7. Communications Express Architecture

The Webmail component of Communications Express gives users full access to e-mail through a standard Web browser. The e-mail portion of Communications Express consists of two components: the client, which reads and interprets the JavaScript language, and the server, which communicates with the Message Store via IMAP. JavaScript files reside on the server and are downloaded to the client. The client extracts data from the JavaScript code to customize Communications Express functions. Communications Express has a spell checker and supports shared folders on the same back-end store. Also, messages can be composed in HTML format or plain text.

Customization

Communications Express can be substantially customized, in effect creating a unique client for an enterprise or service provider offering. All major functionality — composing, filing, reading, and so on — can be modified through the appropriate JavaScript file. Additionally, localizations can be achieved by creating equivalent pages in several languages; the Communications Express code automatically detects the client's preferred language and returns the correct page in the proper language. For more information, see the *Sun Java System Communications Express Customization Guide* at <http://docs.sun.com/app/docs/doc/819-4441>.

Chapter 3

Configuration and Deployment Flexibility

The Java System Messaging Server provides a number of significant configuration capabilities for exceptional deployment flexibility. By intelligently using Internet standards, it reliably supports most mail deployment scenarios with high performance and broad functionality. In particular, several features expand its capabilities beyond those found on other systems, including:

- Web access to e-mail
- Anti-UBE (antispamming)
- Proxy message access configurations
- Proxy server models
- Use of Messaging Server Secure Communications
- Use of High Availability

Web Access to e-Mail

Communications Express is fully integrated, browser-based software that provides access to Java System Messaging Server e-mail via the Webmail component. Communications Express is integrated with the Java System Messaging Server system and is centrally administered so installation involves little more than providing end-users with a URL for the Communications Express server. Figure 8 shows the default Communications Express interface.

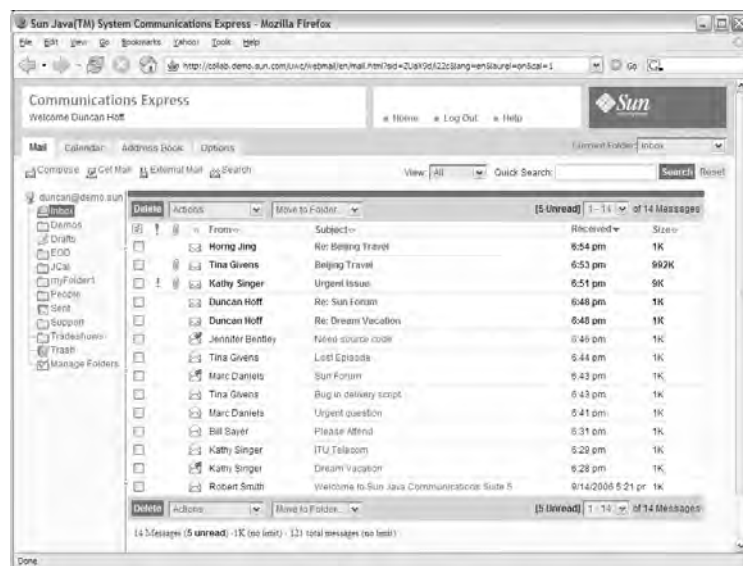


Figure 8. Communications Express

All communication with the browser is done by employing Unicode Transformation Format (UTF)-8, which displays any character set correctly. UTF-8 is also used for composing

messages. Outgoing messages encoded in the character set are determined by the user preferences as well as the characters embedded in the message; UTF-8 is utilized only when the message has a mixture of different character sets.

Anti-UBE (Antispamming)

The Java System Messaging Server provides a wealth of effective tools for dealing with unsolicited bulk e-mail, commonly referred to as spam or UBE. These tools, described in more detail below, include:

- **Access Control:** Rejects mail from known UBE sources; enables control over who can send or receive e-mail within the organization
- **Mailbox Filtering:** Allows individual users to manage their own UBE filters through a Web interface, controlling the nature of mail delivered to their mailboxes
- **Address Verification:** Refuses mail with invalid originator addresses
- **Real-time Blackhole List:** Refuses mail from recognized UBE sources as identified by the Mail Abuse Protection System's Real-time Blackhole List (MAPS RBL), a responsibly managed, dynamically updated list of known UBE sources
- **Relay Blocking:** Prevents abusers from using a mail system as a relay to send their UBE to tens of thousands of recipients
- **Authentication Service:** Enables password authentication in an SMTP server with SASL
- **Sidelining:** Silently sidelines or even deletes potential UBE messages
- **Comprehensive Tracing:** Employs reliable mechanisms for identifying a message's source
- **Conversion Channel:** Integrates with third-party anti-virus or anti-spam products
- **Milter:** Provides a plug-in interface for third-party software to validate, modify, or block messages as they pass through the MTA

These tools can be used individually or in concert. Separately, none of them block all UBE, but together, they provide an effective means of battling unauthorized use of a mail system. The solutions presented are designed to be as efficient as possible to minimize resource costs when dealing with UBE abuse.

Note: Mail traffic passing into, through, and out of the Java System Messaging Server can be separated into distinct channels according to various criteria. Source and destination e-mail address, as well as source IP address or subnet, are among these criteria. Different processing characteristics can then be ascribed to these varying mail flows, or *channels* (see the discussion of channels in the *Message Transfer Agent* section of this paper). Consequently, different access controls, mail filters, processing priorities, and tools can be applied to these channels in varying ways and combinations. For example, mail originating from within a domain can be processed differently from mail originating in the outside world. In addition to channel-based, message-flow classification, another useful classification is mailing list traffic. Traffic for a given mailing list can come into the Java System Messaging Server through a number of different channels and go back out through a number of other channels. When dealing with mailing lists, it is useful to think in terms of the lists themselves, and not in terms of channels. The Java System Messaging Server recognizes this and allows many channel-specific UBE-fighting tools to be applied in a mailing list-specific fashion.

Access Controls

A general-purpose mechanism of the Java System Messaging Server can be employed to reject mail in accordance with a variety of criteria, including the message source, destination e-mail address, and source IP address. This mechanism can be used in a variety of ways to combat UBE. For example, it can refuse mail from specific senders or entire domains (such as mail from spam@public.com). Large lists of screening information can be extended with a database that stores the access criteria. While not UBE-related, this same access control mechanism is also suitable for maintaining a database of internal users who are allowed to send mail out of certain channels. For example, you can restrict on a per-user basis who can send or receive Internet mail.

Mailbox Filtering

The Java System Messaging Server provides per-user mail filters that can be managed from Communications Express. Using these filters, users can control what mail messages are delivered to their mailboxes. For instance, a user tired of a *make money fast!* UBE can specify that any messages with such a subject be rejected. Figure 9 shows the screen used to configure such a rule in the Webmail interface.

