Oracle® Communications Messaging Server

Security Guide

Release 8.1.0

F15149-02

July 2020

ORACLE®

Oracle Communications Messaging Server Security Guide, Release 8.1.0

F15149-02

Contents

Preface

This guide provides guidelines and recommendations for setting up Oracle Communications Messaging Server in a secure configuration.

Audience

This document is intended for system administrators or software technicians who work with Messaging Server. This guide assumes you are familiar with the following topics:

- Messaging protocols, such as IMAP and SMTP
- Oracle Directory Server Enterprise Edition and LDAP
- System administration and networking

Related Documents

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

- *Messaging Server Installation and Configuration Guide*: Provides instructions for installing and configuring Messaging Server.
- *Messaging Server Installation and Configuration Guide for Cassandra Message Store*: Provides instructions for installing and configuring the Cassandra message store.
- *Messaging Server Reference*: Provides additional information for using and configuring Messaging Server.
- *Messaging Server Release Notes*: Describes the fixes, known issues, troubleshooting tips, and required third-party products and licensing.
- *Messaging Server System Administrator's Guide*: Provides instructions for administering Messaging Server.
- *Messaging Server MTA Developer's Reference*: Describes the Messaging Server MTA SDK.

Documentation Accessibility

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

Access to Oracle Support

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

1 Messaging Server Security Overview

This chapter provides an overview of Oracle Communications Messaging Server security.

Basic Security Considerations

The following principles are fundamental to using any application securely:

1. **Keep software up to date.** This includes the latest product release and any patches that apply to it.
2. **Limit privileges as much as possible.** Users should only be given the necessary access to perform their work. User privileges should be reviewed periodically to determine relevance to current work requirements.
3. **Monitor system activity.** Establish who should access which system components, how often they should be accessed, and who should monitor those components.
4. **Install software securely.** For example, use firewalls, secure protocols (such as SSL), and secure passwords. See "Performing a Secure Messaging Server Installation" for more information.
5. **Learn about and use Messaging Server security features.** See "Implementing Messaging Server Security" for more information.
6. **Use secure development practices.** This applies to customers adding plug-ins or custom code to a Messaging Server deployment. For example, take advantage of existing security functionality instead of creating your own application security.
7. **Keep up to date on security information.** Oracle regularly issues security-related patch updates and security alerts. You must install all security patches as soon as possible. See *Critical Patch Updates and Security Alerts* on the Oracle website at:

   `https://www.oracle.com/security-alerts/`

Understanding the Messaging Server Environment

To better understand your security needs, ask yourself the following questions:

1. **Which resources am I protecting?** In a Messaging Server production environment, consider which of the following resources you want to protect and what level of security you must provide:
   - Messaging Server front-end servers
   - Messaging Server back-end servers
   - Dependent resources

2. **From whom am I protecting the resources?** In general, resources must be protected from everyone on the Internet. But should the Messaging Server deployment be protected from employees on the intranet in your enterprise? Should your employees have access to all resources within the environment? Should the system administrators have access to all resources? Should the system administrators be able to access all data? You might consider giving access to highly confidential data or strategic resources to only a few well trusted system administrators. On the other hand, perhaps it would be best to allow no system administrators access to the data or resources.
3. **What will happen if the protections on strategic resources fail?** In some cases, a fault in your security scheme is easily detected and considered nothing more than an inconvenience. In other cases, a fault might cause great damage to companies or individual clients that use Messaging Server. Understanding the security ramifications of each resource helps you protect it properly.

Overview of Messaging Server Security

You manage security for a Messaging Server deployment by taking a **defense in depth** approach. By individually securing the network, hardware platform, operating system, and applications themselves, you make each layer of the architecture secure. Security includes hardening each layer by closing unnecessary network ports and access mechanisms. You also minimize the number of installed software packages so that only those packages required by the system are available. Finally, you secure and insulate the layers from unintended access within the network.

You can implement a Messaging Server proxy server to augment data security. A proxy server placed on the firewall with the Messaging Server behind it prevents attacks on the information on the Messaging Server.

Note: To ensure a completely secure environment, the deployment needs a time server to synchronize the internal clocks of the hosts being secured.

For more information on Messaging Server security, see "Creating a Security Strategy".

Understanding Security Misconceptions

This section describes common misconceptions that are counterproductive to the security needs of your deployment.

- **Hiding Product Names and Versions**. At best, hiding product names and versions hinders casual attackers. At worst, it gives a false sense of security that might cause your administrators to become less diligent about tracking real security problems. In fact, removing product information and version numbers makes it more difficult for the vendor support organization to validate software problems because of their software or of other software. Hackers have little reason to be selective, particularly if there is a known vulnerability in SMTP servers, where they may attempt to access any SMTP server.
- **Hiding names of Internal Machines**. Hiding internal IP addresses and machine names will make it more difficult to:
  - Trace abuse or spam
  - Diagnose mail system configuration errors

- Diagnose DNS configuration errors

  A determined attacker will have no problem discovering the machine names and IP addresses of machines once they find a way to compromise a network.

- **Turning off EHLO on the SMTP Server**. Without EHLO you also lose:
  - NOTARY
  - TLS negotiation
  - Preemptive controls on message sizes

  With EHLO, the remote SMTP client determines if you have a limit and stops trying to send a message that exceeds the limit as soon as it sees this response. But, if must use HELO (because EHLO is turned off), the sending SMTP server sends the entire message data, then finds out that the message has been rejected because the message size exceeds the limits. Consequently, you are left with wasted processing cycles and disk space.

- **Network Address Translation**. If you use NAT to provide a type of firewall, you do not have an end-to-end connection between your systems. Instead, you have a third node which stands in the middle. This NAT system acts as a middleman, causing a potential security hole.

Other Security Resources

For more information on designing a secure Messaging Server deployment, review the Computer Emergency Response Team (CERT) Coordination Center website:

https://www.sei.cmu.edu/about/divisions/cert/index.cfm

Recommended Deployment Topologies

You can deploy Messaging Server on a single host or on multiple hosts, splitting up the components into multiple front-end Messaging Server hosts and multiple back-end hosts.

The general architectural recommendation is to use the well-known and generally accepted Internet-Firewall-DMZ-Firewall-Intranet architecture. For more information on addressing network infrastructure concerns, see the discussion about determining your network infrastructure in *Messaging Server Installation and Configuration Guide*.

The following guidelines provide specific recommendations for Messaging Server:

- Securing Your Firewall/DMZ Architecture
- Using a Firewall to Allow Connections
- Planning Secure High Availability and Load Balancing for Your Deployment

Securing Your Firewall/DMZ Architecture

Secure your Messaging Server infrastructure by determining your Firewall/DMZ architecture. The following topics cover securing your Messaging Server infrastructure:

- The planning of your network infrastructure layout in *Messaging Server Installation and Configuration Guide*.

- The benefits of separating your network into two tiers: the public (user-facing) network, and the private (data center) network in *Messaging Server System Administrator's Guide*.
- The use of MTAs to protect your Messaging Server deployment and to control the flow of message traffic to and from your site in *Messaging Server Installation and Configuration Guide*.
- Network Security
- The two-tiered messaging architecture design that distributes hardware and software resources optimally for Messaging Server in *Messaging Server Installation and Configuration Guide*.

Note: Your firewall/DMZ architecture solution might depend on your anti-spam solution and client capabilities. How you handle firewall and DMZ architecture depends on requirements for a geographically dispersed deployment and whether your deployment is targeted at individual end users or enterprises.

Using a Firewall to Allow Connections

Because the Webmail server (**mshttpd**) supports both unencrypted and encrypted (SSL) communication with mail clients, you might use a firewall between your Messaging Store and your mail clients for added security.

Some guidelines to consider:

- If using a firewall, only allow Convergence server to connect to **mshttpd** (8990, 8991).
- If using a firewall (preferably whitelist-based for Messaging Servers), verify internal service protocols are blocked (watcher 49994, job_controller 27442, ENS 7997, third-party authentication server, msadmind 7633, LMTP, Metermaid, and JMS).

Planning Secure High Availability and Load Balancing for Your Deployment

The following topics describe how to set up a secure high availability and load balancing Messaging Server deployment:

- Designing for service availability in *Messaging Server Installation and Configuration Guide*.
- Configuring Messaging Server for high availability in *Messaging Server Installation and Configuration Guide*.
- Availability planning for Cassandra message store in *Messaging Server Installation and Configuration Guide for Cassandra Message Store*

Operating System Security

This section lists Messaging Server-specific OS security configurations. This section applies to all supported OSs.

Minimizing Operating System Security Risks

In particular, pay attention to:

- OS hardening, turning off unused OS services (especially in Linux)
- OS minimization, using minimal OS packages

Firewall Port Configuration

Messaging Server communicates with various components on specific ports. Depending on your deployment and use of a firewall, you might need to ensure that the firewalls are configured to manage traffic for each component.

Table 1–1 shows the default port numbers for various components.

Table 1–1 Port Configuration

Component	Default Port Number
SMTP	25
POP3 or MMP POP3 Proxy	110
IMAP4 or MMP IMAP Proxy	143
LMTP	225
LDAP	389
SMTP SUBMIT	587
Event Notification Service (ENS)	7997
MSHTTPD	8990
Job Controller	27442
Watcher	49994
SMTP/SUBMIT over SSL	465
Event Notification Service (ENS) over SSL	8997
LDAP over SSL	636
IMAP over SSL or MMP IMAP Proxy over SSL	993
POP3 over SSL or MMP POP Proxy over SSL	995
MSHTTPD over SSL	8991
managesieve	4190
MTQP	1038

You might need to specify a port number other than the default if you have, for example, two or more IMAP server instances on a single host machine, or if you are using the same host machine as both an IMAP server and a Messaging Multiplexor server. For more information about the Multiplexor, see the discussion about the Messaging Multiplexor (MMP) for standard mail protocols in *Messaging Server System Administrator's Guide*.

Keep the following in mind when you specify a port:

- Port numbers can be any number from 1 to 65535.
- Make sure the port you choose is not already in use or reserved for another service.

Close all unused ports, especially non-SSL ports. Opt for SSL-enabled ports, instead of non-SSL ports, for all communications (for example: HTTPS, IIOPS, t3s).

For more information about securing your OS, see your OS documentation.

Secure Communications

Secure connections between applications connected over the Internet can be obtained by using protocols such as Secure Sockets Layer (SSL) or Transport Layer Security (TLS). SSL is often used to refer to either of these protocols or a combination of the two (SSL/TLS). Messaging Server recommends the use of only TLS. However, throughout this guide, secure communications may be referred to by the generic term SSL.

In a Messaging Server deployment, you can enable the use of TLS between most components. See "Implementing Messaging Server Security" for more information.

LDAP Security

To enhance client security in communicating with Directory Server, use a strong password policy for user authentication. For more information on securing Directory Server, see the discussion about Directory Server security in *Oracle Directory Server Enterprise Edition Administration Guide*.

2

Planning Messaging Server Security

This chapter describes how to plan for and protect the various components of your Oracle Communications Messaging Server deployment.

Protecting Messaging Components in Your Deployment

This section describes how to secure components in your Messaging deployment.

Note: With each component, you should use the **chroot** function to limit the number of available commands on each machine. However, because Virtual Machines (VMs) generally provide better security isolation, use VMs instead of **chroot** when possible.

Protecting MTAs

Secure MTAs to protect processing resources and server availability. When messages are relayed from unauthorized users or large quantities of spam are delivered, response time is reduced, disk space is used up, and processing resources (which are reserved for end users) are consumed. Not only does spam waste server resources, it is also a nuisance for your end users.

Note: Not only must you protect your deployment from external unauthorized users, but you might also have to protect your system from internal users as well.

Table 2–1 describes the most common threats to MTAs.

Table 2–1 Common MTA Security Threats

Threat	Description
UBE (Unsolicited Bulk Email) or spam	Refers to the practice of sending electronic junk mail to millions of users.
Unauthorized relaying	Uses another company's SMTP server to relay your email. Spammers often use this technique to cover their tracks. End-users might send complaints back to the sending relay, not to the spammer.
Mail bombs	Characterized by abusers who repeatedly send an identical message to a particular address. The goal is to exceed mailbox quotas with the message.

Table 2–1 (Cont.) Common MTA Security Threats

Threat	Description
Email spoofing	Creates email that appears to have originated from one source when it was actually sent from another source.
Denial of service attacks	Prevents legitimate users of a service from using that service. For example, an attacker attempts to flood a network, thereby preventing legitimate network traffic.

This section on MTA relays describes the following security options you can use in your deployment:

- Integrating Third-Party Anti-spam and Anti-virus Software
- Monitoring Your Security
- Access Controls
- Preventing Relaying From Outside Hosts
- Conversion Channels and Third Party Filtering Tools
- RBL Checking
- Client Access Filters

Integrating Third-Party Anti-spam and Anti-virus Software

Any MTA on the Internet needs anti-spam and anti-virus (AVAS) software (for both outbound and inbound traffic). The recommended mechanism to integrate AVAS software with the Messaging Server MTA is by using the milter plug-in. See "Milter" for more information.

Monitoring Your Security

Monitoring your server is an important part of your security strategy. To identify attacks on your system, you should monitor message queue size, CPU utilization, disk availability, and network utilization. Unusual growth in the message queue size or reduced server response time may identify some of these attacks on MTA relays. Also, investigate unusual system load patterns and unusual connections. Review logs on a daily basis for any unusual activity.

Access Controls

You can use access controls to reject messages from (or to) certain users at a system level. In addition, you can institute more complex restrictions of message traffic between certain users. Also, you might allow users to set up filters on their own incoming messages (including rejecting messages based on contents of the message headers).

If you want to control access with envelope-level controls, use mapping tables to filter mail. If you want to control access with header-based controls, or if users want to implement their own personalized controls, use the more general mailbox filters approach with server-side rules.

Mapping Table Overview

You can control access to your mail services by configuring certain **mapping tables**. Many components of the MTA employ table lookup-oriented information. This type of table is used to transform, that is, **map**, an input string into an output string. Mapping tables are usually presented as two columns. The first (left-hand) column provides possible input strings against which to match (pattern), and the second (right-hand) column gives the resulting output string for which the input string is mapped (template).

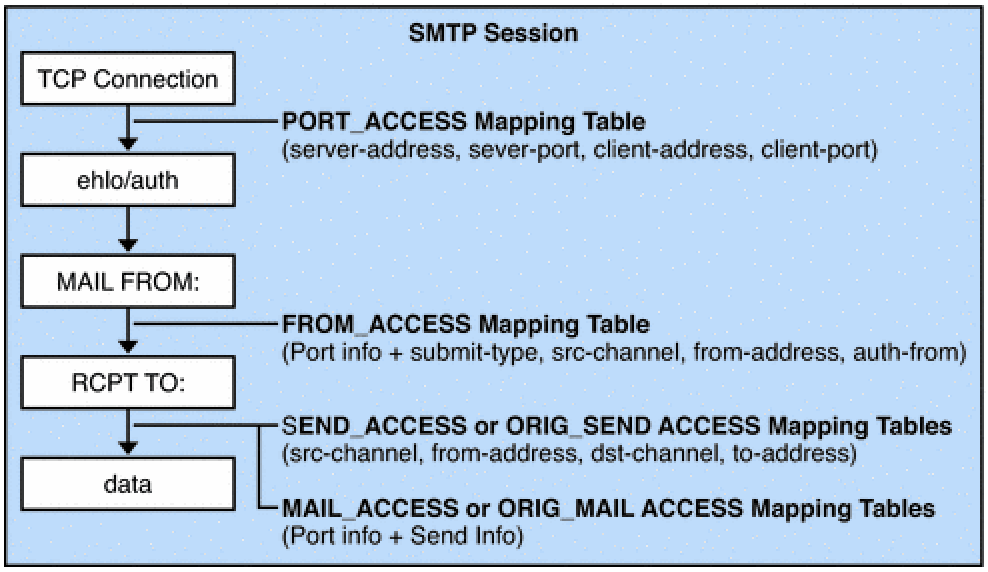
Table 2–2 describes these mapping tables, which enable you to control who can or cannot send mail, receive mail, or both. See *Messaging Server System Administrator's Guide* for more information.

Table 2–2 Access Control Mapping Tables

Mapping Table	Description
SEND_ACCESS	Used to block incoming connections based on envelope **From:** address, envelope **To:** address, source and destination channels. The **To:** address is checked after rewriting, alias expansion, and so on have been performed.
ORIG_SEND_ACCESS	Used to block incoming connections based on envelope **From:** address, envelope **To:** address, source and destination channels. The **To:** address is checked after rewriting but before alias expansion.
MAIL_ACCESS	Used to block incoming connections based on combined information found in **SEND_ACCESS** and **PORT_ACCESS** tables: that is, the channel and address information found in **SEND_ACCESS** combined with the IP address and port number information found in **PORT_ACCESS**.
ORIG_MAIL_ACCESS	Used to block incoming connections based on combined information found in **ORIG_SEND_ACCESS** and **PORT_ACCESS** tables: that is, the channel and address information found in **ORIG_SEND_ACCESS** combined with the IP address and port number information found in **PORT_ACCESS**.
FROM_ACCESS	Used to filter mail based on envelope **From:** addresses. Use this table if the **To:** address is irrelevant.
PORT_ACCESS	Used to block incoming connections based on IP number.

Figure 2–1 illustrates where mapping tables are activated in the mail acceptance process.

Figure 2–1 Mapping Tables and the Mail Acceptance Process


SMTP Session
TCP Connection
ehlo/auth
MAIL FROM:
RCPT TO:
data
PORT_ACCESS Mapping Table
(server-address, sever-port, client-address, client-port)
FROM_ACCESS Mapping Table
(Port info + submit-type, src-channel, from-address, auth-from)
SEND_ACCESS or ORIG_SEND ACCESS Mapping Tables
(src-channel, from-address, dst-channel, to-address)
MAIL_ACCESS or ORIG_MAIL ACCESS Mapping Tables
(Port info + Send Info)

For all the network ports controlled by the MTA service dispatcher, a **PORT_ACCESS** rejection response, if warranted, takes place at the initial connection from a remote host. A **FROM_ACCESS** rejection occurs in response to the **MAIL FROM:** command, before the sending side can send the recipient information or the message data. A **SEND_ACCESS** or **MAIL_ACCESS** rejection occurs in response to a **RCPT TO:** command, before the sending side gets to send the message data. If an SMTP message is rejected, your Messaging Server never accepts or sees the message data, thus minimizing the overhead of performing such rejections. If multiple access control mapping tables exist, Messaging Server checks them all.

Note: If the message is accepted, it can still be filtered by way of conversion channels and user defined filters.

Configuring Anti-Relaying with Mapping Tables

You can also use access control mappings to prevent people from relaying SMTP mail through your Messaging Server system. For example, someone might try to use your mail system to relay junk mail to thousands of mailboxes on your system or on other systems.

By default, Messaging Server prevents all SMTP relaying activity, including relaying by local POP and IMAP mail clients. If clients do not authenticate by using SMTP AUTH, as described in "Enabling Authenticated SMTP," and attempt to submit messages to external addresses through Messaging Server's SMTP server, their submission attempts are rejected. Thus, you will likely want to modify your configuration so that it recognizes your own internal systems and subnets from which relaying should always be accepted.

Preventing Relaying From Outside Hosts

By default, the MTA initial configuration prevents relaying from outside hosts. If you change this, and later want to return to preventing hosts that reside outside your domain from relaying to other hosts outside your domain, use the following procedure.

1. Split incoming mail into different channels. For example:

- IP addresses within your domain go to the **tcp_intranet** channel.
- Authenticated sessions go to the **tcp_auth** channel.
- All other mail is sent to the **tcp_local** channel.

2. Recognize and allow mail from your POP and IMAP clients by using an **INTERNAL_IP** mapping table, as explained in the discussion about filtering mail based on its source or header strings in *Messaging Server System Administrator's Guide*.