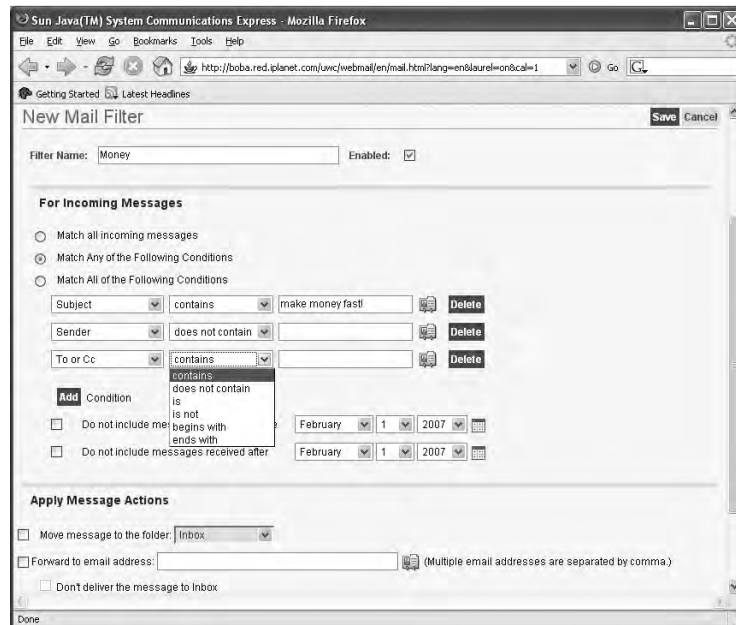


Figure 9. Editing Mail Filter Rules

Mail filtering in the Java System Messaging Server is based on the Sieve filtering language developed by members of the Internet Engineering Task Force (IETF). In addition to filtering mail by means of Sieve-based filters, sites can also implement content-based filtering or virus scanning through the use of third-party content filtering software. Software packages of this type analyze the message's content and, based on built-in or user-specified rules, decide whether the message is acceptable or not. Unacceptable messages can be bounced back to the originator, silently deleted, or held for manual inspection. These software packages are integrated into the Java System Messaging Server environment through the conversion channel, an internal mail-handling channel that can perform arbitrary processing on messages and their constituent parts. For suppliers of content-analyzing software, see the *Other Details* section of this paper.

Address Verification

UBE messages often use invalid originator addresses. The Java System Messaging Server SMTP server takes advantage of this by rejecting messages with invalid originator addresses. If the originator's address does not correspond to a valid host name as determined by a query to the Internet DNS, the message can be rejected. Note that a potential performance penalty can be incurred with such use of DNS. Address verification can be implemented on a per-channel basis, and is enabled with the **mailfromdnsverify** channel keyword described in the "*Configuring Channel Definitions*" section of the *Sun Java System Messaging Server 6.3 Administration Guide* at docs.sun.com/app/docs/coll/1312.2.

The MAPS RBL is a dynamically updated list of known UBE sources identified by source IP address. The Java System Messaging Server SMTP server supports the use of the MAPS RBL, and can reject mail coming from sources identified by the MAPS RBL as originators of UBE. The MAPS RBL is a free service provided through the Internet DNS. For more information on the MAPS RBL, see www.mail-abuse.com.

Use of the RBL by the Java System Messaging Server SMTP server is enabled with the **ENABLE_RBL** option of the MTA Dispatcher. For details, see the MTA Configuration chapter of the *Sun Java System Messaging Server 6.3 Administration Reference* at docs.sun.com/app/docs/coll/1312.2.

Relay Blocking

The features discussed so far focus on preventing users from receiving UBE. However, a comprehensive UBE prevention strategy should also include tactics for preventing abusers from relaying mail through your network to other systems.³ This is all part of the spirit of cooperation that makes the Internet useful in the first place. The Java System Messaging Server has facilities to prevent unauthorized use of a system for mail relaying.

In its simplest form, prevention is achieved by allowing local users and systems to relay mail while rejecting relay attempts from nonlocal systems. Using IP addresses is an easy and secure way to differentiate between local versus nonlocal. It is easy to determine if an IP address is part of your IP network. It is secure because IP addresses are not readily forged; the IP router between your site and the Internet rejects incoming data packets that purport to have as their source IP address an address within your network.

Preventing mail relaying is a subtle topic, and a full discussion is beyond the scope of this document. For more details, see the Mail Filtering and Access Control chapter of the *Sun Java System Messaging Server 6.3 Administration Guide* at docs.sun.com/app/docs/coll/1312.2.

Authentication Services

The Java System Messaging Server SMTP server implements the SASL protocol (RFC 2222). This can be used with popular POP and IMAP clients to provide password-based access to an SMTP server. A typical usage for SASL is to permit mail relaying for external authenticated users. This solves a common problem posed by local users who rely on ISPs when they are working at home or traveling.⁴ Such users, when connecting to a mail system, have nonlocal IP addresses. Relay blocking that takes into account only the source IP address does not permit them to relay mail. This difficulty is overcome through the use of SASL, which allows users to authenticate themselves. Once authenticated, they can relay mail.

3. Abusers who send out tens of thousands of mail messages do so by sending relatively few messages to unsuspecting, store-and-forward mail systems. Each of these few messages may have hundreds or thousands of recipient addresses. The attacked store-and-forward mail system in receipt of a message is then expected (by the abuser) to relay (resend) the message to each of its recipients — recipients that are generally in many different domains unrelated to that of the attacked mail system. In this way, the attacked mail system becomes a mail relay, and the burden of contacting thousands of individual recipient mail systems is foisted onto it by the abuser. Moreover, individual recipients are often deceived into thinking that the attacked system is responsible for the UBE.

4. POP and IMAP clients send e-mail by relaying it through a server system. That way, should the ultimate destination machine not be immediately reachable, the burden of holding onto the e-mail and periodically attempting to deliver it is put upon a presumably more capable and robust MTA rather than on the client (which may be nothing more than a laptop or PDA device).

Sidelining

The access control mechanisms discussed so far can also defer the processing of suspect messages for later manual inspection. Or, rather than sideline these messages, the mechanisms can change the destination address and, route suspect mail to a specific mailbox or simply delete it silently. This tactic is useful when UBE is being received from a known fixed origin and outright rejection may only cause the abuser to change the point of origin. Similar features are available for Java System Messaging Server mailing lists. Great care should be exercised when silently deleting mail to ensure that valid senders are not affected.