Using Mailbox Filters

A filter consists of one or more conditional actions to apply to a message. Messaging Server filters are stored on the server and evaluated by the server. They are sometimes called server-side rules (SSR).

You can create channel-level filters and MTA-wide filters to prevent the delivery of unwanted mail. The server applies filters in the following priority. See *Messaging Server System Administrator's Guide* for more information.

1. **Per-user filters** apply to messages destined for a particular user's mailbox. The filters reject unwanted messages, redirect mail, filter messages into mailbox folders, and so on. End users create these filters by using a client that supports the use of mail filters, such as Convergence.

   If a personal mailbox filter explicitly accepts or rejects a message, then filter processing for that message finishes. A filter template generalizes a Sieve script by replacing **hard-coded** elements of the Sieve script with prompts and input fields. A Java servlet is used to parse the Sieve templates and generate the user interface in the browser. When an end user supplies values in the input fields, the servlet takes those values and saves them in a Sieve script in the user's directory profile entry. The prompts and input fields are presented to the end user through the client interface. If the recipient user had no mailbox filter, or if the user's mailbox filter did not explicitly apply to the message in question, Messaging Server next applies the channel-level filter.

2. **Channel-level filters** apply to each message enqueued to a channel. A typical use for this type of filter is to block messages going through a specific channel. To create a channel-level filter, you must write the filter using Sieve. For specific instructions on creating filters with Sieve, see the discussion about filtering mail based on its source or header strings in *Messaging Server System Administrator's Guide*. If the channel-level filter explicitly accepts or rejects a message, then filter processing for that message finishes. Otherwise, Messaging Server next applies the MTA-wide filter, if one exists.

3. **MTA-wide filters** apply to all messages enqueued to the MTA. You can configure anti-spam software to return a spam score (typically in the header). The MTA-wide filter can then determine the default score that results in rejection of a message or routing of a message to a Spam folder. To create an MTA-wide filter, you must write the filter using Sieve. For specific instructions on creating filters with Sieve, see the discussion about filtering mail based on its source or header strings in *Messaging Server System Administrator's Guide*. By default, each user has no mailbox filter. When a user accesses the Convergence interface to create one or more filters, then their filters are stored in the LDAP Directory.

Conversion Channels and Third Party Filtering Tools

The conversion channel performs body-part-by-body-part conversions on messages through the MTA. This processing can be done by any site-supplied programs or command procedures. The conversion channel can do such things as convert text or images from one format to another, scan for viruses, translate languages, and so forth. Various message types of the MTA traffic are selected for conversion, and specific processes and programs can be specified for each type of message body part. If you are looking to use the conversion channel with a virus scanning program, you can either disinfect, hold, or reject messages. A special conversion channel configuration is consulted to choose an appropriate conversion for each body part. For more information, see the discussion about using predefined channel definitions in the MTA in *Messaging Server System Administrator's Guide*.

Note: Using specialized processing like a conversion channel puts additional load on your system. Be sure to account for it when you plan your sizing strategy.

With the conversion channel, you can use third-party anti-spam and anti-virus software solutions. You can also use the MTA API to create a channel to invoke a remote scanning engine. For more information on the MTA API, see *Messaging Server MTA Developer's Reference*.

In general, it is best that these third-party solutions are shielded from external sites and are only used on back-end or intermediate relays.

For more information, see the discussion about integrating and configuring spam and virus filtering software in *Messaging Server System Administrator's Guide*.

RBL Checking

The Mail Abuse Protection System's Real-time Blackhole List (MAPS RBL) is a list of hosts and networks that are known to be friendly or neutral to abusers who use these hosts and networks to either originate or relay spam, or to provide spam support services.

You can configure your MTAs to compare incoming connections against the MAPS RBL. You can also use DNS-based databases used to determine incoming SMTP connections that might send unsolicited bulk mail.

For more information, see the discussion about filtering mail based on its source or header strings in *Messaging Server System Administrator's Guide*.

Client Access Filters

Messaging Server supports sophisticated access control on a service-by-service basis for POP, IMAP, and HTTP. The Messaging Server access-control facility is a program that listens at the same port as the TCP daemon it serves. The access-control facility uses access filters to verify client identity and it gives the client access to the daemon if the client passes the filtering process.

If you are managing messaging services for a large enterprise or for a service provider, these capabilities can help you to exclude spammers and DNS spoofers from your system and improve the general security of your network.

As part of its processing, the Messaging Server TCP client access-control system performs (when necessary) the following analyses of the socket end-point addresses:

- Reverse DNS lookups of both end points (to perform name-based access control)
- Forward DNS lookups of both end points (to detect DNS spoofing)

The system compares this information against access-control statements called **filters** to decide whether to grant or deny access. For each service, separate sets of Allow filters and Deny filters control access. Allow filters explicitly grant access. Deny filters explicitly forbid access.

When a client requests access to a service, the access-control system compares the client's address or name information to each of that service's filters by using these criteria:

1. The search stops at the first match. Because Allow filters are processed before Deny filters, Allow filters take precedence.
2. Access is granted if the client information matches an Allow filter for that service.
3. Access is denied if the client information matches a Deny filter for that service.
4. If no match with any Allow or Deny filter occurs, access is granted. The exception is the case where there are Allow filters but no Deny filters, in which case lack of a match means that access is denied.

The filter syntax described here is flexible enough that you should be able to implement many different kinds of access-control policies in a simple and straightforward manner. You can use both Allow filters and Deny filters in any combination, even though you can probably implement most policies by using almost exclusively Allows or almost exclusively Denies.

Client access filters are particularly helpful if troublesome domains are a known quantity. While UBE filters must store and process every spam message, client access filters free Messaging Server from having to process any spammed messages. Because client access filters block mail from entire domains, use this feature with caution.

Note the following limitations to client access filters:

- An SMTP client is required to log in before relaying a message.
- Client access filters do not scale well for large deployments.

For more information on client access filters, see "Planning Messaging Server Security".

Protecting the Message Store

The most important data in a messaging server is the user's mail in the Message Store. The mail messages are stored as individual files, which are not encrypted. Consequently, access to the Message Store must be protected.

To secure the Message Store, restrict access to the machine where the store is installed. For information on passwords, see "Planning Messaging User Authentication".

Not only should you create password authentication to the store machine, you might also use tools like VPN access, **ssh**, or **RBAC**, which list valid users that are allowed to login to the machine.

In addition, a two-tiered architecture is recommended over a one-tiered architecture. Because the Message Store performs the most disk intensive work of any components in a messaging system, do *not* have filtering, virus scanning, and other disk-intensive security processes on the same machine. In a two-tiered architecture, you do not have to run UBE filters, anti-relay, and client access filters on the same machine as the Message Store, which can add load to your system. Instead, the MTAs handle that processing. In addition, user access to the store is limited through an MMP in a two-tiered deployment, potentially adding an extra security layer to the Message Store.

If you deploy a one-tiered architecture, be sure to account for the additional security processing and load (like SSL and virus scanning) that you will need. For more information, see the discussion about configuring Messaging Server to provide optimum performance, scalability, and reliability in *Messaging Server Installation and Configuration Guide*.

For additional Message Store security processing, set disk quotas per user to limit disk usage. Also, use administrator alarms if free space thresholds are fast approaching their limits. Like the MTA, be sure to monitor the server state, disk space, and service response times. For more information, see the discussion about managing the message store in *Messaging Server System Administrator's Guide*.

Protecting MMPs

Because the MMP serves as a proxy for the Message Store, it needs to protect access to end user data and guard against unauthorized access. User IDs and passwords provide basic authentication capabilities. In addition, you can use client access filters to limit user login to specific domains or IP address ranges.

Locate the MMP on a different machine (or under a different userID) in front of your POP or IMAP services. You can have front-end machines with just MMP and MTAs, and then have a physically secure network between those front-end machines, the mail stores, and the LDAP servers.

Special security considerations must be given to Convergence access to the message store when your users are logging in from the Internet. In general, you want to ensure that the stores are separated from the outside world by a firewall. Like the MMP, the Webmail Server supports both unencrypted and encrypted (SSL) communication with mail clients.

Regular monitoring of log files can protect against unauthorized access.

Planning Messaging User Authentication

User authentication enables your users to log in through their mail clients to retrieve their mail messages. Methods for user authentication include:

- Plain Text and Encrypted Password Login
- Authentication with Simple Authentication and Security Layer (SASL)
- Enabling Authenticated SMTP
- Certificate-based Authentication with Secure Sockets Layer (SSL)
- Client-based Authentication with Secure Sockets Layer (SSL)

Plain Text and Encrypted Password Login

User IDs and passwords are stored in your LDAP directory. Password security criteria, such as minimum length, are determined by directory policy requirements. Password security criteria is not part of Messaging Server administration. Refer to the latest Directory Server documentation to understand directory server password policies:

https://docs.oracle.com/cd/E19656-01/index.html

An administrator can set a messaging configuration option to determine if plain passwords are allowed or if SSL must be used when transmitting passwords to the server. For more information, see the **service.*xxx*.plaintextminciper** (where *xxx* is

HTTP, **POP**, or **IMAP**) option in *Messaging Server Reference*. The **RestrictPlainPasswords** option provides the equivalent function for the MMP.

Both plain text and encrypted password login can be used with POP and IMAP user access protocols.

Authentication with Simple Authentication and Security Layer (SASL)

SASL (RFC 2222) provides additional authentication mechanisms for POP, IMAP, and SMTP user access protocols. Table 2–3 describes the Messaging Server SASL support for the user access protocols.

Table 2–3 SASL Authentication User Access Protocols Support Matrix

Protocol	Plain	Login	CRAM-MD5	Certificate	APOP
SMTP AUTH	Yes	Yes	Deprecated	Yes	No
POP	Yes	No	Deprecated	Yes	Deprecated
IMAP	Yes	No	Deprecated	Yes	No
HTTP	Yes	No	No	Yes	No

Note:

- When using CRAM-MD5, passwords must be stored in plain text format in the LDAP directory server.
- To use APOP, CRAM-MD5, or DIGEST-MD5, passwords must be stored in plain text format in the LDAP directory server.

If you use SASL, user name and passwords are not encrypted unless SSL is used for the session. (For more information on SSL, see "Encryption with SSL".) The SASL mechanisms, PLAIN and LOGIN, encode authentication information, but can be easily decoded if captured. Despite this limitation, SASL is useful because it can be combined with SMTP AUTH (described in "Enabling Authenticated SMTP") to allow only authenticated users to relay mail through your system. For example, legitimate users can authenticate to the SMTP server, and the SMTP server can then be configured to switch to a different channel. In this way, the message from an authenticated session can come from a different TCP channel than a user that did not authenticate. A message from a user in your internal network can also be switched to differentiate it from a message coming from other sources just based on the IP address of the incoming connection.

For more information on SASL, see "Security and Access Control in Messaging Server".

Enabling Authenticated SMTP

By default, the standard SMTP port (25) is for mail transfer only. Mail relay for submissions from external networks is disabled and authentication is disabled. By default, the standard SMTP submit port (587) is for mail submission and requires authenticated SMTP. As many mail user agents still use port 25 for submission by default, it might be useful to enable SMTP authentication on port 25 for those clients.

By default, users need not submit a password when they connect to the SMTP service of Messaging Server to send a message. You can, however, enable password login to SMTP to enable authenticated SMTP.

Authenticated SMTP (also referred to as SMTP AUTH) is an extension to the SMTP protocol. Authenticated SMTP allows clients to authenticate to the server. The authentication accompanies the message. The primary use of authenticated SMTP is to enable local users who are not in their office to submit mail without creating an open relay that others could abuse. The **AUTH** command is used by the client to authenticate to the server.

Authenticated SMTP provides security in sending messages with the SMTP protocol. To use authenticated SMTP, you do not need to deploy a certificate-based infrastructure. (Certificate authentication is described in "Certificate-based Authentication with Secure Sockets Layer (SSL)".

With authenticated SMTP, the client can indicate an authentication mechanism to the server and perform an authentication protocol exchange.

If you require SMTP AUTH for mail submission, turn on appropriate logging, so any mail abuse can be traced.

For more information on authenticated SMTP, see the conceptual description of the Messaging Server MTA in *Messaging Server System Administrator's Guide*.

Certificate-based Authentication with Secure Sockets Layer (SSL)

Messaging Server uses the SSL protocol for encrypted communications and for certificate-based authentication of clients and servers. This section describes certificate-based SSL authentication. For information on SSL Encryptions, see "Encryption with SSL".

SSL is based on the concepts of public-key cryptography. Although TLS (Transport Layer Security) is functionally a superset of SSL, the names are used interchangeably.

At a high-level, a server which supports SSL needs to have a certificate, public key, private key, and security databases. This helps assure message authentication, privacy, and integrity.

Table 2–4 describes the SSL authentication support with each client access protocol. This table shows whether a secure session (STARTTLS) could be started up over an insecure channel and whether a separate secure channel (SSL on Separate Port) is provided.

Table 2–4 SSL Authentication Support Matrix

Protocol	STARTTLS	SSL on Separate Port
SMTP (RFC 5321)	Yes	No
SMTP Submission (RFC 6409)	Yes	Yes
POP	Yes	Yes
IMAP	Yes	Yes
POP over MMP	Yes	Yes
IMAP over MMP	Yes	Yes
Webmail	No	Yes

The SMTP_SUBMIT, POP, and IMAP protocols provide a way for the client and server to start communication without SSL, and then switch to it by using an equivalent **STARTTLS** command. The SMTP_SUBMIT, POP, and IMAP servers can also be configured to use SSL on an alternate port, for clients which do not implement **STARTTLS**.

In general, SSL requires server authentication (although SMTP relay is an exception to this rule). To use SSL, you must obtain a server certificate for your Messaging Server. The certificate identifies your server to clients and to other servers. Your server can also have any number of certificates of trusted Certificate Authorities (CAs) that it uses for client authentication.

Some protocols require use of the SASL EXTERNAL mechanism with the SSL client certificate to move from un-authenticated to authenticated state.

For more information on SSL, see "Security and Access Control in Messaging Server".

Client-based Authentication with Secure Sockets Layer (SSL)

SSL can perform both client and server authentication. In general, client authentication with SSL is only necessary in high-security sites. Client authentication is not necessarily automatic, as the server needs to know how to convert the subject of the client certificate into a user identity. To authenticate with SSL, the mail client establishes an SSL session with the server and submits the user's certificate to the server. The server then evaluates if the submitted certificate is genuine. If the certificate is validated, the user is considered authenticated.

Third-Party Authentication Server Support

This section contains the following topics:

- Messaging Mutiplexor (MMP) Support
- IMAP/POP/SMTP Support
- Sample Code

Messaging Server provides support for third-party authentication servers by supporting a protocol designed to integrate third-party authentication services. This protocol is documented in the file **authserver.txt**, which is installed as part of the Messaging Server in the **examples/tpauthsdk** directory.

Support for third-party authentication servers addresses two primary problems:

- If your computing infrastructure does not store passwords in LDAP, Messaging Server can be configured to query the authentication server you provide to verify passwords. This circumvents the need to replicate passwords from the third-party authentication service to the LDAP server used by Messaging Server for user information.
- If you want to use an authentication system that provides security or management capabilities not possible with traditional password authentication (such as Kerberos), Messaging Server can be configured to pass SASL mechanisms to the authentication server you provide for processing. While Messaging Server does not directly support Kerberos, it is now possible to write code that adds Kerberos capability to Messaging Server.

The **examples/tpauthsdk** directory also contains sample code for a third-party authentication server that can validate and modify authentication information provided by the Messaging Server.

Messaging Mutiplexor (MMP) Support

To enable third-party authentication in the MMP, the **PreAuth** configuration option must be enabled. For authentication methods other than PLAIN, you must specify the proxy authentication credentials (**StoreAdmin** and **StoreAdminPass**) and add the following configuration option to the MMP's **ImapProxyAService.cfg** and/or **PopProxyAService.cfg** file:

default:AuthenticationServer :56

For plain text logins, the MMP will first perform a normal user lookup in LDAP. Once the user is located, the MMP will connect to the host (localhost loopback) and port (56) specified in the **AuthenticationServer** option to authenticate the user. The MMP will pass the LDAP attributes **inetUserStatus**, **mailUserStatus**, **uid** and **mailHost** to the authentication server. You may also configure the MMP to look up additional LDAP attributes and pass them to the authentication server with the option:

default:AuthenticationLdapAttributes "attr1" "attr2" ...

The authentication server should be running on the same server as the MMP and on a restricted port. The protocol used to communicate between the MMP and the third-party authentication server is not presently secured.

The MMP advertises the additional SASL mechanisms provided by the authentication server through the standard server protocols (for example, through the IMAP capabilities). The authentication server can provide a SASL mechanism that the MMP implements natively. In this case, the authentication server mechanism will take precedence. To see a transcript of the communication between the MMP and authentication server process, add **authserv** to the **default:debugkeys** configuration setting for the MMP and set **default:LogLevel** to **Debug**. State transitions in the HULA authentication subsystem can also be logged through the **hula** debug key.

For more information about configuring and administering multiplexor services, see *Messaging Server System Administrator's Guide*.

IMAP/POP/SMTP Support

To enable third-party authentication with IMAP/POP/SMTP, use the **msconfig** options **auth.authenticationserver** and **auth.authenticationserver** or the **configutil** options **sasl.default.authenticationserver** and **sasl.default.authenticationldapattributes**. These work as they do for the MMP except the **authenticationattributes** is a space-delimited list (quotes are not used). The *local.debugkeys* option provides functionality equivalent to the **default:debugkeys** option, but again as a space separated list.

To debug third-party authentication with store, set msconfig option base.debugkeys or configutil option **local.debugkeys** to include **authserv** (or **authserv hula**) and set **msconfig** option **{imap/pop}.logfile.loglevel** or **configutil** option **logfile.loglevel** to **debug.**

Sample Code

The third-party authentication sample code in the **tpauth** directory is largely suitable for a production environment (it uses a thread-pool model to handle a high volume of connections). However, the Messaging Server product team will not provide support for this sample code. The sample code is designed for use on a system which provides the standard Posix Threads API (for example, Oracle Solaris or Linux). The **Makefile.sample** is for use on Oracle Solaris.

Table 2–5 lists the contents of the **tpauth** directory.

Table 2–5 Contents of the tpauth Directory

File	Contents
README.txt	A link to this document.

Table 2–5 (Cont.) Contents of the tpauth Directory

File	Contents
authserver.txt	Third-Party Authentication Protocol Specification.
Makefile.sample	Use **make -f Makefile.sample** to build the sample code.
authserv.c	The core thread-pool protocol server implementation.
authserv.h	The API called by **authserv.c** to authenticate users.
sample.c	A very simple sample third-party authentication module using plain-text passwords. Third-parties may edit or replace this module to provide authentication services.
sample2.c	A simple CRAM-MD5 example demonstrating use of this interface for non-plain-text mechanisms.

Planning Message Encryption Strategies

This section describes encryption and privacy solutions.

Encryption with SSL

SSL functions as a protocol layer beneath the application layers of IMAP, HTTP, and SMTP. If transmission of messages between a Messaging Server and its clients and between the servers and other servers is encrypted, there is little chance for eavesdropping on the communications. If connecting clients and servers are authenticated, there is little chance for intruders to spoof them.

End-to-end encryption of message transmission requires the use of S/MIME. See the discussion about the basic configuration procedure to set up S/MIME for Convergence in *Convergence System Administrator's Guide* and the discussion about administering S/MIME in *Messaging Server System Administrator's Guide*.

Note: The extra performance overhead in setting up an SSL connection can put a burden on the server. In designing your messaging installation and in analyzing performance, you must balance security needs against server capacity.

The SSL connection process between client and server using HTTP/SSL (HTTPS) is as follows:

1. The client initiates contact using HTTPS. The client specifies which secret-key algorithms it can use.
2. The server sends its certificate for authentication and specifies which secret-key algorithm should be used. It will specify the strongest algorithm which it has in common with the client. If there is no match, the connection will be refused. If the server has been configured to require client authentication, it will ask the client for its certificate at this point.
3. The client checks the validity of the server certificate to ensure that it has:
   - Not expired
   - A known signed Certification Authority
   - A valid signature

- A host name on the certificate that matches the name of the server in the HTTPS request

SSL Ciphers

SSL enables the server and client to negotiate cipher suites. The cipher suites determine the cryptographic algorithms and key sizes used to secure the SSL connection. Over time, cipher suites change. Older cipher suites can become insecure due to key size (for example, DES) or due to discovery of security design errors in algorithms they use (for example, RC4 and MD5). However, sometimes you must use insecure cipher suites for interoperability with old clients that have not yet been updated.

In Messaging Server, cipher suites can be enabled or disabled by default, newer cipher suites can be added, and insecure ones eventually removed from the product. Changes to cipher suites can occur in any Messaging Server patch release. For more information about cipher suites, see the discussion about the **adjustciphersuites** option in *Messaging Server Reference*.

In general, software should support at least two cryptographic algorithms for any given purpose to provide a backup if a weakness in the preferred algorithm is discovered. However, supporting more than two cryptographic algorithms is undesirable because it increases the attack surface and reduces interoperability.

For more information about ciphers, see "Security and Access Control in Messaging Server".

Signed and Encrypted S/MIME

Secure/Multipurpose Internet Mail Extensions (S/MIME) provides a consistent way for email users to send and receive secure MIME data, using digital signatures for authentication, message integrity, and non-repudiation and encryption for privacy and data security. S/MIME version 3.1 (RFC 3851) is supported.

Several email clients support the S/MIME specification, including Microsoft Outlook and Mozilla mail.

You can deploy a secure mail solution by using Messaging Server and S/MIME. Convergence users who are set up to use S/MIME can exchange signed or encrypted messages with other users of Convergence, Microsoft Outlook, and Mozilla mail systems.

For more information on S/MIME and Convergence, see the discussion about configuring and storing certificate information in Messaging Server and Directory Server *and* the discussion about the basic configuration procedure to set up S/MIME for Convergence in *Convergence System Administrator's Guide*. For other clients that support S/MIME, see that client's documentation for information on S/MIME configuration.

Planning a Messaging Server Anti-spam and Anti-virus Strategy

Messaging Server provides many tools for dealing with unsolicited bulk email (UBE, or spam) and viruses. This information describes the various tools and strategies available for your use.

Anti-spam and Anti-virus Tools Overview

As more computers are connected to the Internet, and the ease of doing business online increases, the frequency of security incidents, including spam and viruses, continues to rise. You should plan your Messaging Server deployment to deal with these problems.

Mail traffic passing into, through, and out of Messaging Server can be separated into distinct channels according to various criteria. This criteria includes source and destination email addresses, and source IP address or subnet. You can apply different processing characteristics to these different mail flows, or channels. Consequently, you can use different access controls, mail filters, processing priorities, and tools in different ways and combinations on these channels. For example, you can process mail originating from within your domain differently from mail originating from outside your deployment.

In addition to channel-based message flow classification, another useful classification is mailing list traffic. Traffic for a given mailing list can come into Messaging Server through many different channels and go back out through many different channels. When using mailing lists, you can find it helpful to think in terms of the list itself and not in terms of channels. Messaging Server recognizes this and enables many of the channel-specific spam fighting tools to also be applied in a mailing-list specific fashion.

The following summarizes the anti-spam and anti-virus tools you can use with Messaging Server:

- "Milter". Provides a plug-in interface for third-party software to validate, modify, or block messages as they pass through the MTA.
- "Access Control". Rejects mail from known spam sources and enables control over who can send or receive email within the organization.
- "Mailbox Filtering". Enables users to manage their own spam 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 spam sources as identified by the Mail Abuse Protection System's Real-time Blackhole List (MAPS RBL), a responsibly managed, dynamically updated list of known spam sources.
- "Relay Blocking". Prevents abusers from using a mail system as a relay to send their spam to tens of thousands of recipients.
- "Authentication Services". Enables password authentication in an SMTP server with the Simple Authentication and Security Layer (SASL) protocol.
- "Sidelining Messages". Silently sidelines or even deletes potential spam messages.
- "Comprehensive Tracing". Uses reliable mechanisms for identifying a message’s source.
- "Conversion Channel". Integrates with third-party anti-virus or anti-spam products.
- "MeterMaid". Provides centralized metering and management of connections and transactions.
- "memcached". Can provide the same functions as MeterMaid.

You can use these tools individually or together. No one tool by itself will block all spam. However, taken together, these tools provide an effective means of combating unauthorized use of your mail system. The following sections provide more details on these tools. For more information, see *Messaging Server System Administrator's Guide*.

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. Messaging Server provides a library that emulates the sendmail side of the Milter interface. Consequently, Milters written for sendmail can also be used with 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 Messaging Server, in a single system deployment, or in a two-tier deployment. Messaging Server also supports connecting to multiple Milter servers.

See the topics on milter implementation and milter **spamfilter*N*_config_file** in *Messaging Server Reference*.

Access Control

Messaging Server has a general purpose mechanism that you can use to reject mail in accordance with a variety of criteria. This criteria includes the message source or destination email addresses, and source IP address. For example, you can use this mechanism to refuse mail from specific senders or entire domains (such as mail from *spam@public.com*). Should you have large lists of screening information, you can extend your lists 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 or are not allowed to send mail out of certain channels. For example, you can restrict on a per-user basis who can or cannot send or receive Internet mail.

See "Access Controls" for more information.

Mailbox Filtering

Messaging Server provides mail filters on a per-user, per-channel, and system-wide basis. Per-user channels can be managed from any web browser in Convergence. Using these filters, users can control what mail messages are delivered to their mailbox. For example, a user tired of “make money fast” UBE can specify that any message with such a subject be rejected. Mail filtering in Messaging Server is based on the Sieve filtering language (RFCs 3028 and 3685) developed by the Internet Engineering Task Force (IETF).

See "Using Mailbox Filters" for more information.

You can also implement content-based filtering or virus scanning with third-party content filtering software, such as Cloudmark.

See "Anti-spam and Anti-virus Considerations" for more information.

Address Verification

UBE messages often use invalid originator addresses. The Messaging Server SMTP server can take advantage of this by reflecting 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 DNS server, the message can be rejected. A potential performance penalty can be incurred with such use of the DNS.

You enable address verification on a per-channel basis with the **mailfromdnsverify** channel keyword described in *Messaging Server System Administrator's Guide*.

Real-time Blackhole List

The Mail Abuse Protection System's Real-time Blackhole List (MAPS RBL) is a dynamically updated list of known UBE sources identified by source IP address. The Messaging Server SMTP server supports 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, see:

`https://www.ers.trendmicro.com/`

Use of the RBL by the Messaging Server SMTP server is enabled with the **ENABLE_ RBL** option of the MTA Dispatcher.

Relay Blocking

A comprehensive UBE strategy should include ways to prevent users from receiving UBE access controls, mailbox filtering, address verification, and RBL, and preventing users from unauthorized relay of mail from your system to other systems. This second method is called relay blocking. In its simplest form, relay blocking is achieved by enabling local users and systems to relay mail while rejecting relay attempts from non-local systems. Using IP addresses as the differentiator easily and securely makes this differentiation between local versus non-local. By default, Messaging Server enables relay blocking upon installation.

See "Configuring Anti-Relaying with Mapping Tables" for more information.

Authentication Services

The Messaging Server SMTP server implements the Simple Authentication and Security Layer (SASL, RFC2222) protocol. SASL can be used with POP and IMAP clients to provide password-based access to your SMTP server. A typical usage for SASL is to permit mail relaying for external authenticated users. This solves the common problem posed by local users who use ISPs from home or while traveling. Such users, when connecting to your mail system, will have non-local IP addresses. Any relay blocking that takes into account only the source IP address will not permit these users to relay mail. This difficulty is overcome with SASL, which enables these users to authenticate themselves. Once authenticated, the users are permitted to relay mail.

Sidelining Messages

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

Comprehensive Tracing

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

Conversion Channel

The conversion channel is a very general purpose interface where you can 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 or perform content filtering for company sensitive information.

Integration with Third-party Products

Content-filtering software from third-party suppliers can be hooked in to your deployment through Messaging Server's conversion channel. Channel keywords are used to enable mail filtering using anti-spam and anti-virus products, such as Cloudmark or Proofpoint. You can configure the MTA to filter for all messages or only those going from or to certain channels, or to set the granularity at a per-user level. A user can decide to use spam or virus filtering, or both.

An extensive Sieve support enables great flexibility to set the disposition of the message determined to be a virus or spam. You can take the default action of discarding the virus and spam, or filing the spam into a special folder. But using Sieve, you can forward a copy of the message to some special account, add a custom header, or use the spamtest Sieve extension to take a different action based on a rating returned.

MeterMaid

MeterMaid is a server that can provide centralized metering and management of connections and transactions. Functionally, MeterMaid can be used to limit how often a particular IP address can connect to the MTA. Limiting connections by particular IP addresses is useful for preventing excessive connections used in denial-of-service attacks.

memcached

memcached can provide the same functions as MeterMaid.

Anti-spam and Anti-virus Considerations

This section describes issues to keep in mind when planning your deployment to use anti-spam or anti-virus technologies.

Architecture Issues with Anti-spam and Anti-virus Deployments

The Messaging Server MTA can reside on the same system as the mail filtering system, such as Cloudmark, or you can use separate systems. One of the advantages of separating the MTA from the mail filtering servers is that you can add more processing power for the filtering simply by adding more hardware and cloning the servers. While the system is capable and not overloaded, you can have the mail filtering server software collocated with the MTA.

In general, consider deploying a farm of servers that the MTAs utilize to filter mail. You can configure MTAs to use a list of server names, which essentially the MTAs will load balance on. (This load balancing functionality is usually provided by the filtering SDK.) The advantage of having the server farm is that when you need more processing power, you can simply add more servers.

Mail filtering products tend to be CPU-intensive. Creating an architecture that separates the MTA and the mail filtering products onto their own machines provides for better overall performance of the messaging deployment.

Note: Because mail filtering servers tend to be CPU-intensive in nature, you could end up with an architecture consisting of more mail filtering systems than the MTA hosts they are filtering for.

In larger deployments, consider also creating inbound and outbound mail filtering pools of servers that are associated with the respective inbound and outbound MTA pools. You can also create a **swing** pool that can be utilized as either an inbound or outbound pool, in response to need in either area.

As with the rest of the deployment, you must monitor the mail filtering tier. A threshold of 50 percent CPU utilization is a good rule of thumb to follow. Once this threshold has been met, you must consider adding more capacity to the mail filtering tier.

Security Issues with Anti-spam and Anti-virus Deployments

When planning to deploy anti-spam or anti-virus technology, keep in mind that an incorrect deployment can defeat your security measures. Figure 2–2 shows an incorrect deployment of an anti-spam/anti-virus filter solution.

Figure 2–2 Incorrect Deployment of Anti-spam/virus Solution

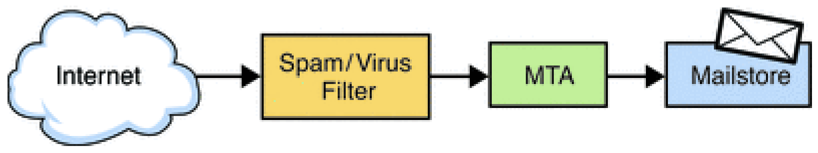
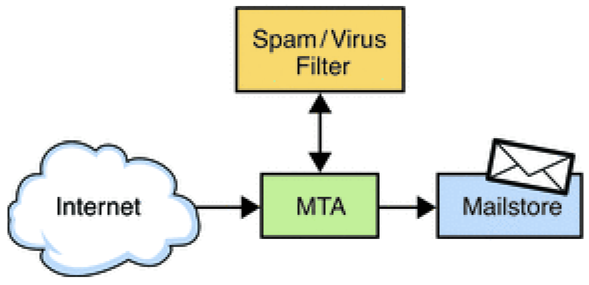


Figure 2–3 shows a correct deployment of an anti-spam/virus filter solution.

Figure 2–3 Correct Deployment of Anti-spam/virus Solution



The MTA performs certain functions well, including:

- Rejecting messages as early as possible

- Per-user configuration and policy
- Email security and routing policy
- Mail queue management

The anti-spam/virus filter is good at determining if an email is spam or has a virus, but is generally not nearly as good at doing the things expected of a good MTA. Thus, do not depend on an anti-spam/virus filter to do those things. Your deployment is more **correct** when the anti-spam/virus filter is well integrated with the MTA, which is the case with Messaging Server. Messaging Server spam filter plug-in support provides all the potential reasons to reject a message early and applies all reasons at the same time.

A robust MTA, such as Messaging Server's, contains security features (SSL/TLS, traffic partitioning by IP address, early address rejection to reduce denial-of-service attacks, connection throttling by IP address/domain, and so on), which are defeated when an anti-spam/virus filter is deployed in front. Furthermore, anti-spam/virus filters that communicate by using the SMTP protocol often do not follow the robustness requirements of SMTP and thus lose email when they should not. A correct deployment should have the anti-spam/virus filter working with a robust MTA.

Implementing an RBL

In general, implementing an RBL provides the most immediate benefit to reducing spam traffic. A good RBL implemented by your MTAs immediately reduces spam by a minimum of 10 percent. In some cases, this number could approach 50 percent.

You can use your RBL and anti-spam/virus filters together. If the anti-spam/virus filter takes care of 95 out of 100 emails for a certain IP address within some amount of time you should add that IP address to your RBL. You can adjust the RBLs for the anti-spam/virus filter’s false positives when you do your analysis. That makes the RBL much more proactive in handling a specific wave of spam.

Developing an Anti-spam and Anti-virus Site Policy

When developing a policy for preventing spam and relaying, strike a balance between providing safety from spam and providing a site where emails are delivered in a timely fashion. The best policy is therefore to initially provide a core set of measures that do not take up too much processing time but trap the majority of spam. You can then define this core set of measures after stress testing the final architecture. Start with the initial measures below. Once you have deployed your system, monitor trapped and non-trapped spam to fine tune the system and replace or add new functions if required.

Use the following set of measures as a starting point for your site's anti-spam and anti-virus policy:

- Anti-relay should be provided by the **ORIG_SEND_ACCESS** settings. This is structured to enable only subscribers and partnership users access to deliver externally bound SMTP mail.
- Use authentication services to validate roaming users. These users verify their identity before being allowed to route externally bound SMTP mail.
- Implement subject line checking for common spam phrases using the system-wide mailbox filters.
- Set a maximum number of recipients using the **holdlimit** keyword. This will have the effect of sidelining potential spam traffic. The initial value could be set at 50

recipients and should be monitored over a period to determine whether a higher or lower value is required.

- Set up dummy accounts that are then manually used by the postmasters to encourage spam to these specific accounts to identify new spam sites.
- A message in which a virus has been detected should not be returned to the original sender and should not be forwarded to the intended recipient. There is no value in this because most viruses generate their own mail with forged sender addresses. It has become very rare that such infected messages will have any useful content.
- Send infected messages to an engine that harvests and catalogues information about the virus. You can then use such information to create threat reports for your system administrators about new virus and worm outbreaks.

3 Performing a Secure Messaging Server Installation

This chapter presents planning information for your Oracle Communications Messaging Server system and describes recommended installation guidelines that enhance security.

For more information about installing Messaging Server, see *Messaging Server Installation and Configuration Guide*.

Installing Infrastructure Components Securely

The following infrastructure components should be installed and secured prior to Messaging Server installation. You must understand how all components in the infrastructure communicate so that you can apply appropriate security measures to every interconnect.

- **Directory Server**: Messaging Server connects to the Oracle Directory Server Enterprise Edition, an LDAP-based directory server for user and group information and for provisioning. See the discussion about enhanced security in *Oracle Fusion Middleware Evaluation Guide for Oracle Directory Server Enterprise Edition*.
- **Directory Server Setup Script**: The **comm_dssetup.pl** script prepares the Directory Server for Messaging Server installation.
- **High Availability Planning**: Plan your deployment to tolerate failure of any one component. The approach to achieve high availability for store machines differs between classic message store and Cassandra message store.
- **DNS Server**: You must ensure that Domain Name System (DNS) is running and configured properly. For details, see *Messaging Server Installation and Configuration Guide*.
- **File System**: See the discussion about recommended file systems for message stores in *Messaging Server Installation and Configuration Guide*.

In addition to dependent products, it is equally important to secure the other components within Unified Communications Suite for secure Messaging Server deployment.

Review the following guidelines for components that impact Messaging Server security:

- **Convergence**: See the discussion about the overview of Convergence security in *Convergence Security Guide* for more information.

- **Connector for Microsoft Outlook**: See *Connector for Microsoft Outlook Security Guide* for more information.
- **Indexing and Search Service**: See *Indexing and Search Service Security Guide* for more information.
- **Contacts Server**: See *Contacts Server Security Guide* for more information.
- **Delegated Administrator**: See *Delegated Administrator Security Guide* for more information.

Credentials Needed to Install Messaging Server Components

The installation prompts for the following authentication credentials:

- User Name and Group Name for Server Processes
- Directory Server manager (bind DN and password)
- Password for server administration

Post-Installation Configuration

By default, when you install and configure Messaging Server, SMTP relay blocking is enabled. That is, Messaging Server rejects attempted message submissions to external addresses from unauthenticated external sources (external systems are any other system than the host on which the server itself resides). This default configuration is quite aggressive in blocking SMTP relaying in that it considers all other systems to be external systems.

Other post-installation steps to configure Messaging Server for a secure installation include:

1. Installing Messaging Server provisioning tools
2. Enabling startup after a reboot
3. Enabling SSL

For instructions about the first two items, see the discussion about the Messaging Server initial configuration in *Messaging Server Installation and Configuration Guide*. See "Security and Access Control in Messaging Server" for information about enabling Messaging Server components for SSL.

Note: Once installation is complete, Oracle recommends encrypting and moving the initial state files and **configure.ldif** file, if generated.

4

Implementing Messaging Server Security

This chapter explains the security features of Oracle Communications Messaging Server. It also provides links to security topics that provide more in-depth information for configuring and administering Messaging Server securely.

Security Features

Messaging Server supports security features that enable you to keep messages from being intercepted, prevent intruders from impersonating your users or administrators, and permit only specific people access to specific parts of your messaging system. The Messaging Server security architecture is part of the security architecture of Oracle servers as a whole. It is built on industry standards and public protocols for maximum interoperability and consistency. For a general overview of Messaging Server security strategies, see "Planning Messaging Server Security".

Messaging Server Security Strategy for your Deployment

Creating a security strategy is one of the most important steps in planning your deployment. Your strategy should meet your organization's security needs and provide a secure messaging environment without being overbearing to your users.

How you set up the following topics impacts your security strategy guidelines:

- Creating a Security Strategy
- Identifying Password Policy Requirements
- Verifying File Ownership for Configuration Files
- Securely Monitoring and Auditing Your Messaging Server Deployment
- Tracking Security Patches
- Identifying Legal-intercept Requirements
- Securing Your Archiving Needs
- Disabling Users in Response to Abuse/Appeal Process
- Utilizing a Disk Consumption Growth Plan
- Preventing Unrelated Usage of Messaging Server Hosts and Virtual Machines
- Determining Security Capabilities of Your Supported Mail Clients

Creating a Security Strategy

Your security strategy needs to be simple enough to administer. A complex security strategy can lead to mistakes that prevent users from accessing their mail, or it can allow users and unauthorized intruders to modify or retrieve information that you do not want them to access.