Comprehensive Tracing

The Java System Messaging Server SMTP server discovers and records crucial origination information about every incoming mail message, including source IP address and corresponding host name. All discovered information is recorded in the message's trace fields (for example, the *Received:* header line), as well as in log files. Availability of such reliable information is crucial in determining the source of UBE, which often has forged headers. Sites can use their preferred reporting tools to access this information, which is stored as plain text.

Conversion Channel

The conversion channel is a general purpose interface that can be used to invoke a script or another program to perform arbitrary body part processing of an email message. The conversion program hands off each MIME body part (not the entire message) to the program or script and can replace the body part with the output of the program or script. Conversion channels can be used to convert one file format to another (for example, text to PostScript), to convert one language to another, perform content filtering for company sensitive information, scan for viruses and replace them with something else, and more. Content-filtering software from third-party suppliers can be hooked into a messaging deployment through the conversion channel.

Using Symantec Brightmail AntiSpam, Symantec AntiVirus Scan Engine, or SpamAssassin

Channel keywords enable mail filtering using Symantec Brightmail AntiSpam, Symantec AntiVirus Scan Engine, or SpamAssassin. The administrator can configure the MTA to filter for all messages or only those going to or from certain channels, or may set the granularity at a per-user level. A user can opt for either spam or virus filtering, or both. (SpamAssassin filters only for spam; Symantec AntiVirus Scan Engine filters only for viruses).

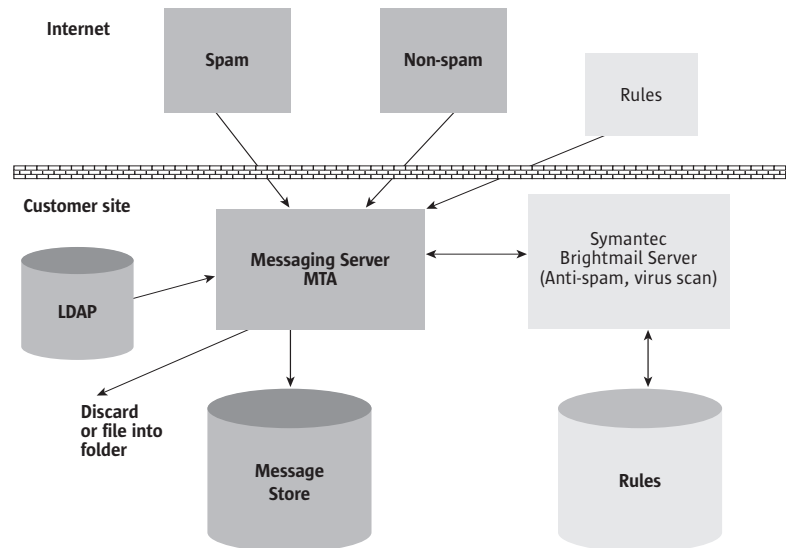


Figure 10. Symantec Brightmail AntiSpam Integration

The Java System Messaging Server's extensive Sieve support allows great flexibility to set the disposition of messages determined to be spam or viruses. The default action is to discard the offending message, or spam may be filed into a special folder. A copy of the message can be forwarded to a special account, a custom header can be added, or the Spamtest Sieve extension can be used to take other action based on a rating returned by SpamAssassin.

The Java System Messaging Server MTA can reside on the same system as the Symantec or SpamAssassin software, or it can be on a separate system. One advantage of separating the MTA from the mail filtering servers is that processing power for filtering can be increased simply by adding more hardware and cloning the servers. When the system is capable and not overloaded, the mail filtering server software can collocate with the MTA.

Milter

Milter refers to the Sendmail Content Management API and also to software written using this API. Milter provides a plug-in interface for third-party software to validate, modify, or block messages as they pass through the MTA. In sendmail, milter consists of support code in sendmail itself and a separate milter library. Filter authors link their filters against this library to produce a server. Sendmail is then configured to connect

to these milter servers. The Java System Messaging Server provides a library that emulates the sendmail side of the milter interface. Consequently, milters written for sendmail can also be used with the Java System Messaging Server. The milter server can run in a variety of configurations. It can run on a separate system of its own, on the same system as the Java System Messaging Server, in a single system deployment, or in a two-tier deployment. The Java System Messaging Server also supports connecting to multiple milter servers.

Summary

The Java System Messaging Server provides a number of features and functions that, taken together, can provide a high degree of protection from mail system abuse. Mail can be blocked at the SMTP server based on authentication of the submitter, IP address, DNS, MAPS RBL, or list of site-specified hosts. Once a message is accepted into the system, it can be filtered for content using third-party software. Users or system administrators can further filter mail using Sieve-based filters.

Note: Associated with many of these features are a myriad of fine details and additional functionality. For example, site-supplied screening code and databases can be interfaced directly to the Java System Messaging Server, and format conversions can be enabled. Particularly annoying to abusers are rejection responses from the SMTP server delayed by, for example, 15 seconds. In addition, content-filtering software from third-party suppliers can be hooked into a deployment through the Java System Messaging Server conversion channel. Some third-party anti-UBE or virus-scanning solutions to consider include:

- Symantec Brightmail AntiSpam: symantec.com
- Sophos Anti-Virus: sophos.com
- Symantec AntiVirus Scan Engine: symantec.com
- Trend Micro InterScan VirusWall: trendmicro.com
- SpamAssassin: spamassassin.org
- Cloudmark Authority: cloudmark.com

It should also be noted that Java System Messaging Server integrates with other antivirus/antispam solutions as well as those mentioned above. For a list of pre-integrated solutions, visit the partner page at http://www.sun.com/software/products/communications/partner_library/index.xml.

Throttling Incoming Connections Using MeterMaid

Sometimes, it becomes necessary to deny a connection, especially in the case of an abusive user who is attempting to deliver an excessive amount of email. In this particular case, the messaging server should respond by limiting the number of connections from the malicious user's IP address or blocking them altogether. The Java System

Messaging Server has historically provided a shared library for message throttling, `conn_throttle.so`, that used an in-memory table of incoming connections to determine when a particular IP address had recently connected too often and should be turned away for a time. While having an in-memory table increased performance, its largest cost was that each individual process on each server maintained its own table.

The Java System Messaging Server now provides MeterMaid, a repository process that replaces `conn_throttle.so`, providing similar functionality but extending it across the messaging server installation. MeterMaid represents the officer patrolling the streets, looking for those who have exceeded their allotted amount. MeterMaid is a single repository of the throttling information that can be accessed by all systems and processes within the messaging server environment. It continues to maintain an in-memory database to store this data to maximize performance.

The Message Multiplexor

Normally, the Java System Messaging Server product acts as both mail delivery server and message access server; that is, the server can handle requests to send or retrieve mail from mailboxes. The Java System Messaging Server can also be configured as a proxy message access server by using the Message Multiplexor (MMP) functionality, as shown in Figure 11.

The MMP looks exactly like a real message server to a client; however, it is only a front-end proxy to a real server. The MMP accepts POP and IMAP requests for mailbox access, authenticates the requester's password, and then forwards the request to the server containing the desired mailbox.

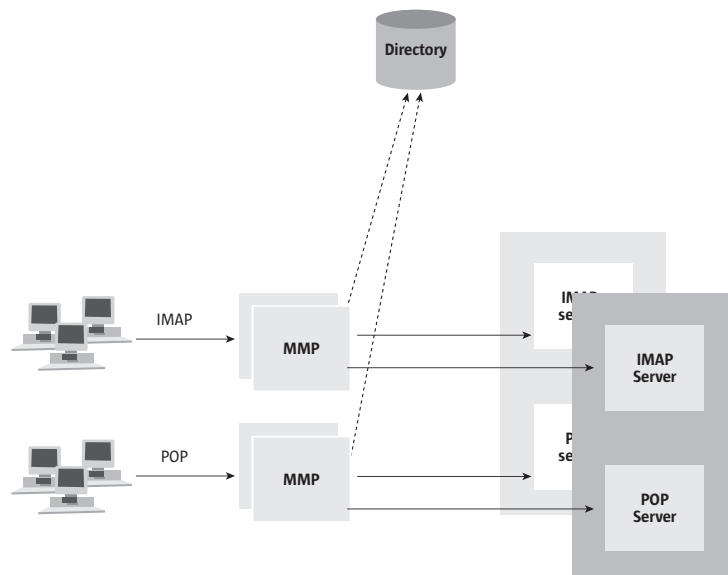


Figure 11. Message Multiplexor

The MMP uses the same LDAP directory (or a replica) to authenticate the user and find out where the user's mailbox really resides. Once authentication and connection to the real mail server has been made, the proxy acts as a simple pipe between the client and the real mail server, forwarding whatever one sends to the other until either the client or the server closes the connection.

Although a proxy server does not allow for message storage, from the client's point of view, the proxy acts just like a regular mail server. Because the proxy communicates with the real mail server by using POP and IMAP, from the back-end mail server's point of view, the proxy appears as just another client. The MMP supports SSL, so the link between the client and the MMP can be secured.

A proxy function in the Webmail server allows horizontal scalability of Webmail. It is different from the MMP proxy server, and employs HTTP to communicate with the Webmail portion of Communications Express and IMAP to communicate with the IMAP server on the back end. Webmail scalability is illustrated below. Note that the Webmail servers can be distributed throughout the network and do not need to reside on the Message Store.

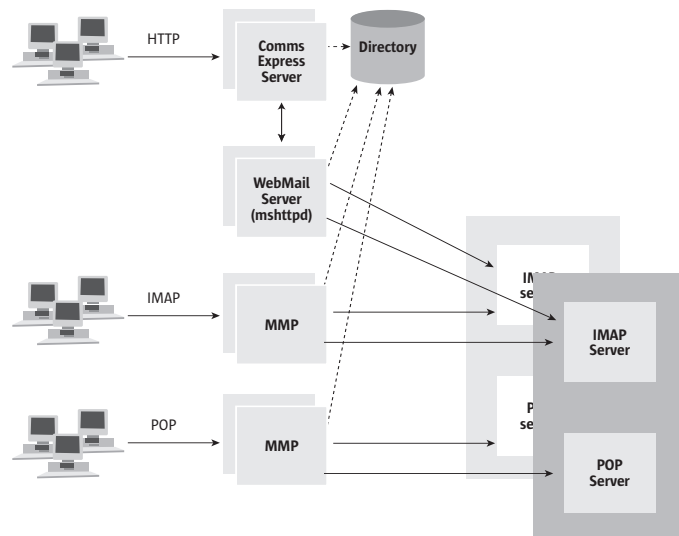


Figure 12. Message Multiplexor and Webmail Servers

MMP Server Models

MMP is useful for a number of applications. How the MMP server is deployed depends on the e-mail system's configuration and what the goals are. This section describes a possible model (see Figure 13) where MMP servers could be used. In this scenario, a load balancer routes SMTP, POP, and IMAP traffic to an MMP, which uses LDAP information to determine if the user is on a server in New York or London, and then routes traffic accordingly.

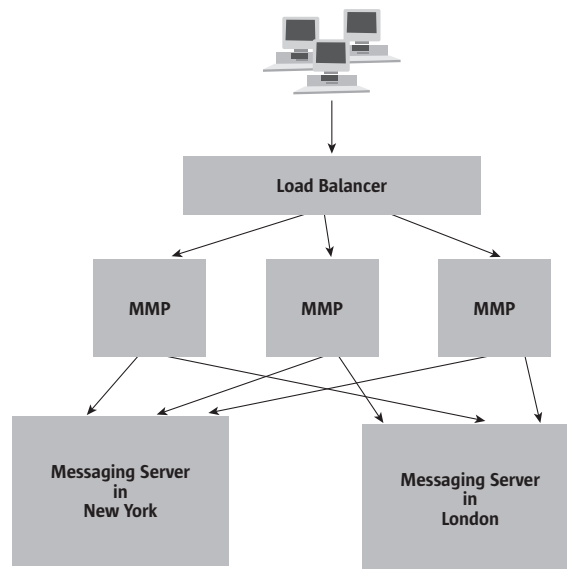


Figure 13. A Simple MMP Deployment Scenario

MMP for Horizontal Scalability

Horizontal scalability is the ability to expand the capacity of a Java System Messaging Server environment by adding more servers. MMP servers make horizontal scalability possible by having clients point to a single host name that they can use for accessing mail. MMP does the work of routing protocol traffic to and from the appropriate Message Store server. Since MMP servers allow clients to access their mail folders through a host name that is independent of the actual Message Store host name, capacity can be added without any burden of reconfiguration on clients (for example, reconfiguring the message access server on each client).

In implementations requiring multiple server systems that do not have MMP servers, users must specify a server host name to retrieve mail. By employing multiplexors, the Java System Messaging Server allows users to access messages through a single virtual mail server, while any number of actual mail servers perform actual message storage and retrieval.

By offering only one virtual mail server, ISPs and corporate administrators can add additional mailbox capacity by simply placing more servers behind the proxies. In a deployment such as this, users log into the system using the domain name mail.isp.net (for an enterprise deployment, this might be mail.enterprise.com). Mail requests are routed through the system and sent to a proxy server through a load balancer or round-robin DNS (a DNS that can return more than one IP address in round-robin fashion to distribute load among multiple proxy servers). The MMP server authenticates the user through a replicated LDAP directory and then sends a request to the appropriate message access server. Additional capacity is achieved by locating more message access servers behind the proxies.