The five steps to developing a security strategy, as listed in RFC 2196, the Site Security Handbook, include:

- Identifying what you are trying to protect. For example, your list might include hardware, software, data, people, documentation, network infrastructure, or your organization's reputation.
- Determining what you are trying to protect it from. For example: unauthorized users, spammers, or denial of service attacks.
- Estimating how likely threats are to your system. If you are a large service provider, your chances of security threats could be greater than a small organization. In addition, the nature of your organization could provoke security threats.
- Implementing measures that will protect your assets in a cost-effective manner. For example, the extra overhead in setting up an SSL connection can put a performance burden on your Messaging deployment. In designing your security strategy, you must balance security needs against server capacity.
- Continuously reviewing your strategy and making improvements each time a weakness is found. Conduct regular audits to verify the efficiency of your overall security policy. You can do this by examining log files and information recorded by the SNMP agents. For more information on SNMP, see *Messaging Server System Administrator's Guide*.

Your security strategy should also plan for:

- Physical Security
- Server Security
- Operating System Security
- Network Security
- Messaging Security
- Application Security

Physical Security

Limit physical access to important parts of your infrastructure. For example, place physical limits on routers, servers, wiring closets, server rooms, or data centers to prevent theft, tampering, or other misuse. Network and server security become a moot point if any unauthorized person can walk into your server room an unplug your routers.

Server Security

Limiting access to important operating system accounts and data is also part of any security strategy. Protection is achieved through the authentication and access control mechanisms available in the operating system.

In addition, you should install the most recent operating environment security patches and set up procedures to update the patches once every few months and in response to security alerts from the vendor.

Operating System Security

Reduce potential risk of security breaches in the operating environment by having an ongoing plan to apply OS security updates promptly when they become available. Also, consider using a file change control and audit tool, such as Tripwire for Servers, or Oracle Solaris Fingerprint Database, to track changes in files and detect possible intrusion.

Network Security

The recommended deployment configuration, to support both horizontal scalability and service security, is to place the access layer of the architecture behind a firewall. In a two-tiered architecture, use two firewalls, creating a DMZ. This enables access to the information delivery elements, the calendar and messaging front ends, while protecting the main service elements on the internal network behind a second firewall. Such a configuration also enables the access layer and data layer elements to be scaled independently, accommodating traffic and storage elements.

Limiting access to your network is an important part of your security strategy. Normally, overall access to networks is limited with firewalls. However, email must be made available outside your site. SMTP is one such service that allows for this.

To secure your network, you should:

- Turn off all operating system-provided services that listen on ports that you do not use.
- Place your application servers behind a packet filter, which drops external packets with an internal source IP address. A packet filter forbids all connections from the outside except for those ports that you explicitly specify.

Messaging Security

Messaging Server offers the following sets of security features:

- **Protecting Messaging Components in Your Deployment**. With this set of options, you can secure your MTA relays, message stores, Webmail clients, and multiplexing services. In addition, you'll learn about third-party spam filter options.
- **User Authentication**. These options enable you to determine how your users will authenticate to your mail servers, preventing unauthorized users from gaining access to your system.
- **Message Encryption**. Using this set of options, you can perform user authentication and protect the message itself by using authenticated SMTP and certificates for digital signatures, encryption, and Secure Sockets Layer (SSL).

See "Planning Messaging Server Security" for more information.

Application Security

Messaging Server provides features that ensure the security and integrity of business communications. Messaging Server offers extensive built-in security features, such as:

- Authentication
- Message and session encryption
- Virus and spam protection
- Archiving and auditing of communications

Implementing Secure Connections

Messaging Server supports the SSL/TLS security standards. SSL/TLS enables all communication between clients and servers to take place inside an encrypted session.

A more commonly used mechanism for data security protects the data across the wire only (that is, from client to server) by using SSL encryption on the connections used to transmit data between various messaging agents. This solution is not as complete as public key encryption, but is far easier to implement and is supported by many more products and service providers.

What problem does using SSL from client to server solve? An organization assumes that it controls its own corporate network and that data transmitted on that network is safe from non-employees. Mail sent to anyone from outside the corporate network using the corporation's infrastructure transmits the data over an encrypted connection to the corporation's network. Likewise, all mail picked up by a corporate user from outside the corporate network will be transmitted over an encrypted connection. Thus, if the enterprise's assumption about the safety of the internal network is true, and its employees use only sanctioned servers for transmission between themselves and other employees, mail between employees is safe from external attack.

What problem doesn't this solution solve? First of all, this approach does not protect the data from unintended viewing by non-intended recipients of the data who have access to the organization's internal network. Secondly, there is no protection offered for data being transmitted between employees and their external partners, customers, or suppliers. The data travels across the public Internet in a completely insecure fashion.

However, this problem can be remedied by configuring SSL encryption between MTA routers at both the enterprise's and customer's network. This type of solution requires setup for each private connection you want to use. In so doing, you add an important additional layer of security with customer or partner data being sent or received through email. Using MTAs and SSL, companies can save money by using the public Internet as the transport, but force the MTAs to use SSL for their partners. This solution does not take into account other network traffic to and from partners. Nevertheless, mail is usually a large proportion of the traffic, and because companies can pay based on data transmitted, using the public Internet is usually cheaper.

Identifying Password Policy Requirements

The user password login process is described in *Messaging Server System Administrator's Guide*.

Review the following additional password policy recommendations:

1. Select a password policy system that meets your requirements and any additional requirements you might add at a later time.
2. Require your users to create high quality passwords on your site identity system's password change web page. Do not require your users to change passwords too frequently as it cause users to write their passwords on paper.
3. Directory Server has password policy capabilities, but if you enable password expiration, be sure the administrator and service accounts used by Messaging Server are exempt from expiration.
4. Keep a strong administration password.
5. Maintain administrative access policies for Messaging Server hosts.
6. Create Delegated Administrator access policies for domains.
7. If needed by Oracle Support, plan how to gather configuration files, excluding passwords. If you use Unified Configuration, this task is made much easier.

Verifying File Ownership for Configuration Files

Related topics include:

- The discussion about ownership of a mail server user account in *Messaging Server System Administrator's Guide*.
- The discussion about identifying, analyzing, and resolving user mailbox directory problems in *Messaging Server System Administrator's Guide*.

Securely Monitoring and Auditing Your Messaging Server Deployment

Monitoring your server is an important part of your security strategy. To identify attacks on your system you should monitor message queue size, CPU utilization, disk availability, and network utilization. Unusual growth in the message queue size or reduced server response time may identify some of these attacks on MTA relays. Also, investigate unusual system load patterns and unusual connections. Review logs on a daily basis for any unusual activity.

See the discussion about monitoring the Messaging Server for signs of problems in *Messaging Server System Administrator's Guide*, which includes topics on:

- Automatic Monitoring and Restart
- Daily Monitoring Tasks
- Utilities and Tools for Monitoring
- Monitoring User Access to the Message Store
- Monitoring System Performance
- Monitoring Disk Space
- Monitoring the MTA
- Monitoring LDAP Directory Server
- Monitoring the Message Store
- SNMP Support
- How to Monitor MeterMaid

Additional guidelines for secure monitoring:

- Ensure you have the right monitoring and auditing tools for your specific deployment and that you have contingency plans in place.
- Enable MTA logging.

Tracking Security Patches

Be sure to install the most recent operating environment security patches and set up procedures to update the patches once every few months and in response to security alerts from the vendor. Be sure to pay close attention to NSS patches.

Identifying Legal-intercept Requirements

The following topics provide an overview for message archiving for legal and compliance purposes. For more information, see:

- The discussion about the legal obligation to maintain strict retrievable email records in *Messaging Server System Administrator's Guide*.
- The discussion about the **imarchive** utility in *Messaging Server System Administrator's Guide*.

- The discussion about archiving messages coming into and out of the Messaging Server using the Compliance and Content Management Solution (for legacy systems only) in *Messaging Server System Administrator's Guide*.

Determine which Messaging Server capture mechanism is best to meet those requirements in your jurisdiction before responding to a compliance request.

Securing Your Archiving Needs

Once you have satisfied legal requirements, use your third-party archiving system in your jurisdiction so that it can be configured to delete messages from the archive (or make them unreadable by discarding encryption keys). Refer to "Identifying Legal-intercept Requirements" for message archiving options.

Disabling Users in Response to Abuse/Appeal Process

The following topics describe enabling and disabling users, accounts, and services in response to the abuse/appeal process. See:

- The discussion about **mailAllowedServiceAccess** in *Schema Reference*.
- The discussion about **SEND_ACCESS** and **ORIG_SEND_ACCESS** mappings in *Messaging Server Reference*.
- The discussion about enabling and disabling services at different levels in *Messaging Server System Administrator's Guide*.

Utilizing a Disk Consumption Growth Plan

Unusual disk consumption may identify some attacks on MTA relays.

For more information, see:

- The discussion about configuration options for monitoring disk and partition usage in *Messaging Server System Administrator's Guide*.
- The discussion about Message Store partitions and adding storage in *Messaging Server System Administrator's Guide*.
- The discussion about MTA performance tuning in *Messaging Server Reference*.

Preventing Unrelated Usage of Messaging Server Hosts and Virtual Machines

Oracle recommends that you do not use Messaging Server hosts or virtual machines for unrelated tasks. Single purpose hosts and virtual machines are better for securing your deployment. Be sure to turn off any unused Messaging Server services.

Determining Security Capabilities of Your Supported Mail Clients

For information on security and access control for mail clients and mail client infrastructure, refer to "Security and Access Control in Messaging Server" where the following topics are covered:

- Configuring Authentication Mechanisms in Messaging Server
- Configuring Client Access to POP, IMAP, and HTTP Services
- Configuring Encryption and Certificate-Based Authentication
- User/Group Directory Lookups Over SSL

Consider these questions when designing your Messaging Server security strategy:

- Do your mail clients support SMTP Authentication?

- Do your mail clients support standard SSL (STARTTLS on port **143** for IMAP, STLS on port **110** for POP, STARTTLS on port **587** for SMTP Submission)?
  - If not, do they support separate port SSL? (IMAPS on port **993**, POPS on port **995**, SSL SMTP Submission on standard port **465**)?
- Do you have a plan in place to handle accidental/inappropriate blacklisting of your site by reputation services?

MTA Security Guidelines

Following secure guidelines protect your MTAs from unauthorized users, large quantities of spam, reduced response time, and used up disk space and resources. "Protecting MTAs" outlines general guidelines to protect your MTAs. This section provides additional details in the following topics:

- About Messaging Server Anti-spam and Anti-virus Solutions
- Creating a Narrow Scope of MTA Relay Blocking in INTERNAL_IP Mapping Table
- Using LMTP to Connect to Inbound MTAs and in Multi-tier Deployments
- Greylisting
- Forbidding Emailing Executable Code
- Using and Configuring MeterMaid for Access Control
- Using and Configuring memcache for Access Control
- Setting MTA Recipient Limits
- Using Sieve Securely
- Using the MTA to Fix Messages from Bad Clients
- Configuring Secure ETRN Command Support

About Messaging Server Anti-spam and Anti-virus Solutions

Refer to the following topics on anti-spam and anti-virus solutions:

- Planning a Messaging Server Anti-spam and Anti-virus Strategy
- Integrating spam and virus filtering programs in *Messaging Server System Administrator's Guide*.
- Milter implementation and use in *Messaging Server Reference*.
- Using the Sender Policy Framework to detect and reject forged email in *Messaging Server System Administrator's Guide*.
- Protecting Against Email Spammers
- Blocking emails based on DNS Realtime Blocklists (RBL) data in *Messaging Server System Administrator's Guide*.

In addition, consider these guidelines:

- Make sure your domain's MX records point directly to Messaging Server's MTA and have the Messaging Server call out to anti-spam/anti-virus systems preferably through the spam plug-in or Milter mechanism.
- Filter both inbound and outbound mail.
- Consider restricting outbound port **25** to outbound MTAs only.

Creating a Narrow Scope of MTA Relay Blocking in INTERNAL_IP Mapping Table

To use the **INTERNAL_IP** mapping table for MTA Relay Blocking, refer to the following discussions:

- On preventing unauthorized users from relaying SMTP mail through your system in *Messaging Server System Administrator's Guide*.
- On mail filtering and access control in *Messaging Server System Administrator's Guide* and the **mailfromdnsverify** channel option in *Messaging Server Reference*.
- On using access control mappings to prevent users from relaying SMTP mail through your system in *Messaging Server System Administrator's Guide*.

When you run the initial Messaging Server configuration with the **configure** command, SMTP relay-blocking is enabled by default. Later, if you add channels, you might need to adjust your configuration to ensure that blocks apply to the new channels as appropriate.

Using LMTP to Connect to Inbound MTAs and in Multi-tier Deployments

Using LMTP between the relays and the back end Message Stores simplifies the deployment, which means there are fewer points of attack. For more information, see:

- The discussion about configuring the LMTP delivery mechanism in *Messaging Server System Administrator's Guide*.
- The discussion about channel configuration in *Messaging Server Reference*.
- The discussion about implementing Local Message Transfer Protocol (LMTP) for Messaging Server in *Messaging Server System Administration Guide*.

Greylisting

See "Protecting Against Email Spammers."

Forbidding Emailing Executable Code

See the discussion of the predefined conversion channel in *Messaging Server System Administrator's Guide*.

Using and Configuring MeterMaid for Access Control

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

For more information, see:

- The discussion about using MeterMaid to limit how often a particular IP address can connect to the MTA in *Messaging Server System Administrator's Guide*.
- The MeterMaid Reference in *Messaging Server System Administrator's Guide*.

Using and Configuring memcache for Access Control

memcache is a server that can provide functionality that is similar to MeterMaid. It allows you to access and manipulate data using the memcache protocol. For more information, see the discussions about **memcache** in *Messaging Server Reference*.

Setting MTA Recipient Limits

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

Using Sieve Securely

Review your **MAX_*** options settings relevant to Sieve filter limits, especially **MAX_NOTIFYS**.

> **Note:** Notify, Forward, and Redirect can potentially increase the load of generating new messages. You must consider if abusers could exploit such features by generating message loops or exponential growth of messages.

For Sieve external lists, enable setup carefully only allowing specific criteria. Some Sieve filter user education/Sieve filter creation interface guidelines to consider:

- Discourage users from attempting to personally block spam by using Sieve.
- Check that the interface generates efficient Sieves (for example: lists, wildcard matches, and so on)

Review:

- The discussion about the extensions that Messaging Server supports in *Messaging Server System Administrator's Guide*.

Using the MTA to Fix Messages from Bad Clients

If users use email clients that are especially vulnerable to buffer overruns, malicious embedding in malformed header lines, and so on, consider configuring the MTA with maximal MTA MIME processing and fixing up messages passing through the MTA with the **inner** MTA channel option.

Configuring Secure ETRN Command Support

Consider explicitly configuring the **ETRN** commands that the MTA honors. See the **ETRN_ACCESS** mapping table, the ***etrn** channel options, and the **ALLOW_ETRNS_PER_SESSION** TCP/IP channel option.

For more information, see:

- The discussion about the **ETRN_ACCESS** mapping table in *Messaging Server Reference*.
- The discussion about channel configuration in *Messaging Server Reference*.

Storing BadGuy Details in Memcached Server

When an authentication fails from a particular client IP address on messaging multiplexer (MMP), the IP address details are stored as a **BadGuy**. Subsequent authentication attempts from the same IP address are considered as bad guys and delayed for authentication.

Memcached server is a distributed memory caching system. It stores **BadGuy** information after calculating the badness of an IP address on every authentication failure. If an authentication failure is followed by a successful authentication, the successful IP address is removed from the **BadGuy** list.

Note: By default, the in-process memory system stores **BadGuy** information.

This section provides additional details in the following topics:

- Installing Memcached Server
- Configuring Bad Guys for Memcached Server
- Clearing Memcached Server Data

Installing Memcached Server

You must install and configure **memcached** server and MMP to store **BadGuy** details.

Multiple MMP processes can connect to the same **memcached** server and share the **BadGuy** information. Therefore, a **BadGuy** information of an IP address synchronizes with all MMPs and the same **BadGuy** information is accessible across all MMPs.

To install and configure **memcached** server, see:

`https://www.tecmint.com/install-memcached-on-centos-7/`.

See the discussion on configuring MMP in *Messaging Server Security Guide*.

Configuring Bad Guys for Memcached Server

Before configuring bad guys for **memcached** server:

- Ensure that **memcached** server is installed and running in a host machine. The host machine can be a local machine or a separate server where MMP is running.

Note: In a host machine, the default port is set to **11211**. However, you can set up a different port.

To configure bad guys with **memcached** server:

1. Go to the *MessagingServer_home*/**bin** directory.
2. Set **mmp.memcached_enable** value to **1**.

Note: By default, the **mmp.memcached_enable** parameter is set to **0**, in-process memory.

Table 4–1 lists the configuration parameters for setting the **memcached** port and **memcached** hostname.

Table 4–1 Memcached port and hostname configuration parameters

Configuration Parameter	Description
mmp. memcached_port set port	In Messaging Server, **memcached_port** is configured to the listening port of the **memcached** server.

Table 4–1 (Cont.) Memcached port and hostname configuration parameters

Configuration Parameter	Description
mmp.memcached_host set hostname	**memcached_host** is the hostname of the server where **memcached** server is running. If the hostname is not specified, the local hostname of an MMP is considered during the initial configuration of the MMP.

For more information, see the discussion about MMP badguy throttling and MMP connection limits in *Messaging Server Reference*.

Clearing Memcached Server Data

Memcached server stores **BadGuy** information from all MMPs. Therefore, when you refresh or restart MMP servers, the **memcached** server information is not removed. If you want to remove **memcached** server details, use one of the following commands:

- flush_all
- restart memcached

For more information on clearing **memcached** server data, see **memcached** documentation.

ENS Security Guidelines

This section describes securing ENS Server (7997) with firewall and/or TCP Access Control Filters.

Note: You can turn on ENS SSL and password based authentication. For additional information, see the discussion about ENS SSL and ENS password based authentication in *Messaging Server System Administrator's Guide*.

For more information about deploying the Event Notification Service (ENS) with Messaging Server, see *Messaging Server System Administrator's Guide*.

Note: The current implementation of ENS does not provide security on events that can be subscribed to. Thus, a user could register for all events, and portions of all other users' mail. Because of this it is strongly recommended that the ENS subscriber be on the **safe** side of the firewall at the very least.

A firewall system generally controls what TCP/IP communications are allowed between internal networks and the external world. Firewalls prevent packets considered to be unsafe from passing through.

Message Store Security Guidelines

The most important data in the Messaging Server is the data in the Message Store. Physical access and **root** access to the Message Store must be protected. "Protecting Messaging Components in Your Deployment" outlines general guidelines to protect your Message Store. In addition, you should review the discussion about managing the message store in *Messaging Server System Administrator's Guide*. This section provides additional details in the following topics:

- Securing Your Backup System
- Options for Securing Messaging Server
- Being Aware of IMAP ACLs
- Disabling IMAP Shared Folders if Not Needed

Securing Your Backup System

The process for backing up and restoring the Messaging Server is described in *Messaging Server System Administrator's Guide*.

Some security guidelines to consider for Message Store backup:

- Be sure that such a system does not leave unneeded data.
- Backup systems that encrypt data improve your security if you manage the encryption keys properly.

Options for Securing Messaging Server

The **http.feedback.notspam** (Unified Configuration) or **service.feedback.notspam** (legacy configuration), **http.feedback.spam** (Unified Configuration) or **service.feedback.spam** (legacy configuration), and **http.ipsecurity** (Unified Configuration) or **service.http.ipsecurity** (legacy configuration) options are used to secure Messaging Server. See:

- **http.feedback.notspam** (Unified Configuration) or **service.feedback.notspam** (legacy configuration) and **http.feedback.spam** (Unified Configuration) or **service.feedback.spam** (legacy configuration) in *Convergence System Administrator's Guide*.
- **http.ipsecurity** (Unified Configuration) or **service.http.ipsecurity** (legacy configuration) in "Recovering From Phishing Attacks That Have Compromised User Accounts".