This deployment enables easy expansion of capacity and, by virtue of load balancer or round-robin DNS, allows mail access proxies to be treated as field-replaceable units. If mail.isp.net (or mail.enterprise.com) needs to expand Message Store capacity to accommodate new customers, it can do so by either expanding the capacity of an existing Message Store server (by adding system resources) or employing an entirely new Message Store server. In either case, clients are not required to change their mail server host name settings.

Messaging Server Secure Communications

The Java System Messaging Server provides secure communications through a variety of techniques such as TLS, S/MIME, and Certificates. TLS is an open, nonproprietary security protocol that provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection between client and server or server and server. The Java System Messaging Server employs TLS to ensure security between mail client and server by encrypting the session in which e-mail content is transferred between the Java System Messaging Server and e-mail clients. MMP, SMTP, POP, and IMAP4 servers also support Start TLS commands to negotiate TLS with clients using the regular port.

Deploying a secure mail solution with the Java System Messaging Server is also possible by using the Secure/Multipurpose Internet Mail Extension (S/MIME). Communications Express mail users who are set up to use S/MIME can exchange signed or encrypted messages with other users of Communications Express, Microsoft Outlook Express, and Mozilla mail systems. S/MIME provides the following capabilities to the Communications Express user:

- Creates a digital signature for an outgoing mail message to assure the message's recipient that the message was not tampered with and is from the person who sent it
- Encrypts an outgoing mail message to prevent anyone from viewing, changing, or otherwise using the message's content before the message arrives in the recipient's mailbox
- Verifies the digital signature of an incoming signed message with a process involving a certificate revocation list (CRL), which is a list of certificates that have been revoked
- Automatically decrypts an incoming encrypted message so the recipient can read the message's content
- Exchanges signed or encrypted messages with other users of an S/MIME-compliant client such as Communications Express Mail and Mozilla mail systems

The use of S/MIME requires that at least one private and public key pair, including a certificate in standard X.509 V3 format, must be issued to each Communications Express S/MIME user. This certificate assures other mail users that the keys really belong to the person who uses them. Once key pairs and their certificates are issued, they may be stored on a smart card or in a local key store on the mail user's client machine. They are also stored in an LDAP directory so that they are available to other mail users who are creating S/MIME messages. When these requirements are met, users will be able to sign, encrypt, and decrypt email messages. The draft of the message to be encrypted can also be saved in encrypted form in the Message Store. As a result, S/MIME provides end-to-end security.

Finally, Java System Messaging Server may be used in conjunction with third-party key management products to form a secure messaging solution.

Java System Messaging Server High Availability

A major advantage of the Java System Messaging Server over competitive products is its superior scalability, which enables a large number of users to be populated on a single server. Although this provides excellent price-versus-performance advantages, it could result in a single point of failure, where one failing machine could interrupt e-mail access for an entire user community.

To ensure reliability, the Java System Messaging Server can be configured to be highly available by using clustering software. Java System Messaging Server supports both Sun Cluster and Veritas Cluster Server software. A Sun Cluster is a loosely coupled system of nodes that provides a single system image to clients of network services or applications such as Web, mail, or calendar services. A key feature of Sun Cluster software is that upon detecting a fault, it transparently relocates a service's daemons from one node to another, including host name, Internet Protocol (IP) address, and access to devices configured as part of the service, thus providing a single system image. This capability provides automatic failover when a system shutdown or failure occurs.

The Sun Java System Messaging Server supports all high-availability (HA) topologies that are supported by Sun Cluster technology in asymmetric, symmetric, and N+1 configurations. In the asymmetric configuration, each node in the cluster is a complete Solaris™ Operating System (OS) installation, with its own disk that contains the OS, Java System Messaging Server binaries, and Sun Cluster agents. Since each node has at least one network interface connected to the public network, users can connect through this interface to read their mail messages. Each node has at least two additional private network interfaces that connect to corresponding private network interfaces on the other cluster members. These interfaces are used by the Sun Cluster software on each node for system status monitoring and cluster configuration data sharing. Only one pair of private network interfaces is in use at any given time; the other is a redundant interface, so there is no single point of failure.

Each node has a connection to the disk cluster that contains the Message Store, message queues, directory contents, and configuration files. Although both nodes are continuously connected to the disk cluster, the volumes in the disk cluster are most likely mounted on only one of the nodes at any given time using HAStoragePlus technology. Sun Cluster software also includes the Sun Enterprise Volume Manager™ software, based on VERITAS Volume Manager. This product allows a logical volume to be mirrored across multiple physical volumes, providing uninterrupted service even if a physical disk fails.

Only one node in the cluster runs the Java System Messaging Server at any given time. In an asymmetric HA configuration, the configuration files, message queues, and the Message Store reside on a shared disk. As a result, when a failover occurs, the disk may be unmounted from the failing system and mounted on the surviving system. The logical IP address is then configured on the public network interface of the other system. Users and mail agents on the public network always connect by using the logical IP address. Reconnecting after a failure automatically connects to the other system. A failover appears to be a very quick crash and reboot of a single system.

Since the SMTP and POP protocols automatically connect, perform a transaction, and then disconnect, users and agents using those protocols may not even notice that a system failure has occurred. IMAP clients tend to connect and stay connected; when the failover occurs, most pop up a dialog box to inform users about the dropped connection and ask if they would like to reconnect.

In the Java System Messaging Server asymmetric HA configuration, the secondary node in the cluster is idle as far as Java System Messaging Server usage is concerned. This node, however, remains a fully functional Solaris OS system, and is available for other work as long as procedures to terminate or limit the other work after a failover are in place. When the CPU speed and memory size of the two nodes in the cluster are alike (recommended) in this asymmetric HA configuration, performance does not suffer during a failover.

In the symmetric configuration, two messaging services are configured to run, each connected with its own logical host. Normally, logical hosts are mastered by two different nodes. When one service fails, both logical hosts are mastered by the same physical node, which usually results in a degradation of performance. However, the services keep running. Put another way, in the symmetric configuration, both nodes are active and each node is typically connected to its own store and its own configuration within the shared disk. When one node fails, the load of the failed node is assumed by the surviving node and the surviving node must then handle its own load in addition to the load of the failed node. Hence, the surviving node must be capable of handling the increased load for the time it takes to replace or repair the failed node. Of course, the administrator should repair the failed node as quickly as possible.

A symmetric configuration makes sense when resources are scarce or when large servers that cannot be dedicated as backups are involved.