Being Aware of IMAP ACLs

See the following discussions:

- Granting permission for other users to access folders in *Messaging Server System Administrator's Guide*
- Describing the tasks that you use to administer shared folders in *Messaging Server System Administrator's Guide*

Disabling IMAP Shared Folders if Not Needed

Disable shared folders if not in use. See the discussion about disabling shared folders in *Messaging Server System Administrator's Guide*.

MMP Security Guidelines

The MMP serves as a proxy for the Message Store, therefore, it needs to protect access to end user data and guard against unauthorized access. "Protecting MMPs" outlines general guidelines.

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

For more information, see the discussion about MMP badguy throttling and MMP connection limits in *Messaging Server Reference*.

User Authentication Guidelines

User authentication allows end users to securely log in through their mail clients to retrieve their mail messages. "Planning Messaging User Authentication" outlines general guidelines. This section adds the following topics:

- Acquiring SSL Server Certificates for the Server Domains
- Requiring SMTP Authentication for Mail Submission

Acquiring SSL Server Certificates for the Server Domains

Refer to "Certificate-Based Authentication for Messaging Server" for more information.

Note the following recommendations:

- Acquire SSL server certificates for server domains to which your users will connect from a third-party CA. If you also want to secure inter-deployment connections, recommended for a geographically distributed deployment and to meet legal requirements in some jurisdictions, get certificates for your Directory Servers and back-end IMAP/POP storage servers.
- Purchasing Certificate Authority (CA) service or software for your enterprise may be cost effective if you have many hosts in your deployment. Be sure to use at least 2048-bit RSA with SHA 256 signatures per current guidelines unless your jurisdiction does not permit that or some of your mail clients do not support that.
- The **certutil** provided with Solaris and Messaging Server Installer can be used with the **-g 2048** and **-Z SHA256** switches. Once enabled, you can configure SSL.

Some guidelines for SSL include:

- Having a plan for SSL certificate or CA expiration.
- Turning on SSL where required (external services, possible internal services)
- Requiring SSL where possible (**RestrictPlainPasswords**, **plaintextmincipher**)

Requiring SMTP Authentication for Mail Submission

SMTP Authentication, or **SMTP Auth** (RFC 2554) is the preferred method of SMTP submission server security. **SMTP Auth** allows only authenticated users to send mail through the MTA. For more information, see:

- The discussion about channel configuration in *Messaging Server Reference*.
- The discussion about SMTP authentication and SASL in *Messaging Server Reference*.
- The discussion about requiring password submission to login to Messaging Server in *Messaging Server System Administrator's Guide*.
- The discussion about the **authrewrite** channel option in *Messaging Server Reference*.

Message Encryption Guidelines

"Planning Message Encryption Strategies" covers S/MIME and Encryption with SSL for encryption and privacy solutions. Review the following guidelines and recommendations:

- Determining SSL Cipher Suites

- Using Solaris Crypto Framework in Place of NSS Default Software Token

Determining SSL Cipher Suites

See "Configuring Encryption and Certificate-Based Authentication".

Using Solaris Crypto Framework in Place of NSS Default Software Token

To use the cryptographic support in modern SPARC CPUs, you can configure NSS to use the Solaris Crypto Framework. See the discussion about SPARC systems and cryptographic framework in *Managing Encryption and Certificates in Oracle Solaris 11.2*.

Security Considerations for Developers

For secure programming best practices, refer to *Messaging Server MTA Developer's Reference*.

5 Using Role-Based Access Control

This chapter describes how to use role-based access control (RBAC) and the required setup for Oracle Solaris and Linux OS where privileges are available.

Overview of Role-Based Access Control

Role-based access control (RBAC), a feature in Oracle Solaris and Linux, permit non-privileged users to have access to certain privileged functionality, under certain specified circumstances. At a minimum, in Solaris, you can grant the equivalent of **setuid root** to a particular program, but only when run by a certain user. RBAC enables you to fine-tune access to privileges so that they are available in a restricted environment and only when needed.

In addition, Oracle Solaris includes privileges that give finer-grained access so that a process that requires elevated access can be granted just the minimum access necessary to satisfy its needs without having to use the traditional **UID 0** full-access. For example, a program that needs to bind to a privileged port (typically one with a port number that is less than 1024, such as port 25 for SMTP) would have needed **root** access just for that one activity. With privileges, the program can use the **net_privaddr** privilege for Solaris and **cap_net_bind_service** capability for Linux to grant the access needed to bind to the port without having full root access. By compartmentalizing privileged functions, security is greatly enhanced.

You can use RBAC for both methods, and each improves Oracle Communications Messaging Server security.

Theory of Operations

Role-based access control is managed through several files that are located in the **/etc** and **/etc/security** directories. You first create a profile that defines the new access that can be granted to the Messaging Server user account. Then you list all the special access that is granted to that profile. Finally, the Messaging Server user account is given access to the new profile.

The special access permitted by the profile is managed through intermediate commands that run the programs with the defined access. The **pfexec(1)** command is generally responsible for running a program that can then be given elevated access. **pfexec** is used by the Messaging Server **start-msg**, **stop-msg**, and **imsimta** (through the **imtacli** program) commands, and the **job_controller**, to take advantage of role-based access controls.

For more information about role-based access controls, see **rbac(5)**.

Setting Up and Using RBAC for Solaris

Caution: Implementing role-based access controls involves modifying system files that provide security definitions for the operating system and incorrect modifications may result in potential problems.

The following steps make direct modifications to files in the **/etc/security** directly, which can also be made by using the Oracle Solaris Management Console (**smc(1m)**).

Assumptions in the Examples: The following example commands assume that the Messaging Server is installed in the **/opt/sun/comms/messaging64** directory and that the Messaging Server processes are using **mailsrv** as the Unix user.

1. Copy *MessagingServer_home***/examples/rbac/MessagingServer.html** to the **/usr/lib/help/profiles/locale/C** directory. This file is referenced by the Messaging Server profile definition. For example:

```
cp /opt/sun/comms/messaging64/examples/rbac/MessagingServer.html
/usr/lib/help/profiles/locale/C
```

2. Append the contents of *MessagingServer_home***/examples/rbac/prof_attr.example** to **/etc/security/prof_attr**. This is the Messaging Server profile definition.

```
cat /opt/sun/comms/messaging64/examples/rbac/prof_attr.example >>
/etc/security/prof_attr
```

3. Edit *MessagingServer_home***/examples/rbac/exec_attr.example** to replace *msg.RootPath* with the actual path for your Messaging Server installation. For this example, instances of *msg.RootPath* are replaced with **/opt/sun/comms/messaging64**.
4. Append the contents of the edited *MessagingServer_home***/examples/rbac/exec_attr.example** to **/etc/security/exec_attr**. This defines the special permissions granted to the Messaging Server profile.

```
cat /opt/sun/comms/messaging64/examples/rbac/exec_attr.example >>
/etc/security/exec_attr
```

5. Modify the user account used by the Messaging Server to have access to this new profile.

```
usermod -P 'Oracle Communications Messaging Server' mailsrv
```

6. Modify the dispatcher process privilege, so that the dispatcher is able to successfully start. Edit the **/etc/security/exec_attr** file and add **proc_taskid**, for example:

```
Oracle Communications Messaging
Server:solaris:cmd:::/opt/sun/comms/messaging64/lib/dispatcher:privs=net_
privaddr,proc_taskid
```

7. Set the **rbac** option to 1 to fully enable RBAC usage. For example:

```
msconfig set rbac 1
msconfig show rbac
role.base.rbac = 1
```

Once the RBAC has been set up, the Messaging Server user has sufficient access so as not to require being run as **root**, to use the following commands:

- **start-msg**
- **stop-msg**
- **imsimta restart** | **shutdown** | **startup** | **stop**

Setting Up and Using RBAC for Linux

Messaging Server uses privileged ports. Therefore, the processes that non-root users start cannot bind with these ports. To allow a non-root user to perform operations on Messaging Server, you must set the **cap_net_bind_service** capability to the **effective** and **permitted** set for executable files. Then executable files acquire the capability and provide permissions to bind to the privileged ports. These elevated privileges allow non-root users to perform operations on Messaging Server.

The following example shows setting up **cap_net_bind_service** with **effective** and **permitted** set to the **imapd** executable file:

```
/usr/sbin/setcap cap_net_bind_service+ep MessagingServer_home/lib/imapd
```

For more information on permitted and effective set, see https://man7.org/linux/man-pages/man7/capabilities.7.html.

See "Messaging Server Privileges and Executable Files" for information on executable files and privileges.

Non-root users can perform the following Messaging Server operations by obtaining appropriate capabilities on Linux OS:

- Start Messaging Server
- Stop Messaging Server
- Set various msconfig options
- Execute imsimta commands

After elevating the privileges of executable files, the dynamic linker/loader or **ld.so** does not link with libraries in an untrusted path which is the location where a non-root user has set up Messaging Server. If a non-root user wants to run such executable files, the non-root user should add the Messaging Server library path to **ld.so** trusted path.

Configuring Non-Root Users with Messaging Server

You must elevate port-specific privileges to executable files to configure non-root users to start Messaging Server services.

Prerequisites

Table 5–1 lists the OS and platform compatibility requirements to configure non-root users with Messaging Server on Linux OS.

Table 5–1 Operating system and platform compatibility

Operating System	Supported versions	Minimal Kernel version
Oracle Linux 6	8.0.2.4	2.2
Oracle Linux 7	8.0.2.4 and 8.1.0.1	2.2

To configure non-root users with Messaging Server:

1. Log in as root.
2. Install Messaging Server where a non-root user is the owner.
3. Configure Messaging Server with user name of the non-root user who wants to set up RBAC configuration.
4. Set the **rbac** option to **1** to enable RBAC usage. For example:

   ```
   ./msconfig set rbac 1
   ```

5. Set the file capabilities as a root user. For more information on file capabilities, see `http://man7.org/linux/man-pages/man7/capabilities.7.html`.
6. In the **/etc/ld.so.conf.d/** location, create the **ucsmsld.conf** file.
7. Add the *MessegingServer_home*/**lib** path as a root user in the **ucsmsld.conf** file.

   Note: By default, the **lib** location of Messaging Server is */opt/sun/comms/***messaging64/lib**.

8. Run **ldconfig** to add the Messaging Server library path to dynamic linker trusted path.
9. Log in as a non-root user and execute the following command to start all the processes that are assigned to the non-root user:

   MessegingServer_home/**bin**/**start-msg**

Messaging Server Privileges and Executable Files

Messaging Server executable files installed under non-root users directory must possess Linux capabilities to start the processes.

Table 5–2 lists the Messaging Server executable files for which the privileges have to be raised by setting Linux capabilities.

Table 5–2 Executable Files and privileges

Executable File	Privilege
AService	cap_net_bind_service
dispatcher	cap_net_bind_service
imapd	cap_net_bind_service
popd	cap_net_bind_service

Note: If you use the **pipe** channel, the **pipe_master** executable requires the following capabilities to get the privilege: **cap_dac_override**, **cap_fowner**, and **cap_setuid**.

Reference Information

For more information about role-based access controls, see the following sources:

- Oracle Solaris 10 documentation: *System Administration Guide*: Security Services (Roles, Rights Profiles, and Privileges)

- man pages: **smc(1M)**, **usermod(1M)**, **prof_attr(4)**, **exec_attr(4)**, **privileges(5)**, **rbac(5)**

6 Protecting Against Email Spammers

This chapter describes how to protect Oracle Communications Messaging Server against spam, viruses, and other attacks.

Overview of Email Spammers and Compromised User Accounts

Spammers are now using sophisticated phishing attacks to target individual organizations and collect valid login details from ill-informed and overly-trusting account owners. Phishers then use these compromised account details to send spam emails by authenticating to the Messaging Server MTA and Webmail processes, thus bypassing this security restriction.

As the spam emails are delivered to external recipients, Realtime Blacklists (RBLs) are listing these sending organizations. This in turn is causing legitimate non-spam emails to be rejected by organizations that use these same RBLs.

This document provides best-practice information on how to protect your organization against phishers and compromised user accounts. It provides proactive and reactive methods to reduce the impact of compromised accounts.

Preventing Outbound Spam: Proactive Methods

Reduce the chances that a targeted phishing attack succeeds by implementing preventative measures such as:

- Educating your account holders. This is the best method to proactively avoid problems. For example, send regular reminders that your organization will *never* ask for account details by using email, and that users must immediately report such emails. Set up an appropriate role account for this task.
- Implementing anti-spam and anti-virus applications that check for phishing style email. For more information, see the discussion about integrating spam and virus filtering programs in *Messaging Server System Administrator's Guide*.
- Blocking known phishing addresses or common role accounts from sending emails from outside the organization, for example: *helpdesk@domain.com*, *security@domain.com*, and so on. For more information, see:

  `https://code.google.com/archive/p/anti-phishing-email-reply/`
- Using good password policies. Stop easy-to-guess passwords (this includes administration accounts and role accounts, that is, **uid=admin**, **calmaster**, and so on) to protect against dictionary attacks. Use password expiry to force users to change passwords on a regular basis.

- Using authenticated emails with different *From:* addresses (especially if not for the organization) increase the chances that your email accounts are used for sending out spam, or indeed used for additional phishing attacks against other organizations.

Preventing Outbound Spam: Reactive Measures

Despite the best preventative measures, spammers can still acquire valid account details. By putting in place mechanisms to limit the number of email messages that users can send, you reduce the impact of compromised accounts. You should use these limiting techniques on both outgoing and incoming email.

Blocking Submissions of Local Senders Who Might Be Spammers

When a compromised user account is used to send emails to a large number of external email addresses, it is highly probable that some of these email addresses will be invalid or trigger spam filtering mechanisms at the recipient server end and be rejected. With the **LOG_ACTION** mapping table and MeterMaid, it is possible to restrict email upload based on these rejections. For further details, see the discussion about blocking potential spammers in *Messaging Server System Administrator's Guide*.

Rate Limiting All Outgoing Email

Rate limit outgoing email as shown in this example. Use different levels of restrictions depending on the *trust* of the IP address of the client sending the email. For example:

- Most emails for internal auth-send
- Less emails for internal non-auth-send
- Less emails again for external-auth-send
- Less emails again for **mshttpd** source (Webmail emails) because for practical reasons, a human cannot send lots of emails through Webmail in a short period of time

Rate Limiting Submission Based on the Authenticated Sender

Rate limiting submission based on the authenticated sender using memcache can be configured with a single msconfig command:

```
msconfig> set mapping:FROM_ACCESS.rule "TCP|*|SMTP*|MAIL|tcp_*|*|*"
"$C$;R$[IMTA_LIB:check_memcache.so,throttle,0,memcache.example.com:22122,
sendlim -$4,10,300]$X4.2.3|$NRate$ too$ high$E"
```

This example will limit users to 10 messages every 5 minutes (300 seconds).

In this example "memcache.example.com:22122" would be replaced by the address and port of the actual memcached server in the deployment.

MeterMaid can also be configured to limit the number of messages an authenticated user can send in a number of minutes regardless of source (SMTP, Webmail).

Rate Limiting Only Outgoing Spam

Implement scanners/spam filtering on outgoing email. One idea is to use a spam filter to flag messages as *spammy*. It will also call a Sieve action that calls a mapping rule, which calls MeterMaid to monitor the count of these emails (on env-from address). If the number of emails exceeds some threshold then perform an action on the email such as: hold, capture a copy, discard, bounce, and so forth.

An example:

Configure your anti-spam scanner to add an X-header to all outbound messages that indicates whether the message is spam.

```
X-Spam-Score-Internal: ****
```

Add the following to a channel that processes your outbound mail. This will cause a sieve filter to be executed for all messages dequeued from that channel. You can also use a **destinationfilter**, depending on your environment.

```
sourcefilter file:IMTA_TABLE:authspam.filter
```

Create a sieve filter called **authspam.filter** in your config directory. It checks to see if the message is rated as spam (from the X-header) and it extracts the env-from and env-to from the message. It makes a call to a mappings table with the env-from and the env-to as arguments. It then rejects the message back to the env-from if it gets a positive response from the mappings. The next step after identifying a compromised account is to prevent further misuse of the account by spammers and address any negative consequences such as being listed on blacklist. The following techniques will provide a starting point:

```
require ["variables","reject","envelope"];

only limit messages rated as spam
if header :contains "X-Spam-Score-Internal" "****" {

    pull out the envelope from address
    if envelope :all :matches "from" "*" {
        set "FROM_ADDR" "${1}";

        pull out the envelope to address
        if envelope :all :matches "to" "*" {
            set "TO_ADDR" "${1}";

            perform FILTER_limitauthspam mapping callout
            if limitauthspam "${FROM_ADDR}|${TO_ADDR}" {
                set "RESULT" "${0}";

                reject the message
                reject "Your account has been sending a lot of messages that
appear to be spam. ";
            }
        }
    }
}
```

Put this in the mappings. The sieve script makes a call to this mapping to query MeterMaid. The mapping includes exemptions if the env-to matches recipients that you want to be able to receive spam messages from your users.

```
FILTER_limitauthspam

 *|is-spam@*                   $N$E
 *|not-spam@*                  $N$E
 *|abuse@*                     $N$E
 *|postmaster@*                $N$E
 *|*      $[IMTA_LIB:check_metermaid.so,throttle,limitauthspam,$0]$0$Y
```

Set these options to enable the MeterMaid database. This will cause MeterMaid to allow 50 outbound spam messages per hour for each env-from address. Table 6–1 shows the MeterMaid database options in both Unified Configuration and legacy configuration, and the value of each option.

Table 6–1 MeterMaid Database Options

Unified Configuration Option	Legacy Configuration Option	Value
metermaid.local_table:limitauthspam.data_type	**metermaid.table.limitauthspam.data_type**	string
metermaid.local_table:limitauthspam.quota	**metermaid.table.limitauthspam.quota**	50
metermaid.local_table:limitauthspam.quota_time	**metermaid.table.limitauthspam.quota_time**	3600
metermaid.local_table:limitauthspam.table_options	**metermaid.table.limitauthspam.options**	nocase
metermaid.local_table:limitauthspam.max_entries	**metermaid.table.limitauthspam.max_entries**	1000

Note: This will not work if the messages are not rated as spam. 419 scams are notorious for slipping through spam filters.

It is possible for the spammer to forge their env-from address. If this occurs, the sieve must be updated to accommodate. Or, do not allow outgoing email with a different *From:* address.

Reject/Discard All Outbound Spam

If your tolerance for outbound spam is high, and you do not care about the occasional message being blocked by your spam filter, rejecting or discarding all outbound spam message back to the sender is an effective way to deal with the event of a compromised account.

You may want to disable **IP reputation checks** in your spam scanner for when it processes your outbound mail since many consumer IPs will be on blacklists.

If you are rejecting the messages back to the sender, be careful that you are only rejecting mail to authenticated senders. If you want to prevent outbound mail that you are forwarding, then you should not reject the mail since it will backscatter out to the internet and get your servers blacklisted. Consider discarding or quarantining this mail instead.

Setting Up a No Phishing Zone

Experienced Messaging Server administrators know that dealing with spam is a high-priority job requiring constant attention as spammers evolve and refine their methods of attacks. Recently, many administrators have noted the rise of phishing attacks, especially against (but not exclusively) Webmail clients.