In the N+1 configuration, multiple messaging servers are configured to run as active nodes. Each node typically has its own configuration; there is no sharing of configuration data between these nodes although the disk itself is a shared resource. Essentially, then, each node mounts its own filesystem within the shared disk. In addition to the N active nodes, there is also a single idle node that serves as the backup node. If any of the active nodes fails, the backup node assumes the load of the failed node. Using HAStoragePlus, the failed node most likely unmounts the filesystem to which it is connected, and the standby node mounts the filesystem. Hence, the filesystem becomes a part of the resource group (that collection of resources that the cluster manages as a unit) and is mounted locally on the node that belongs to that resource group.

All Messaging Server Sun Cluster configurations support both HAStorage and HAStoragePlus resource types for making filesystems highly available within a Sun Cluster environment. HAStoragePlus is the recommended resource type since it can work with any filesystem that the Solaris OS supports (UFS, VxFS, etc) and results in significant disk-I/O performance improvement over HAStorage. Finally, Sun Cluster provides Solaris Zones support, thus enabling the administrator to install Java System Messaging Server in multiple zones within a Sun Cluster environment.

Unified Messaging

The term *unified messaging* (UM) refers to a deployment whereby users receive many types of communication — e-mail, fax, and voice mail — in a single mailbox. The Java System Messaging Server can be used effectively as a Message Store in this type of environment. Coupled with a UM product that provides telephony access, it leads to a comprehensive messaging solution.

The Java System Messaging Server is an excellent choice as a UM back-end. The highly scalable and high-performance Message Store is an ideal repository for large numbers of messages that must be accessed quickly. The Message Store provides streaming access to large messages, such as lengthy voice mails, by means of the IMAP PARTIAL command, which allows clients to stream data from the store rather than waiting for the entire attachment to be downloaded. The Sun Java System Message Queue is also key in deploying unified messaging systems. The Java System Message Queue service implements the Java Messaging Service (JMS) specification, providing a message broker, interfaces to create clients that produce or consume messages, and administrative services and control. No polling of the Message Store is necessary with JMS. Its most common use is to light a lamp on a user's phone when new voice mail arrives. The Java System Messaging Server also supports the use of the event notification service (ENS) for interoperability with other components of the Java Communications Suite. Eventually as all components of the Java Communications Suite migrate to the Java System Message Queue as their primary notification service, support for ENS will be phased out.

Figure 14 shows a high-level view of a UM deployment. An e-mail client connects to the IMAP server to access mail, just as always. The only difference in this scenario is that some of the messages may be voice mail messages, with sound files as attachments. The messages can be played back on the client system and then filed or deleted, just like e-mail messages. The telephone connects to the system through a third-party telephony interface. The specifics of this interface differ with various providers, but the basic idea is that the user is given, through the telephone, an interface to voice mail and possibly other messages as well.

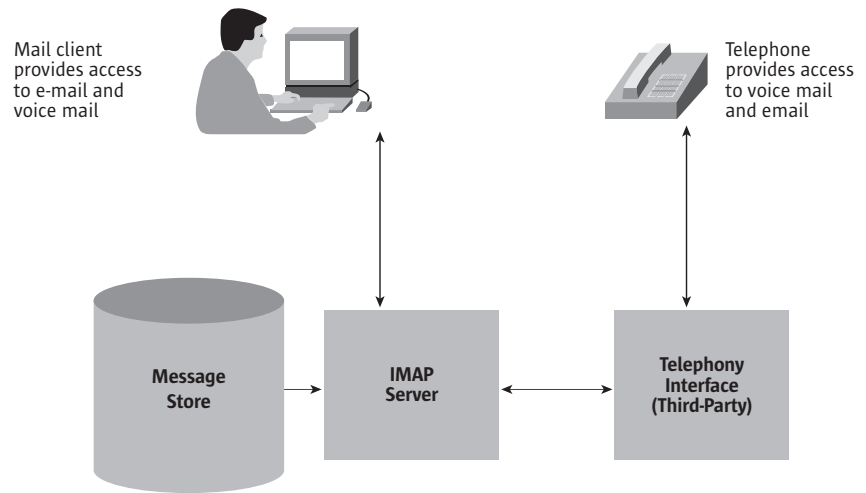


Figure 14. Unified Messaging Deployment

It is important to understand that technology such as text to speech and speech recognition is not provided with the Java System Messaging Server. However, third-party UM applications provide such technology.

Third parties that have worked with Sun to ensure that their UM products work well with the Java System Messaging Server include:

- UTStarcom — utstarcom.com
- LogicaCMG — logicacmg.com
- Lucent — lucent.com
- Mobeon — mobeon.com

A summary of the current features provided by UM partners is listed in Appendix C — *UM Vendors and Features*. Check with each vendor for specific details and availability.

APIs and Programmable Extensibility

The Java System Messaging Server provides APIs at several levels to allow for extensibility and modification of the base system.

Message Transfer Agent

The MTA can be extended programmatically through several access points: High-level API, low-level API, and callout hooks.

With the high-level MTA API, known as *callable send*, sites can easily enable an existing application for messaging. Through the use of a single API call, applications can send MIME messages complete with attachments and all the rich messaging semantics to which experienced messaging users are accustomed.

If functionality beyond the basic sending of messages is required, a low-level API can be used that exposes much of the MTA message processing machinery, providing access to:

- A powerful MIME-parsing engine
- Address handling and rewriting capabilities
- Mapping tables
- Database routines
- Message submission and delivery
- Log file processing

This API enables sites to write custom MTA channels that interface with proprietary e-mail systems or add value to messaging service offerings, for example, a channel that scans messages for objectionable or illegal content.

Sites can also extend the functionality of the MTA by writing code that is called at various hook-in points:

- **Rewrite Rules** allow additional processing of addresses beyond the MTA's already extensive abilities.
- **Mapping Tables** allow specialized code to interact with the string mapping capabilities of the Java System Messaging Server. Sites have used this capability to develop intelligent, DoS attack prevention software.
- **Alias Files** allow sites to reference existing databases or files containing mail alias information.

Message Store

The Message Store is accessible through the Internet-standard IMAP interface (RFC 2060 and RFC 3051). This conformance to open standards allows the development of IMAP standard-conformant applications that work with the Message Store. With the rich command set available in IMAP, applications can perform automated reporting, searching, and analysis of the Message Store. No other API is published, and direct access to the Message Store is discouraged.

Java System Message Queue and Event Notification Server

An administrator or developer can configure Java System Messaging Server to deliver notifications to two different messaging services: Sun Java System Message Queue and the Event Notification Service. The Java System Message Queue service implements the Java Messaging Service (JMS) specification. The JMS API is a messaging standard that allows application components based on the Java Platform, Enterprise Edition (Java EE) to create, send, receive, and read messages. It enables distributed communication that is loosely coupled, reliable, and asynchronous.