Long time Messaging Server administrators have been exchanging ideas and collaborating on all aspects of Messaging Server, including anti-spam/anti-virus techniques, by using the Info-iMS@sonnection.nl forum. In brief, this alias is the independent discussion forum for those interested in Messaging Server and all its permutations. If you are a Messaging Server administrator and haven't yet subscribed to this alias, we highly recommend that you do so.

An email thread from July 2008 highlighted the phishing problem, especially in the EDU space. Many ideas were suggested on how to combat this particular spam issue.

The following is a summary of anti-spam techniques to consider:

- Examine the sent folder to get the source IP of the submission, then **null route** the IP address on the Webmail front ends.
- Configure MeterMaid. MeterMaid limits the number of messages a user can send in a number of minutes regardless of source (SMTP, Webmail).
- Use the **imsconnutil -k -u** *uid* command to disconnect the offending user account.
- Block the offending IP address at your firewall.
- Set the **inetuserstatus** attribute for the offending user to **inactive**, change the user's password, then clear the queue(s). This technique is in response to an attack, rather than preventing or detecting the attack.
- Enable the Directory Server audit log. Monitor for changes to directory entries, such as signature files and reply-to addresses, by using a script and **crontab** to classify likely compromised accounts.
- For more information about how to deploy the Messaging Server MTA and anti-spam/anti-virus scanning systems, see the discussion about Spam and virus filtering in *Messaging Server Reference*.
- Call out to MeterMaid from the **FROM_ACCESS** mapping table, passing the user authentication as data rather than (or perhaps in addition to) calling out to MeterMaid from the **PORT_ACCESS** mapping table, passing the source IP as data. This technique limits how many messages some (authenticated) user can submit.
- Use *Postfix/Policyd*. Then change the default smtphost of Webmail to use it.
- Use this list of these password phishing reply addresses:
  `http://code.google.com/p/anti-phishing-email-reply/`
- Implement scanning systems on both incoming and outgoing email.
- Use the `https://talosintelligence.com/` database.
- You can use **LOG_ACTION** to block submissions of local senders who might be sending spam.

Recovering From Phishing Attacks That Have Compromised User Accounts

The next step after identifying a compromised account is to prevent further misuse of the account by spammers and address any negative consequences such as being listed on a real-time blacklist. The following techniques will provide a starting point:

- Prevent further logins of the comprised user account:
  - Mark account as inactive (**mailUserStatus: inactive**).
  - Change the password of the account.
  - Advise the local IT support helpdesk that access to the account has been blocked so that should the owner contact the IT help desk they can work with the customer to use improved password policies, and so on, in the future.
- Kill any existing logins by using the **imsconnutil -k -u** *uid* command.
- Block the IP address used to send the email at your network firewall.
- Kill any existing Webmail sessions to prevent re-use.

  - Increase logging to **Information**. This is required to capture the session ID information:

    Unified Configuration: **msconfig set http.logfile.loglevel Information**

    Legacy Configuration: **configutil -o logfile.http.loglevel -v Information**
  - Disable HTTP IP security. With IP security enabled, only the IP address that initially logged into the Webmail process will be able to logout.

    Unified Configuration: **msconfig set http.ipsecurity 0**

    Legacy Configuration: **configutil -o service.http.ipsecurity -v no**
  - Restart **mshttpd** processes.

    ```
    stop-msg http;start-msg http
    ```
  - If you find an account is compromised, locate the login string with the SID (session ID), for example:

    ```
    [05/May/2009:12:23:21 +1000] server httpd[7257]: Account Information:
    login [129.158.87.204:51539] user001
    plaintext sid=YvgZdFHgwx0
    ```
  - Change/reset the password for the compromised account.
  - Use **wget** to log out of the session:

    ```
    wget -o /dev/null "https://server_name/cmd.msc?sid=session ID&cmd=logout"
    for example:
    wget -o /dev/null
    "https://server1.example.com/cmd.msc?sid=YvgZdFHgwx0&cmd=logout"
    ```
- Find and remove any existing spam email sent through the compromised account in the **tcp_local** MTA queue.
- Find out if you have been blacklisted: Spamcop, Realtime Blackhole List Lookup.
  - To be able to remove yourself from a blacklist depends on the list. For example, see: `https://www.spamhaus.org` and `https://ers.trendmicro.com/`.
- Vary the IP address of your outgoing SMTP client for the **tcp_local** channel.
  - Bind outgoing email to an IP address by using the **interfaceaddress** SMTP channel option.
  - If an IP address gets blacklisted, shift to another IP address (be careful if you are using SPF).
- Enable **Directory Server audit log**: monitor for changes, such as signature files and reply-to address, by using a script and **crontab** to classify likely compromised accounts; remove modifications.

Greylisting Webmail

The following proof-of-concept instructions describe how to enable greylisting of emails that are sent through the Convergence Webmail process. Use the third-party gross daemon and plug-in to provide greylisting functionality:

`https://code.google.com/archive/p/gross`

One advantage of the gross daemon is that you can configure greylisting only if the sender's IP address is also on a blacklist.

Installing and Configuring Greylisting for Webmail

1. Download, compile, configure, and start the gross daemon. See:

   http://code.google.com/archive/p/gross/

2. Copy the **grosscheck.so** library file, compiled as part of the compilations of the gross daemon, to the MTA server's *MessagingServer_home*/**lib** directory.
3. Compile and install the **c-ares** library on the MTA server.

   Note: If the platform that is running the gross daemon is different from the MTA server platform, recompile the gross daemon to get a compatible **grosscheck.so** library.

4. Configure the MTA by creating a new file in the *MessagingServer_home*/**config** directory called **greylist.sieve** containing the following code:

```
require ["ereject","variables"];
Need to extract IP address from Received Header
Require UWC6.3p4 and above to add the Forward-For: header
if (header :matches "Received" "*(Forwarded-For: *)*") {
   set "IP_ADDR" "${2}";

   Need to extract header from address
   if (header :matches "From" "*<*>*") {
       set "FROM_ADDR" "${2}";

       Perform FILTER_GREYLIST mapping callout
       set "RESULT" greylist("${IP_ADDR}|${FROM_ADDR}|uwc");

       Block if greylist check returns a value -- indicating that they are a
'bad' sender
       if (not string :is "${RESULT}" "")
         { *NOTE* erejec is used instead of erejec(t) to workaround bug
#6704720
           erejec "Delivery failed. Please wait 10 seconds and try sending
again...";
       }
   }
}
```

5. Add the following to the *MessagingServer_home*/**config/mappings** file:

```
FILTER_GREYLIST

 ! use gross to check all triplets (client_ip,sender,recipient)
 *|*|*
$C$[IMTA_LIB:grosscheck.so,grosscheck,129.158.87.192,,5525,$0,$1,$2,]$Y$E
 * $Y
```

6. Add the following to the source channel in the *MessagingServer_home*/**config/imta.cnf** file that accepts email from the **mshttpd** process, that is, **tcp_intranet**, **tcp_uwc**:

```
sourcefilter file:IMTA_TABLE:greylist.sieve
```

7. Recompile and restart the MTA. **imsimta cnbuild; imsimta restart**

Troubleshooting Your Greylisting Deployment

1. Configure the **gross.conf** file to use a blacklist that returns a result for *all* IP addresses, for example:

   ```
   dnsbl = relays.ordb.org
   ```

2. Run the **grossd** process in the foreground, for example: **gross -d**
3. Attempt to send a test email. You should see a message similar to the following in the gross output:

   ```
   Fri Jul 18 16:34:53 2008 #9: a=greylist d=2 w=1 c=129.158.87.66 s=uwc
   r=user@example.com m=relays.ordb.org+1
   ```

   Webmail users should receive an error message in their email client such as:

   ```
   SMTP Error 5.7.1 Delivery Failed. Please wait 10 seconds and try sending again
   ```

   If users receive such an error, instruct them to Click **OK**, wait 10 seconds, then click the **Send** button again. The following message should then appear in the gross output:

   ```
   Fri Jul 18 16:42:48 2008 #a: a=match d=0 w=0 c=129.158.87.66 s=uwc
   r=user@example.com
   ```

   The email should also be accepted.

HTML Filtering in Convergence

When an **mshttpd** client requests HTML processing, that request is rejected by default. Thus, when deploying Convergence 3.0.1.1.0 and greater, you must configure Convergence to filter embedded HTML content from email messages, because such content could contain malicious code. When HTML filtering is enabled, Convergence searches incoming messages and removes specified elements, attributes, and protocols, and then permits the email to be accessed by the user. After you have configured Convergence to filter HTML, you configure Messaging Server to accept **mshttpd** client requests.

By default, in Convergence HTML filtering is disabled.

Configuring HTML filtering consists of the following tasks:

- Enabling HTML Filtering in Convergence
- Enabling Messaging Server to Accept mshttpd Client Requests

Enabling HTML Filtering in Convergence

To enable HTML filtering in Convergence, see the discussion about HTML filtering in *Convergence System Administrator's Guide*.

Enabling Messaging Server to Accept mshttpd Client Requests

You enable Messaging Server to accept **mshttpd** client requests by setting the **http.convergencefilterenabled** option to **1**.

In addition, once processing is enabled, **mshttpd** generates a log warning when a request is made to process an unknown type (that **mshttpd** does not know how to process).

Domain Keys Identified Mail (DKIM)

DKIM is a cryptographic signature-based method to authenticate email senders. With DKIM, email senders generate public and private key pairs. The public key is published to DNS records, and the matching private keys are stored in a sender's outbound email servers.

See the discussion about DKIM (Doc ID 2681977.1) on the My Oracle Support Web site: `https://support.oracle.com/portal/`.

7

Security and Access Control in Messaging Server

Oracle Communications Messaging Server supports security features that enable you to keep messages from being intercepted, prevent intruders from impersonating your users or administrators, and permit only specific users access to specific parts of your messaging system.

The Messaging Server security architecture is part of the security architecture of Oracle servers as a whole. It is built on industry standards and public protocols for maximum interoperability and consistency.

About Server Security

Server security encompasses a broad set of topics. In most enterprises, ensuring that only authorized people have access to the servers, that passwords or identities are not compromised, that people do not misrepresent themselves as others when communicating, and that communications can be held confidential when necessary are all important requirements for a messaging system.

Perhaps because the security of server communication can be compromised in many ways, there are many approaches to enhancing it. This information focuses on setting up encryption, authentication, and access control. It discusses the following security-related Messaging Server topics:

- **User ID and password login:** Requiring users to enter their user IDs and passwords to log in to IMAP, POP, HTTP, or SMTP, and the use of SMTP password login to transmit sender authentication to message recipients.
- **Encryption and authentication:** Using the TLS and SSL protocols to encrypt communication and authenticate clients.
- **Administrator access control:** Using the access-control facilities to delegate access to a Messaging Server and some of its individual tasks.
- **TCP client access control:** Using filtering techniques to control which clients can connect to your server's POP, IMAP, HTTP, and authenticated SMTP services.

Not all security and access issues related to Messaging Server are treated in this chapter. Security topics that are discussed elsewhere include the following:

- **Physical security:** Without provisions for keeping server machines physically secure, software security can be meaningless.
- **Message-store access:** You can define a set of message-store administrators for the Messaging Server. These administrators can view and monitor mailboxes and can

control access to them. For details, see the discussion about message store management in *Messaging Server System Administrator's Guide*.

- **End-user account configuration:** End-user account information can be primarily maintained by using the Delegated Administrator product.
- **Filtering unsolicited bulk email (UBE):** See the discussion about mail filtering and access control in *Messaging Server System Administrator's Guide*.
- Secure Multipurpose Internet Mail Extensions (S/MIME) is described in the discussion about administering SMIME in Convergence in *Convergence System Administrator's Guide*.

About HTTP Security

Messaging Server includes an http-based service that provides a custom protocol called WMAP. This service is only used by the Convergence web client, runs on ports 8990/8991 and should be disabled if Convergence is not used. For more information about http access to email and security thereof, see the documentation for Convergence.

Configuring Authentication Mechanisms in Messaging Server

An authentication mechanism is a particular method for a client to prove its identity to a server. Messaging Server supports authentication methods defined by the Simple Authentication and Security Layer (SASL) protocol and it supports certificate-based authentication. The SASL mechanisms are described in this information. For more information about certificate-based authentication, see "Configuring Encryption and Certificate-Based Authentication".

Overview

Messaging Server supports the following SASL authentication methods for password-based authentication.

- **PLAIN.** This mechanism passes the user's plaintext password over the network, where it is susceptible to eavesdropping. Secure Sockets Layer (SSL) can be used to alleviate the eavesdropping problem. For more information, see "Configuring Encryption and Certificate-Based Authentication".
- **APOP.** A challenge/response authentication mechanism that can be used only with the POP3 protocol. Defined in RFC 1939. Should be used only by sites that have legacy usage of this mechanism.
- **CRAM-MD5.** A challenge/response authentication mechanism similar to APOP, but suitable for use with other protocols as well. Defined in RFC 2195. Should be used only by sites that have legacy usage of this mechanism.
- **LOGIN.** This is equivalent to PLAIN and exists only for compatibility with pre-standard implementations of SMTP authentication. This mechanism is only enabled for use by SMTP.

With a challenge/response authentication mechanism, the server sends a challenge string to the client. The client responds with a hash of that challenge and the user's password. If the client's response matches the server's own hash, the user is authenticated.

Note: The POP, IMAP, and SMTP services support all SASL mechanisms. The HTTP service supports only the plaintext password mechanism.

Table 7–1 shows some SASL and SASL-related configuration options. For the complete listing of options, see *Messaging Server Reference*.

Table 7–1 Some SASL and SASL-related Options

Option	Description
auth.has_plain_passwords	Boolean to indicate that directory stores plaintext passwords which enables APOP and CRAM-MD5. Default: 0 (False)
auth.auto_transition	Boolean. When set and a user provides a plain text password, the password storage format is transitioned to the default password storage method for the directory server. This can be used to migrate from plaintext passwords to APOP and CRAM-MD5. Default: 0 (False)
imap.allowanonymouslogin	This enables the SASL ANONYMOUS mechanism for use by IMAP. Default: 0 (False)
imap | pop | smtp | http.plaintextmincipher	If this is greater than 0, then disable use of plaintext passwords unless a security layer (SSL or TLS) is activated. This forces users to enable SSL or TLS on their client to log in, which prevents exposure of their passwords on the network. The MMP has an equivalent option **restrictplainpasswords**. Default: 0
auth.searchfilter	This is the default search filter used to look up users when one is not specified in the **inetDomainSearchFilter** for the domain. The syntax is the same as **inetDomainSearchFilter**. (See *Schema Reference*). Default: **(&(uid=%U)(objectclass=inetmailuser))**

To Configure Access to Plaintext Passwords

To work, the CRAM-MD5 or APOP authentication methods require access to the users' plaintext passwords. You must perform the following steps:

1. Configure Directory Server to store passwords in cleartext.
2. Configure Messaging Server so that it knows Directory Server is using cleartext passwords.

To Configure Directory Server to Store Cleartext Passwords

To enable the CRAM-MD5 or APOP mechanisms, you must configure the Directory Server to store passwords in cleartext. If you are using a Directory Server prior to version 6, the following instructions apply. (For version 6 or later, refer to the latest Directory Server documentation.)

1. In the **Directory Server Console**, open the Directory Server you want to configure.
2. Click the **Configuration** tab.
3. Open **Data** in the left pane.
4. Click **Passwords** in the right pane.
5. From the Password encryption drop-down list, choose **cleartext**.

Note: This change only impacts users created in the future. You must transition or reset existing users' passwords after this change.

To Configure Messaging Server for Cleartext Passwords

You can configure Messaging Server so that it knows the Directory Server is able to retrieve cleartext passwords. This makes it safe for Messaging Server to advertise APOP and CRAM-MD5:

1. Enable the **auth.has_plain_passwords** option.

   ```
   msconfig set auth.has_plain_passwords 1
   ```

2. To disable these challenge/response SASL mechanisms, disable the **auth.has_plain_passwords** option (set the value to 0).

Note: Existing users cannot use APOP and CRAM-MD5 until their password is reset or migrated (see "Transitioning Users").

Transitioning Users

You can specify information about transitioning users. An example would be if a user password changes or if a client attempts to authenticate with a mechanism for which they do not have a proper entry.

- Set the **auth.auto_transition** option.

  ```
  msconfig set auth.auto_transition value
  ```

  For the value, you can specify one of the following:

  – **0** - Do not transition passwords. This is the default.

  – **1** - Do transition passwords.

To successfully transition users, you must set up ACIs in the Directory Server that enable Messaging Server write access to the user password attribute. To do this, perform the steps in the following task.

To Transition Users

If you are using a Directory Server prior to version 6, the following instructions apply. (For version 6 or later, refer to the latest Directory Server documentation.)

1. In Console, open the Directory Server you want to configure.
2. Click the **Directory** tab.
3. Select the base suffix for the user/group tree.
4. From the **Object** menu, select **Access Permissions**.
5. Select the ACI for **Messaging Server End User Administrator Write Access Rights**.
6. Click **ACI Attributes**.
7. Add the **userpassword** attribute to the list of existing attributes.
8. Click **OK**.

Configuring Client Access to POP, IMAP, and HTTP Services

Messaging Server provides two ways to control client access, one for non-dispatcher services and one for dispatcher services. This information describes the TCP client access control mechanism used by non-dispatcher services, like the IMAP and POP servers. The proxy servers, MMP and **mshttpd**, also use this mechanism (by using options such as **popproxy.domainallowed**, **imapproxy.domainallowed**, and so on). The internal ENS server uses the mechanism as well, but does not perform authentication and thus does not use LDAP attributes. See the discussion of how to filter mail based on its source or header strings in *Messaging Server System Administrator's Guide*.

Note: The MMP behaves differently with respect to access control than the other services. For example, the MMP **IMAP** service controls both IMAP and IMAP+SSL services (that is, controls both ports 143 and 993). The other Messaging Server services treat IMAP and IMAP+SSL as separate services, that is, IMAP+SSL on port 993 has its own access control that is separate from IMAP on port 143.

If you are managing messaging services for a large enterprise or an Internet service provider, be sure to also implement protection from spammers and DNS spoofers to improve the general security of your network. See "Protecting Against Email Spammers" for more information.

If controlling access by IP address is not an important issue for your enterprise, you do not have to create any filters. If minimal access control is all you need, see "Mostly Allowing" for instructions.

How Client Access Filters Work

The Messaging Server access-control facility for TCP clients is an implementation of the TCP wrapper concept. A TCP wrapper is a program that listens at the same port as the TCP daemon it serves. It uses access filters to verify client identity, and it gives the client access to the daemon if the client passes the filtering process. The design of the Messaging Server TCP wrapper is based on the Unix **Tcpd** access-control facility (created by Wietse Venema).

As part of its processing, the Messaging Server TCP client access-control system performs (when necessary) the following analyses of the socket end-point addresses:

- Reverse DNS lookups of both end points (to perform name-based access control)
- Forward DNS lookups of both end points (to detect DNS spoofing)

The system compares this information against access-control statements called **filters** to decide whether to grant or deny access. For each service, separate sets of Allow filters and Deny filters control access. Allow filters explicitly grant access. Deny filters explicitly forbid access.

When a client requests access to a service, the access-control system compares the client's address or name information to each of that service's filters, in order, by using these criteria:

- The search stops at the first match. Because Allow filters are processed before Deny filters, Allow filters take precedence.
- Access is granted if the client information matches an Allow filter for that service.
- Access is denied if the client information matches a Deny filter for that service.

- If no match with any Allow or Deny filter occurs, access is granted, except in the case where there are Allow filters but no Deny filters, in which case lack of a match means that access is denied.

The filter syntax described here is flexible enough that you should be able to implement many different kinds of access-control policies in a simple and straightforward manner. You can use both Allow filters and Deny filters in any combination, even though you can probably implement most policies by using almost exclusively Allows or almost exclusively Denies.