The Event Notification Service is a component bundled with the Java System Messaging Server and Sun Java System Calendar Server. It is a proprietary service that uses a publish-and-subscribe model for sending and receiving event notifications. Like JMS, clients do not need to constantly poll the Message Store. ENS is useful for integrating the messaging server notification capability with other components of the Java Communications Suite that also support ENS. However, when given the choice between ENS and the Java System Message Queue for implementing new notifications, the developer should use the Java System Message Queue notification capability rather than ENS, as it provides a more standard, flexible, and reliable approach to event notification than does ENS. Other benefits of Java System Message Queue include:

- Users can produce messages to a topic or a queue or to both of these delivery methods. Anyone subscribing to the topic will receive the message. If no one subscribes to the topic, the message is discarded. Messages delivered to a queue remain in the queue until either the message times out or a consumer retrieves the message. Only one subscriber can receive the message even if there are multiple consumers waiting for messages in the queue
- Java System Message Queue offers enhanced load balancing during message distribution, especially when messages are produced to a queue
- Java System Message Queue supports multiple notification plug-ins that can produce messages to a topic, a queue, to the Event Notification Service, and so on
- Java System Message Queue provides a reliable notification delivery mechanism. The queue can be configured to be persistent such that if a server goes down, the messages can be retrieved and made available to the appropriate consumer

Java System Directory Server

The Java System Directory Server is accessible through standard LDAP calls. Detailed information is available in the *Sun Java System Directory Server Enterprise Edition 6.0* documentation at docs.sun.com/app/docs/coll/1224.1.

Java System Access Manager

The Java System Access Manager is customizable. Detailed information is available in the *Sun Java System Access Manager 7.1* documentation at docs.sun.com/app/docs/coll/1292.2.

Chapter 4

Appendices

Appendix A — System Requirements

Mail server configurations can vary significantly depending on user requirements.

Supported platforms include:

- Solaris 10 and 9 Operating Systems on SPARC™ platforms
- Solaris 10 and 9 Operating Systems on x86 platforms
- Red Hat Enterprise Linux Advanced Server 3 and 4

Disk requirements are entirely dependent on the size of the Message Store deployed. Similarly, memory requirements are tied to usage patterns. Routing servers and POP deployments do not require huge amounts of memory. IMAP and Webmail deployments require much more memory, depending on the number of simultaneous connections supported. In all cases, disk I/O performance is crucial.

Appendix B — Standards Support

Basic Message Structure

- 0822 Standard for the Format of ARPA Internet Text Messages D. Crocker, August 13, 1982. IETF Standard #11 STANDARD (Obsoletes RFC 0733; updated by RFC 1123, RFC 1138, RFC 1148, RFC 1327, and RFC 2156).
- 1123 Requirements for Internet Hosts — Application and Support R.T. Braden, October 1, 1989. IETF Standard #3 STANDARD (Updates RFC 0822; updated by RFC 2181).
- 2822 Internet Message Format P. Resnick, April 2001.

Access Protocols and Message Store

- 1730 Internet Message Access Protocol — Version 4
M. Crispin, December 1994. Proposed (Obsoleted by RFC 2060 and RFC 2061).
- 1731 IMAP4 Authentication Mechanisms
J. Myers, December 1994. Proposed.
- 1939 Post Office Protocol — Version 3
J. Myers and M. Rose, May 1996. IETF Standard #53 STANDARD (Obsoletes RFC 1725; updated by RFC 1957 and RFC 2449).
- 1957 Some Observations on Implementations of the Post Office Protocol (POP3)
R. Nelson, June 1996.
- 2060 Internet Message Access Protocol — Version 4rev1
M. Crispin, December 1996. Proposed (Obsoletes RFC 1730).

- 2061 IMAP4 Compatibility with IMAP2bis
M. Crispin, December 1996. Informational (Obsoletes RFC 1730).
- 2062 Internet Message Access Protocol — Obsolete Syntax
M. Crispin, December 1996. Proposed.
- 2086 IMAP4 ACL Extension
J. Myers, January 1997. Proposed.
- 2087 IMAP4 QUOTA Extension
J. Myers, January 1997. Proposed.
- 2088 IMAP4 Nonsynchronizing Literals
J. Myers, January 1997. Proposed.
- 2180 IMAP4 Multiaccessed Mailbox Practice
M. Gahrns, July 1997. Informational.
- 2342 IMAP4 Namespace
M. Gahrns and C. Newman, May 1998. Proposed.
- 2359 IMAP4 UIDPLUS Extension
J. Myers, June 1998. Proposed.
- 2449 POP3 Extension Mechanism
R. Gellens, C. Newman, and L. Lundblade, November 1998. Proposed (Updates RFC 1939; Note: MMP does not support RFC 2449).
- 2683 IMAP4 Implementation Recommendations
B. Leiba, September 1999. Informational (Note: The Java System Messaging Server also supports experimental CHILDREN and LANGUAGE extensions).
- 3501 Internet Message Access Protocol — Version 4 rev1
M. Crispin, March 2003.
- 3516 IMAP4 Binary Content Extension
L. Nerenberg, April 2003.

SMTP and Extended SMTP

- 0821 Simple Mail Transfer Protocol
J. Postel, August 1, 1982. IETF Standard #10 STANDARD (Obsoletes RFC 0788; Note: The Java System Messaging Server suppresses duplicates but uses a better method than the suggestion in RFC 1047).
- 0974 Mail Routing and the Domain System
C. Partridge, January 1, 1986. IETF Standard #14 STANDARD.
- 1123 Requirements for Internet Hosts — Application and Support
R.T. Braden, October 1, 1989. IETF Standard #3 STANDARD (Updates RFC 0822; updated by RFC 2181).

- 1428 Transition of Internet Mail from Just-Send-8 to 8 bit-SMTP/MIME
G. Vaudreuil, February 1993. Informational.
- 1652 SMTP Service Extension for 8 bit-MIME Transport
J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker, July 1994. Draft (Obsoletes RFC 1426).
- 1869 SMTP Service Extensions
J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker, November 1995. IETF Standard #10 STANDARD (Obsoletes RFC 1651).
- 1870 SMTP Service Extension for Message Size Declaration
J. Klensin, N. Freed, and K. Moore, November 1995. IETF Standard #10 STANDARD (Obsoletes RFC 1653).
- 1893 Enhanced Mail System Status Codes
G. Vaudreuil, January 1996. Proposed.
- 1985 SMTP Service Extension for Remote Message Queue Starting
J. De Winter, August 1996. Proposed.
- 2034 SMTP Service Extension for Returning Enhanced Error Codes
N. Freed, October 1996. Proposed.
- 2442 The Batch SMTP Media Type
N. Freed, D. Newman, J. Belissent, and M. Hoy, November 1998. Informational.
- 2476 Message Submission
R. Gellens, J. Klensin, December 1998. Proposed.
- 2821 Simple Mail Transfer Protocol
J. Klensin, April 2001. Proposed (Obsoletes RFC 821, RFC 974, and RFC 1869; updates RFC 1123).
- 2920 SMTP Service Extension for Command Pipelining
N. Freed, September 2000. IETF Standard #60 STANDARD (Obsoletes RFC 2197).
- 3028 Sieve: A Mail Filtering Language
T. Showalter, January 2001. Proposed.
- 3207 SMTP Service Extension for Secure SMTP Over Transport Layer Security
P. Hoffman, February 2002.
- 3431 Sieve Extension: Relational Tests
W. Segmuller, December 2002.

Delivery Status Notifications

- 1891 SMTP Service Extension for Delivery Status Notifications
K. Moore, January 1996. Proposed.
- 1892 The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages
G. Vaudreuil, January 1996. Proposed.
- 1894 An Extensible Message Format for Delivery Status Notifications
K. Moore and G. Vaudreuil, January 1996. Proposed (Updated by RFC 2852).

Message Content and Structure

Note — RFC 1341 is obsolete. RFC 1524 is for mailcap, which the server itself does not use.

- 1847 Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted
J. Galvin, S. Murphy, S. Crocker, and N. Freed, October 1995. Proposed.
- 2017 Definition of the URL MIME External Body Access Type
N. Freed, K. Moore, and A. Cargille, October 1996. Proposed.
- 2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
N. Freed and N. Borenstein, November 1996. Draft (Obsoletes RFC 1521, RFC 1522, and RFC 1590; updated by RFC 2184 and RFC 2231).
- 2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types
N. Freed and N. Borenstein, November 1996. Draft (Obsoletes RFC 1521, RFC 1522, and RFC 1590; updated by RFC2646).
- 2047 Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text
K. Moore, November 1996. Draft (Obsoletes RFC 1521, RFC 1522, and RFC 1590; updated by RFC 2184 and RFC 2231).
- 2048 Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures
N. Freed, J. Klensin, and J. Postel, November 1996. IETF BCP #13 Best Current Practice (Obsoletes RFC 1521, RFC 1522, and RFC 1590; updated by RFC 3023).
- 2049 Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples
N. Freed and N. Borenstein, November 1996. Draft (Obsoletes RFC 1521, RFC 1522, and RFC 1590).
- 2231 MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations
N. Freed and K. Moore, November 1997. Proposed (Obsoletes RFC 2184; updates RFC 2045, RFC 2047, and RFC 2183).
- 2298 An Extensible Message Format for Message Disposition Notifications
R. Fajman, March 1998.

Security

- 1731 IMAP4 Authentication Mechanisms
J. Myers, December 1994. Proposed (Note: The Java System Messaging Server supports the optional APOP security mechanism defined in RFC 1939).
- 2195 IMAP/POP AUTHorize Extension for Simple Challenge/Response
J. Klensin, R. Catoe, and P. Krumviede, September 1997. Proposed (Obsoletes RFC 2095).

- 2222 Simple Authentication and Security Layer (SASL)
J. Myers, October 1997. Proposed (Updated by RFC 2444; Note: The Java System Messaging Server supports the EXTERNAL mechanism for TLS client certificates in RFC 2222).
- 2246 The TLS Protocol Version 1.0 T. Dierks, C. Allen, January 1999. Proposed.
- 2487 SMTP Service Extension for Secure SMTP over TLS
P. Hoffman, January 1999. Proposed.
- 2505 Antispam Recommendations for SMTP MTAs
G. Lindberg, February 1999. IETF BCP #30 Best Current Practice.
- 2554 SMTP Service Extension for Authentication
- 2595 Using TLS with IMAP, POP3, and ACAP
C. Newman, June 1999. Proposed (Note: MMP does not support proxy auth facility in
- 2831 Using Digest Authentication as an SASL Mechanism
P. Leach and C. Newman, May 2000. Proposed (Note: MMP does not support Digest-MD5 (RFC 2831); directory standards compliance information can be obtained from the Java System Directory Server product team).

Monitoring

- 2789 Mail Monitoring MIB
N. Freed, S. Kille, March 2000. Proposed (Obsoletes RFC 2249 and RFC 1566).
- 2788 Network Services Monitoring MIB
N. Freed and S. Kille, March 2000. Proposed (Obsoletes RFC 2248 and RFC 1565).

Appendix C — UM Vendors and Features

LogicaCMG (uONE)

- Service provider-ready solution
- Relies on Voice over IP (Cisco proprietary)
- Voice mail, e-mail, and fax support
- Multiline and family account support
- Video mail (Optional)
- Distribution lists
- Supports call waiting indicators (uNotify)
- Text to Speech (TTS) support: L&H, SpeechWorks, and Nuance
- Speech recognition (ASR): Nuance
- Voice portal: E-mail and Voice eXtensible Markup Language (vXML) browser
- Integration: Java System Messaging Server and Java System Directory Server

Lucent EBS

- IP-based Centrex, PBX business solution for telcos and large businesses
- Integrates with the Java System Portal Server, Java System Calendar Server, Java System Messaging Server, and Java System Directory Server
- Instant messaging (IM)

Lucent AnyPath

- UM solution for service providers (Mostly telco)
- Voice mail, e-mail, and fax support
- Supports external IMAP and POP e-mail accounts
- Proprietary voice Message Store
- Integrated Java System Messaging Server for regular e-mail
- Integrated Java System Portal Server 6.0
- Mobile Wireless Access Protocol (WAP) support via Java System Portal Server Mobile Access
- Direct Public Switched Telephone Network (PSTN)
- Address book: Speed dial, global, and personal

Mobeon

- Voice mail, e-mail, and fax support
- Original Equipment Manufacturer (OEM) Reseller of the Java System Messaging Server and Java System Directory Server 5.1
- Short Message Service (SMS) and Mail Management System (MMS) (Options)

Appendix D — Additional Resources

The complete Sun Java System Messaging Server 6.3 documentation set is available online at docs.sun.com/app/docs/coll/1312.2.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com



© 2007 Sun, Sun Microsystems, the Sun logo, Java, Java Card, JavaScript, JavaServer Pages, JSP, J2EE, Solaris, and Sun Enterprise Volume Manager are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Mozilla and Netscape are trademarks or registered trademarks of Netscape Communications Corporation in the United States and other countries. All rights reserved.
SunWIN# 500022 Lit.#SWWP12522-0 04/07