The following sections describe filter syntax in detail and give usage examples. "To Create Access Filters for Services" gives the procedure for creating access filters.

Filter Syntax

Filter statements contain both service information and client information. The service information can include the name of the service, names of hosts, and addresses of hosts. The client information can include host names and host addresses. Both the server and client information can include wildcard names or patterns.

The very simplest form of a filter is:

service:hostSpec

where *service* is the name of the service (such as **SMTP**, **POP**, **IMAP**, or **HTTP**) and *hostSpec* is the host name, IPv4 address, or wildcard name or pattern that represents the client requesting access. When a filter is processed, if the client seeking access matches *client*, access is either allowed or denied (depending on which type of filter this is) to the service specified by *service*. Here are some examples:

```
imap: roberts.newyork.example.com
pop: ALL
http: ALL
```

If these are Allow filters, the first one grants the host **roberts.newyork.example.com** access to the IMAP service, and the second and third grant all clients access to the POP and HTTP services, respectively. If they are Deny filters, they deny those clients access to those services. For descriptions of wildcard names such as **ALL**, see "Wildcard Names".

Either the server or the client information in a filter can be somewhat more complex than this, in which case the filter has the more general form of:

serviceSpec:clientSpec

Where *serviceSpec* can be either *service* or *service@hostSpec*, and *clientSpec* can be either *hostSpec* or *user@hostSpec*. Where *user* is the user name (or a wildcard name) associated with the client host seeking access. Here are two examples:

```
pop@mailServer1.example.com: ALL
imap: srashad@xyz.europe.example.com
```

If these are Deny filters, the first filter denies all clients access to the SMTP service on the host **mailServer1.example.com**. The second filter denies the user **srashad** at the host **xyz.europe.example.com** access to the IMAP service.

Finally, at its most general, a filter has the form:

serviceList:clientList

where *serviceList* consists of one or more *serviceSpec* entries, and *clientList* consists of one or more *clientSpec* entries. Individual entries within *serviceList* and *clientList* are separated by blanks and/or commas.

In this case, when a filter is processed, if the client seeking access matches any of the *clientSpec* entries in *clientList*, then access is either allowed or denied (depending on which type of filter this is) to all the services specified in *serviceList*. Here is an example:

```
pop, imap, http: .europe.example.com .newyork.example.com
```

If this is an Allow filter, it grants access to POP, IMAP, and HTTP services to all clients in either of the domains **europe.example.com** and **newyork.example.com**. For information on using a leading dot or other pattern to specify domains or subnet, see "Wildcard Patterns".

You can also use the following syntax:

"+" or "-" *serviceList*:**$next_rule*

+ (allow filter) means the daemon list services are being granted to the client list.

- (deny filter) means the services are being denied to the client list.

* (wildcard filter) allow all clients to use these services.

$ separates rules.

The following example enables multiple services on all clients.

```
+imap,pop,http:*
```

The following example shows multiple rules, but each rule is simplified to have only one service name and uses wildcards for the client list. (This is the most commonly used method of specifying access control in LDIF files.)

```
+imap:ALL$+pop:ALL$+http:ALL
```

An example of how to disallow all services for a user is:

```
-imap:*$-pop:*$-http:*
```

Wildcard Names

Table 7–2 shows the wildcard names that represent service names, host names or addresses, or user names:

Table 7–2 Wildcard Names for Service Filters

Wildcard Name	Explanation
ALL, *	The universal wildcard. Matches all names.
LOCAL	Matches any local host (one whose name does not contain a dot character). However, if your installation uses only canonical names, even local host names will contain dots and thus will not match this wildcard.
UNKNOWN	Matches any host whose name or address is unknown. Use this wildcard name carefully. Host names may be unavailable due to temporary DNS server problems - in which case all filters that use *UNKNOWN* will match all client hosts. A network address is unavailable when the software cannot identify the type of network it is communicating with - in which case all filters that use *UNKNOWN* will match all client hosts on that network.

Table 7–2 (Cont.) Wildcard Names for Service Filters

Wildcard Name	Explanation
KNOWN	Matches any host whose name *and* address are known. Use this wildcard name carefully: Host names may be unavailable due to temporary DNS server problems - in which case all filters that use *KNOWN* will fail for all client hosts. A network address is unavailable when the software cannot identify the type of network it is communicating with - in which case all filters that use *KNOWN* will fail for all client hosts on that network.
DNSSPOOFER	Matches any host whose DNS name does not match its own IP address.

Wildcard Patterns

You can use the following patterns in service or client addresses:

- A string that begins with a dot character (.). A host name is matched if the last components of its name match the specified pattern. For example, the wildcard pattern *.example.com* matches all hosts in the domain *example.com*.
- A string of the form *n.n.n.n/m.m.m.m*. This wildcard pattern is interpreted as a *net/mask* pair. A host address is matched if *net* is equal to the bitwise AND of the address and *mask*. For example, the pattern **123.45.67.0/255.255.255.128** matches every address in the range **123.45.67.0** through **123.45.67.127**.
- A string of the form *n.n.n.n/p*. This wildcard pattern is interpreted as a **CIDR** where *p* is the routing prefix. The corresponding subnet mask, *mask*, is *p* one bits followed by 32-*p* zero bits for a total of 32 bits. A host address is matched if the bitwise AND of *n.n.n.n* and *mask* is equal to the bitwise AND of the address and *mask*. For example, the pattern **123.45.67.0/25** matches every address in the range **123.45.67.0** through **123.45.67.127**.

EXCEPT Operator

The access-control system supports a single operator. You can use the **EXCEPT** operator to create exceptions to matching names or patterns when you have multiple entries in either *serviceList* or *clientList*. For example, the expression:

*list1***EXCEPT***list2*

means that anything that matches **list1** is matched, *unless* it also matches **list2**.

Here is an example:

```
ALL: ALL EXCEPT isserver.example.com
```

If this were a Deny filter, it would deny access to all services to all clients except those on the host machine **isserver.example.com**.

EXCEPT clauses can be nested. The expression:

```
list1 EXCEPT list2 EXCEPT list3
```

is evaluated as if it were:

```
list1 EXCEPT (list2 EXCEPT list3)
```

Server-Host Specification

You can further identify the specific service being requested in a filter by including server host name or address information in the *serviceSpec* entry. In that case the entry has the form *service@hostSpec*.

You might want to use this feature when your Messaging Server host machine is set up for multiple Internet addresses with different Internet host names. If you are a service provider, you can use this facility to host multiple domains, with different access-control rules, on a single server instance.

Filter Examples

The examples in this section show a variety of approaches to controlling access. In studying the examples, keep in mind that Allow filters are processed before Deny filters, the search terminates when a match is found, and access is granted when no match is found at all.

The examples listed here use host and domain names rather than IP addresses. Remember that you can include address and netmask information in filters, which can improve reliability in the case of name-service failure.

Mostly Denying

In this case, access is denied by default. Only explicitly authorized hosts are permitted access.

The default policy (no access) is implemented with a single, trivial deny file:

```
ALL: ALL
```

This filter denies all service to all clients that have not been explicitly granted access by an Allow filter. The Allow filters, then, might be something like these:

```
ALL: LOCAL @netgroup1
ALL: .example.com EXCEPT externalserver.example.com
```

The first rule permits access from all hosts in the local domain (that is, all hosts with no dot in their host name) and from members of the group **netgroup1**. The second rule uses a leading-dot wildcard pattern to permit access from all hosts in the **example.com** domain, with the exception of the host **externalserver.example.com**.

Mostly Allowing

In this case, access is granted by default. Only explicitly specified hosts are denied access.

The default policy (access granted) makes Allow filters unnecessary. The unwanted clients are listed explicitly in Deny filters such as these:

```
ALL: externalserver.example1.com, .example.asia.com
ALL EXCEPT pop: contractor.example1.com, .example.com
```

The first filter denies all services to a particular host and to a specific domain. The second filter permits nothing but POP access from a particular host and from a specific domain.

Denying Access to Spoofed Domains

You can use the **DNSSPOOFER** wildcard name in a filter to detect host-name spoofing. When you specify **DNSSPOOFER**, the access-control system performs forward or reverse DNS lookups to verify that the client's presented host name matches its actual IP address. Here is an example for a Deny filter:

```
ALL: DNSSPOOFER
```

This filter denies all services to all remote hosts whose IP addresses do not match their DNS host names.

Controlling Access to Virtual Domains

If your messaging installation uses virtual domains, in which a single server instance is associated with multiple IP addresses and domain names, you can control access to each virtual domain through a combination of Allow and Deny filters. For example, you can use Allow filters like:

```
ALL@msgServer.example1.com: @.example1.com
ALL@msgServer.example2.com: @.example2.com
...
```

coupled with a Deny filter like:

```
ALL: ALL
```

Each Allow filter permits only hosts within *domainN* to connect to the service whose IP address corresponds to **msgServer.example*N*.com**. All other connections are denied.

Controlling IMAP Access While Permitting Access to Webmail

If you want to allow users to access Webmail, but not access IMAP, create a filter like this:

+imap:access_server_host,access_server_host

This permits IMAP *only* from the access server hosts. You can set the filter at the IMAP server level by using **imap.domainallowed**, or at the domain/user level with LDAP attributes.

To Create Access Filters for Services

You can create Allow and Deny filters for the IMAP, POP, or HTTP services. You can also create them for SMTP services, but they have little value because they only apply to authenticated SMTP sessions. See the discussion of how to filter mail based on its source or header strings in *Messaging Server System Administrator's Guide*.

To Create Filters by Using the Command Line

You can also specify access and deny filters at the command line as follows:

- To create or edit access filters for services:

  ```
  msconfig set service.domainallowed filter
  ```

  where *service* is **POP**, **IMAP**, or **HTTP** and *filter* follows the syntax rules described in "Filter Syntax".

- To create or edit deny filters for services:

  ```
  msconfig set service.domainnotallowed filter
  ```

  where *service* is **POP**, **IMAP**, or **HTTP** and *filter* follows the syntax rules described in "Filter Syntax". For a variety of examples, see "Filter Examples".

- Restart the relevant services when you make changes to their access filters.

Configuring Encryption and Certificate-Based Authentication

This information describes encryption and certificate-based authentication for Messaging Server in a Unified Configuration.

Encryption and Certificate-Based Authentication Overview

Messaging Server uses the Transport Layer Security (TLS) protocol, otherwise known as the Secure Sockets Layer (SSL) protocol, for encrypted communications and for certificate-based authentication of clients and servers. Messaging Server supports SSL versions 3.0 and 3.1. TLS is fully compatible with SSL and includes all necessary SSL functionality.

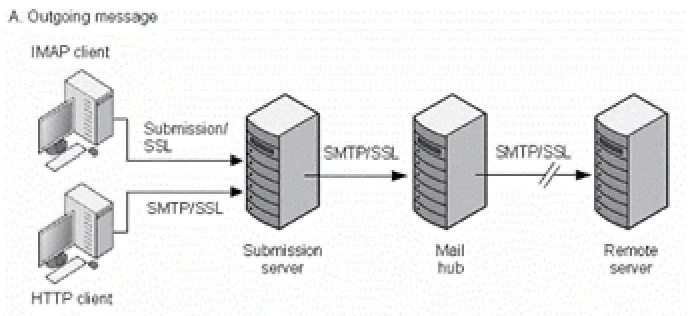
If transmission of messages between a Messaging Server and its clients and between the server and other servers is encrypted, there is a reduced chance for eavesdropping on the communications. If connecting clients are authenticated, there is also a reduced chance for intruders to impersonate (spoof) them.

SSL functions as a protocol layer beneath the application layers of IMAP4, HTTP, POP3, and SMTP. If SSL is needed with SMTP, it is normally handled on the standard SMTP relay port (25) or the standard SMTP submission port (587) with the STARTTLS command. Alternatively, SMTP submission with SSL can be handled by the port that is often used for SSL submission (465).

HTTP and HTTP/SSL require different ports. IMAP and IMAP/SSL, and POP and POP/SSL can use the same port or different ports. SSL acts at a specific stage of message communication for both outgoing and incoming messages.

Figure 7–1 shows the route of an encrypted outgoing message with Messaging Server.

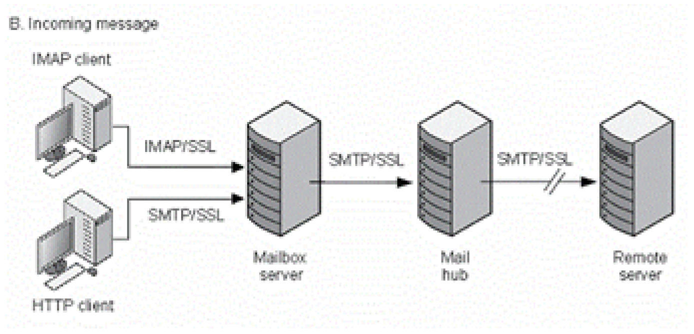
Figure 7–1 Outgoing Encrypted Communications with Messaging Server



Note: To enable encryption for outgoing SMTP connections, you must modify the channel definition to include the **tls** channel keywords, such as **maytls**, **musttls**, and so on.

Figure 7–2 shows the route of an encrypted incoming message with Messaging Server.

Figure 7–2 Incoming Encrypted Communications with Messaging Server



SSL provides hop-to-hop encryption, but the message is not encrypted on each intermediate server.

There is a small performance cost to consider in setting up an SSL connection when planning server capacity. To help reduce this cost, cryptographic accelerators can be used.

Obtaining Certificates

Whether you use SSL for encryption or for authentication, you must obtain a server certificate for your Messaging Server. The certificate identifies your server to clients and to other servers. For more information on **certutil**, see: `https://www.mozilla.org/projects/security/pki/nss/tools/certutil.html`.

This section consists of the following subsections:

- To Manage Internal and External Modules
- Creating a Password File
- Obtaining and Managing Certificates

To Manage Internal and External Modules

A server certificate establishes the ownership and validity of a key pair, the numbers used to encrypt and decrypt data. Your server's certificate and key pair represent your server's identity. They are stored in a certificate database that can be either internal to the server or on an external, removable hardware card (smartcard).

Oracle servers access a key and certificate database using a module conforming to the Public-Key Cryptography System (PKCS) #11 API. The PKCS #11 module for a given hardware device is usually obtained from its supplier and must be installed into the Messaging Server before the Messaging Server can use that device. The pre-installed **Internal PKCS # 11 Module** supports a single internal software token that uses the certificate database that is internal to the server.

Setting up the server for a certificate involves creating a database for the certificate and its keys and installing a PKCS #11 module. If you do not use an external hardware token, you create an internal database on your server, and you use the internal, default module that is part of Messaging Server. If you do use an external token, you connect a hardware smartcard reader and install its PKCS #11 module.

You can manage PKCS #11 modules, whether internal or external, through **modutil**:

1. Connect a hardware card reader to the Messaging Server host machine and install drivers.

2. To install the PKCS #11 module for the installed driver, use the **modutil** command found in *MessagingServer_home* **/lib**.

> **Note:** The main use for the **modutil** command is to switch from NSS security to Solaris Cryptographic Framework PKCS #11 provider.

Creating a Password File

You can create a password file for the certificate or certificate database, to avoid having to type it multiple times (it is needed by at least three server processes), and to facilitate unattended server restarts.

Passwords themselves are generated when their certificate database is created, for example, when using the **certutil** command.

In Unified Configuration, SSL passwords for key files are stored in the **xpass.xml** file. Use the **msconfig set -prompt "sectoken:Internal (Software) Token.tokenpass"** command to change. This command causes the **msconfig** command to prompt for the password without an echo.

> **Caution:** Because the administrator is not prompted for the module password at server startup, it is especially important that you ensure proper administrator access control to the server and proper physical security of the server host machine and its backups.

Obtaining and Managing Certificates

Whether you use SSL for encryption or for authentication, you must obtain a server certificate for your Messaging Server. The certificate identifies your server to clients and to other servers. Use the **certutil** command to manage certificates and key databases. Use **certutil** with appropriate options (**-d** followed by a directory location with a **sql:** prefix and **-Z SHA256 -g 2048**), or other third-party certificate generation tools, to create certificates and certificate requests with up-to-date security strength.

To run SSL on Messaging Server, you must either use a self-signed certificate or a Public Key Infrastructure (PKI) solution which involves an external Certificate Authority (CA). For a PKI solution, you need a CA-signed server certificate that contains both a public and a private key. This certificate is specific to one Messaging Server. You also need a trusted CA certificate, which contains a public key. The trusted CA certificate ensures that all server certificates from your CA are trusted. This certificate is sometimes also called a CA root key or root certificate. For instructions on how to create and install a self-signed CA certificate and key, see "SSL/TLS Tasks".

Implementing Secure Connections Using Two Different Certificate Authorities (CAs)

You can implement SSL connections between server and clients, for example, from Messaging Server, and to other servers in your deployment as well (Web Server, Calendar Server, Directory Server). If desired, you can use two different Certificate Authorities (CAs), one for the server and one for the client.

In such a scenario, you can use one CA to issue server certificates, and another CA to issue client certificates. If you want the client to accept the server's certificate as genuine, you must load the CA certificate for the server into the client's certificate database. If you want the server to accept the client as genuine, you must load the CA certificate for the client into the server's certificate database.

To Enable SSL and Selecting Ciphers

SSL certificates can be installed and configured by using the **certutil** utility and by running the appropriate **msconfig** commands to enable SSL for the required services.

About Ciphers

A **cipher** is the algorithm used to encrypt and decrypt data in the encryption process. SSL cipher suites control the level of protection required between SSL client and server. Different cipher suites have different properties and use different cryptographic algorithms. At any time a specific cryptographic algorithm might be weakened or compromised by new research in cryptography. The ability to change the default cipher suites allows the software to adapt as security technology changes. As CPUs get faster, the key size necessary to provide several years of comfortable protection increases, even if the algorithm is considered state-of-the-art.

The default set of SSL cipher suites used may change over time as more secure ones are introduced and weaker ones are deprecated. It is expected most deployments will be happy with the default set of cipher suites and it is generally not a good idea to adjust the available cipher suites without reason.

However, here are some scenarios where it may be helpful to adjust the cipher suites:

1. A site with specific security policies may want to provide a fixed list of cipher suites to use that is set by site policy rather than simply using state-of-the-art suites provided by the NSS library. Such a site would typically configure this setting to '**-ALL,...**' where '...' contains the cipher suite names.
2. A site which is experimenting with higher performance or more secure cipher suites that require installation of special server certificate types, for example, the elliptic curve cipher suites. Such a site would enable these additional suites once installation was complete using a setting such as '**+TLS_ECDH_RSA_WITH_AES_128_CBC_SHA**' to enable an ECDH_RSA cipher suite from RFC 4492.
3. If a site is forced to continue supporting a particularly old client that only supports weak cipher suites, they can be explicitly enabled (for example '**WEAK+DES**' enables the single-DES cipher suites.
4. In the event the cryptographic research community discovers a vulnerability in one or more of the ciphers enabled by default, this provides a mechanism to immediately disable those ciphers.

SSL Ciphers for Messaging Server

See the **ssladjustciphersuites** option in *Messaging Server Reference* for a list of available cipher suites.

This list excludes the SSL2 cipher suites as Messaging Server has not supported SSL2 since the Messaging Server 6.0 release. While the standard names for cipher suites (as published in TLS RFCs) are preferred, there is limited support for legacy names used in previous releases and for some OpenSSL names.

The following configuration options can be used to turn on all cipher suites excluding the weak ones:

- For all services: **base.ssladjustciphersuites "all"**
- For individual services: *service***.ssladjustciphersuites "all"** where *service* is **imapproxy**, **popproxy**, or **MMP**

However, be advised to instead only turn on the specific cipher suite needed for interoperability. The 56-bit ciphers are not as weak as the 40-bit ciphers so if it's possible to only enable those, the following cipher suite works: +TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA.

Adjusting the SSL Ciphers for Messaging Server

In Unified Configuration, the **.cfg** files are no longer used, and the configuration is stored in the Unified Configuration itself. In Unified Configuration, you enable and modify the configuration by running the **msconfig** command to set the appropriate options in the **imapproxy**, **popproxy**, and **smtpproxy** configuration groups. For example, in Unified Configuration, the following command sets the SSL Cipher suite for the IMAP proxy:

```
msconfig set imapproxy.ssladjustciphersuites cipher suite
```

Specify SSL Certificate

- To specify the SSL certificate to be presented by Messaging Server, run the following command:

  ```
  msconfig set base.sslnicknames certname
  ```

  For example:

  ```
  msconfig set base.sslnicknames "Server-Cert"
  ```

There is also a per-service configuration setting for the SSL server certificate nickname. The settings are as follows:

mta.sslnicknames for the SMTP and Submit servers, **imap.sslnicknames** for the IMAP server, **pop.sslnicknames** for the POP server, and **http.sslnicknames** for Webmail server.

The **.sslnicknames* options have the same meaning as (and override) the *base.sslnicknames* option. Specifically, this is a comma-separated list of NSS certificate nicknames. Although more than one nickname is permitted in the list, each nickname must refer to a different type of certificate (for example, an RSA certificate and a DSS certificate) so the setting will almost always be only one nickname. A nickname can be unqualified in which case the NSS software token or default token will be searched, or it can have the form *security-module*:*nickname* in which case the specified security module will be searched for that nickname. This is needed for certificates stored in hardware tokens or places other than the default NSS database.

This does not permit the use of more than one NSS software token in the product. In particular, there is only one **cert8.db** and **key3.db** for IMAP, POP, SMTP, and HTTP. NSS does not permit that.

Note: To enable SSL encryption for outgoing messages, you must modify the channel definition to include the **tls** channel options, such as **maytls**, **musttls**, and so on.

Configuring Individual Messaging Processes for SSL

This section describes the procedures you use to configure the various Messaging Server processes for SSL.

To Configure MMP for SSL

- Use the **msconfig** command to set the following configuration options to enable SSL:

```
msconfig set mmp.enable 1
msconfig set imapproxy.tcp_listen:imapproxy1.ssl_ports 993
msconfig set popproxy.tcp_listen:popproxy1.ssl_ports 995
```

To Configure IMAP for SSL

- Use the **msconfig** command to set the following configuration options to enable SSL:

```
msconfig imap.enablesslport 1
msconfig imap.enable 1
msconfig imap.sslport 993
msconfig imap.sslusessl 1
```

To Configure POP for SSL

- Use the **msconfig** command to set the following configuration options to enable SSL:

```
msconfig pop.enablesslport 1
msconfig pop.enable 1
msconfig pop.sslport 995
msconfig pop.sslusessl 1
```

To Configure HTTP for SSL

- Use the **msconfig** command to set the following configuration options to enable SSL:

```
msconfig http.enablesslport 1
msconfig http.enable 1
msconfig http.sslport 443
msconfig http.sslusessl 1
```

To Configure SMTP for SSL

1. To enable SSL encryption for outgoing messages, modify the channel definitions to include the TLS channel options such as **maytls**, **musttls**, and so on.
2. (Optional) Use the **msconfig** command to support SMTP submission with SSL on port 465 instead of the default port 587, which is enabled when a server certificate is installed and often used for SMTP submission with STARTTLS:

```
msconfig set dispatcher.service:SMTP_SUBMIT.ssl_ports 465
```

To Verify the SSL Configuration

1. Use the **netstat** command to verify that the service is running.

```
netstat -an | grep service.sslport
```

   where:*service* is a keyword **MMP**, **IMAP**, **POP**, **HTTP**, or **SMTP**. For example:

```
msconfig show imap.sslport
993
netstat -an | grep 993
      *.993              *.*                0      0 49152      0 LISTEN
```

2. Check for errors in the Messaging Server log files. Log files are located in the *MessagingServer_home*/**log** directory. For example, check the **IMAP** log for SSL initialization errors (ASockSSL_Init errors).

Configuring Indexed Search Converter for SSL

You can configure the Indexed Search Convertor (ISC) component of Cassandra message store to use SSL. To enable SSL, you set the appropriate Messaging Server configuration options and add a certificate into the Java keystore. You can use either a self-signed or CA-signed certificate. In addition, you must use the **certutil** command to import the same self-signed certificate into the Messaging Server NSS certificate database as a trusted authority for LMTP to connect to ISC over SSL. If you use a self-signed certificate, you must export and copy it into the Field Input Transformer (FIT) node's **config** directory.

Note: Unlike other Messaging Server components, the ISC and FIT components support Java keystore instead of the NSS certificate database. Thus, you must use the **keytool** command to manage certificates for the ISC and FIT.

Configuring ISC for SSL Using a Self-Signed Certificate

To configure the ISC for SSL using a self-signed certificate:

1. On ISC nodes, enable SSL by setting the following configuration options:

```
msconfig
msconfig> set isc.sslusessl 1
msconfig# set base.ssljkspath path_to_Java_keystore
msconfig# set base.ssljkspassword
Password: Java_keystore_password
Verify:
msconfig# write
msconfig> exit
```

2. On FIT nodes, enable SSL by setting the **fit.sslusessl** configuration option:

```
cd /opt/sun/comms/cas/bin
config-fit --sslusessl 1
```

3. On message access layer nodes (in general, the same node as ISC node), enable SSL for LMTP by setting the following configuration options:

```
msconfig
msconfig> set isc_client.sslusessl 1
msconfig# set isc_client.ischosts list_of_ISC_hosts
msconfig# set isc_client.server_port 8070
msconfig# write
msconfig> exit
```

   The host names must match the common name (cn) in the ISC node's certificate.

4. On an ISC node, generate a self-signed certificate:

```
keytool -genkey -keyalg RSA -alias selfsigned -keystore
/tmp/keystore.jks -storepass Java_keystore_password -validity 360 -keysize
2048
```

5. On the ISC node, export the certificate for use in the FIT.

```
keytool -exportcert -keystore /opt/sun/comms/messaging64/config/keystore.jks
-alias selfsigned -rfc -file isc.crt
```

6. Copy the resultant **isc.crt** certificate file to all FIT nodes in the *fit_install_path*/**config** directory. (The default is **/usr/share/dse/fit**).

7. On message access layer nodes, import the certificate into the Message Server certificate database (**cert9.db**).

   ```
   certutil -A -d sql:/opt/sun/comms/messaging64/config -n iscSelfCert -i
   /opt/sun/comms/messaging64/config/isc.crt -t P
   ```

8. Restart the ISC:

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

9. Reload the FIT by using the Solr administration console.

Configuring ISC for SSL Using a CA-Signed Certificate

To configure the ISC for SSL using a CA-signed certificate:

1. On ISC nodes, enable SSL by setting the following configuration options:

   ```
   msconfig
   msconfig> set isc.sslusessl 1
   msconfig# set base.ssljkspath path_to_Java_keystore
   msconfig# set base.ssljkspassword
   Password: Java_keystore_password
   Verify:
   msconfig# write
   msconfig> exit
   ```

2. On FIT nodes, enable SSL by setting the **fit.sslusessl** configuration option:

   ```
   cd /opt/sun/comms/cas/bin
   config-fit --sslusessl 1
   ```

3. On message access layer nodes (in general, the same node as ISC node), enable SSL for LMTP by setting the following configuration options:

   ```
   msconfig
   msconfig> set isc_client.sslusessl 1
   msconfig# set isc_client.ischosts list_of_ISC_hosts
   msconfig# set isc_client.server_port 8070
   msconfig# write
   msconfig> exit
   ```

   The host names must match the common name (cn) in the ISC node's certificate.

4. On an ISC node, generate and import a CA-signed certificate.

   a. Generate a keypair:

      ```
      keytool -genkeypair -keysize 2048 -alias isc -keyalg RSA -dname
      "CN=Hostname,O=Org,L=Location,ST=State,C=CountryCode" -keystore
      keystore.jks -storepass Java_keystore_password -validity 360
      ```

   b. Generate a certificate request:

      ```
      keytool -certreq -alias isc -keystore keystore.jks -storepass Java_
      keystore_password -file isc.csr
      ```

   c. Get the CA-signed certificate by using certificate the request file, **isc.csr**.

   d. Import the CA certificate (**cacert.crt**) and CA-signed certificate (**isc.crt**) into the keystore:

      ```
      keytool -importcert -trustcacerts -alias CA -file  cacert.crt -keystore
      ```

```
    keystore.jks -storepass Java_keystore_password
    keytool -importcert -alias isc -file isc.crt -keystore keystore.jks
    -storepass Java_keystore_password
```

5. On message access layer nodes, import the certificate into the Message Server certificate database (**cert9.db**).

   ```
   certutil -A -d sql:/opt/sun/comms/messaging64/config -n CA -i cacert.crt -t P
   ```

6. Restart the ISC:

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

7. Reload the FIT by using the Solr administration console.

Setting Up Certificate-Based Login

In addition to password-based authentication, Oracle servers support authentication of users through examination of their digital certificates. In certificate-based authentication, the client establishes an SSL session with the server and submits the user's certificate to the server. The server then evaluates whether the submitted certificate is genuine. If the certificate is validated, the user is considered authenticated.

To Set Up Certificate-Based Login

1. Obtain a server certificate for your server. (For details, see "Obtaining Certificates".)
2. Install the certificates of any trusted certificate authorities (CAs) that will issue certificates to the users that your server will authenticate. As long as there is at least one trusted CA in the server's database, the server requests a client certificate from each connecting client.
3. Turn on SSL. (For details, see "To Enable SSL and Selecting Ciphers".)
4. Set up a certificate mapping for your server. For example, to make the default certificate map match by the email address in the certificate subject, make the following configuration changes at the **msconfig** prompt:

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

   For more information about certificate mapping and the **dncomp** and **filtercomps** options, see the discussion about certificate-based client authentication in *Messaging Server System Administrator's Guide*.

Once you have taken these steps, when a client establishes an SSL session so that the user can log in to IMAP or HTTP, the Messaging Server requests the user's certificate from the client. If the certificate submitted by the client has been issued by a CA that the server has established as trusted, and if the certificate identity matches an entry in the user directory, the user is authenticated and access is granted (depending on access-control rules governing that user). There is no need to disallow password-based login to enable certificate-based login.

If password-based login is allowed (which is the default state), and if you have performed the tasks described in this section, both password-based and certificate-based login are supported. In that case, if the client establishes an SSL session and supplies a certificate, certificate-based login is used. If the client does not use SSL or does not supply a certificate, the server requests a password.

User/Group Directory Lookups Over SSL

It is possible to do user/group directory lookups over SSL for MTA, MMP, and IMAP/POP/HTTP services. The prerequisite is that Messaging Server must be configured in SSL mode.

Set the **base.ugldapusessl** option to 1. For example:

```
msconfig set base.ugldapusessl 1
```

Set the **base.ugldapport** to option **636** to enable this feature. For example:

```
msconfig set base.ugldapport 636
```

You can also use LDAP + STARTTLS by setting **base.ldaprequiretls** to 1. For example:

```
msconfig set base.ldaprequiretls 1
```

Note: Failure to configure Messaging Server correctly for secure LDAP communication can cause Messaging Server commands and utilities, such as **imsimta test -rewrite**, to fail.

8 Certificate-Based Authentication for Messaging Server

This chapter describes the installation, configuration, and standards for certificate-based authentication for Oracle Communications Messaging Server using SSL/TLS.

Introduction: SSL/TLS, Client Certificates and CRLs

The following discusses issues related to configuring Messaging Server to use client certificates with Certificate Revocation Lists (CRLs) and SSL/TLS.

Authentication Technology Overview

Messaging Server supports three types of end-user authentication:

1. Passwords in the clear
2. Passwords over SSL/TLS
3. Client certificates over SSL/TLS

Option 1 is for very low security sites, option 2 is current best practice, and option 3 is usually for very high security sites. Some parts of Messaging Server have additional options. CRAM-MD5/APOP passwords are slightly better on the wire, but worse on the back-end than those in the clear. Therefore, CRAM-MD5/APOP passwords are not recommended. HTTP-only WebSSO systems most commonly use passwords over SSL/TLS.

By default, only passwords in the clear are enabled for Messaging Server. When SSL/TLS is configured, both passwords in the clear and passwords over SSL/TLS are enabled.

SSL/TLS Overview

Transport Layer Security (TLS) is the Internet proposed standard most widely used to secure application protocols such as POP, IMAP, SMTP, and HTTP. TLS is a newer and more secure version of the Secure Sockets Layer (SSL) protocol designed by Netscape. SSLv2 is an insecure legacy version of SSL which is unsupported. From the release 8.1, SSLv3 is also no longer supported.

SSL provides session encryption, server certificate authentication and optional client certificate authentication. The actual security algorithms used by SSL/TLS are controlled by a negotiated **cipher suite** and by the authentication certificates.

STARTTLS is the command used in an application protocol (for example, POP, IMAP, SMTP, SMTP Submission, LDAP) to start SSL/TLS. SSL/TLS is started on the application's standard port rather than a special port reserved for SSL/TLS usage only. Clients discover the availability of SSL/TLS by default and cache this information. Thus, clients are secure-by-default without requiring explicit configuration.

Note: A new security compliance rule requires TLS 1.0 and TLS 1.1 to be disabled by default. This means TLS 1.2 or 1.3 must be used when connecting to Messaging Server security in the default configuration. The **tlsminversion** option can be set to TLS1.0 or TLS 1.1 to restore the previous default behavior.

Certificate Authentication Overview

Certificate Authentication involves several components:

- Public key (information shared with other parties)
- Associated private key (secret information)
- Certificate authority (an authority that asserts someone in possession of a particular private key has a certain identity)
- Certificate (a public key with one or more identities signed by itself and/or a certificate authority)

A Trust Anchor is the certificate for a well known certificate authority (CA) that is distributed with the software. Messaging Server does not enable any trust anchors by default. For authentication, the public key and its associated public key algorithm (typically RSA) are used to verify that the other party knows the associated private key.

A certificate's primary identity is called the **certificate subject** and is represented in *attribute=value,attribute=value* format. Typical certificates use a convention to encode a domain name or email address in the certificate subject. A certificate can also have a *subjectAltName* with an explicit DNS name or email syntax.

A **certificate revocation list** (CRL) is a list of certificates which are no longer valid signed by the relevant CA.

Certificate and Key Storage Overview

Certificates and keys must be stored somewhere. By default, Messaging Server stores certificates in a file in the product's configuration directory called **cert8.db** or **cert9.db**. Keys are stored in an encrypted file in the configuration directory, called either **key3.db** or **key4.db**. The encryption password to the key file is stored in **sslpassword.conf** so the server can reboot without administrative intervention. In general, the **certutil** tool manages certificates and the **pk12util** tool manages keys and certificate/key pairs.

SSL/TLS/Certificate Standards Overview

The following standards (or a subset thereof) are implemented in Messaging Server:

- TLS 1.0 (RFC 2246) defines the underlying transport layer security protocol independent of how it integrates with various applications. The most recent version of TLS is 1.3 (RFC 8446).

- TLS Renegotiation Indication Extension (RFC 5746) fixes a security bug present in all versions of SSL and TLS prior to TLS 1.3.
- SASL EXTERNAL (RFC 2222 section 7.4) is a protocol authentication mechanism typically used by POP, IMAP, and SMTP clients to explicitly tell a server to use external authentication, such as client certificates. As our servers support administrative proxy authentication (where an administrator's credentials are used to impersonate a regular user), this mechanism is required for that feature to work correctly with client certificate authentication. EXTERNAL occurs automatically when the following conditions are met:
  - A CA certificate is installed and is marked as trusted to validate peer certificates in the certificate DB.
  - **certmap.conf** is configured.
  - The client offers a valid certificate during a successful SSL negotiation.
  - The server can locate the user account associated with the certificate based on the **certmap.conf**. If **verifycert** is on the certificate, it has to match the one in the user's LDAP directory entry.
- IMAP STARTTLS (RFC 3501 sections 6.2.1 & 11.1) is the standard for use of TLS with IMAP; and it requires use of SASL EXTERNAL.

  Note: IMAPS on port 993 is not a standard and no rules exist for its use with client certificates beyond what can be inferred from RFC 3501.

- POP STARTTLS (RFC 2595 sections 2 & 4) is the standard for use of TLS with POP; and it requires use of SASL EXTERNAL.

  Note: POPS on port 995 is not a standard and no rules exist for its use with client certificates beyond what can be inferred from RFC 2595 and RFC 1939 (section 4 is particularly relevant and states that a POP session starts in AUTHORIZATION state and remains there until a mechanism such as SASL EXTERNAL is used to transition to TRANSACTION state).

- SMTP STARTTLS (RFC 3207) is the standard for use of TLS with SMTP (RFC 2821) and SMTP Submission (RFC 4409). It does not describe how client certificates are used for submission authentication; so we follow the model of the POP/IMAP standards that are explicit. Use of SSL on port 465 for SMTP submission is a standard protocol with no documented interoperability rules; we follow the same rules that we do on the standard submission port with **STARTTLS** when this is configured.
- HTTPS (RFC 2818) is a de-facto standard for use of SSL/TLS on the **HTTPS** port (443). The **mshttpd** daemon that runs on port 8991 by default when SSL is enabled implements SSL/TLS, but with the safer wild-carding rules present in this de-facto standard.
- PKCS#12 is an industry standard binary format used to store a certificate and a private key, optionally encrypted by a password. It typically has a **.p12** or **.pk12** file extension when stored in a file.

- PEM is an ASCII encoding used to store certificates and/or keys and/or CRLs and/or PKCS#12. It typically has a **.pem** file extension.
- PKCS#11 is an industry standard that defines how a security library talks to software or hardware that can implement certificate and key storage, and encryption algorithms. The **modutil** tool controls use of alternate PKCS#11 modules with Messaging Server. There is a Solaris-specific PKCS#11 module present in Solaris 10 with support for cryptographic hardware acceleration present in newer Sparc chips that can be used instead of the default PKCS#11 software module.
- PKIX (RFC 3280) is a standard for verifying certificates and using CRLs. Messaging Server supports a commonly used subset of PKIX.
- Online Certificate Status Protocol (OCSP, RFC 2560) is a protocol to check the status of a client certificate with a remote server, removing the need to fetch and copy CRLs. Messaging Server does not support this protocol.

SSL/TLS Tools Available in Messaging Server Installer

This section outlines the available SSL/TLS tools in Messaging Server Installer.

Utilities Used to Manage Certificates

The following utilities are located in the Messaging Server **lib** directory.

- The **certutil** tool is used to manage certificates. When creating a certificate or certificate request, include **-d** followed by a database directory with a **sql:** prefix, and **-Z SHA256 -g 2048** to enable modern security.
- The **pk12util** tool is used to manage keys or certificate and key pairs. To import or export PKCS#12 files, and PEM-encoded PKCS#12, use the **-a** option.
- The **modutil** tool is used to manage PKCS#11 modules.
- The **crlutil** tool is used to manage CRLs.
- The **msgcert** tool is no longer available because it used an inadequate key size and deprecated signature algorithm. Use the **certutil** tool with the previously mentioned options instead.
- The **configure** tool used to configure the Messaging Server has a **-ssl** option that allows configuration against an **LDAPS**(LDAP over SSL on port 636) server. However, use of this tool requires the administrator to create the Messaging Server's configuration directory manually, in advance, and to use **certutil** to import the LDAP server's certificate and mark it as a trusted peer.

Certificate and Key Storage

The command **msconfig set base.ssldblegacy** (Unified Configuration) or **configutil -o local.ssldblegacy -v 0** (legacy configuration) enables support for a newer format (**cert9.db**, **key4.db**, **pkcs11.txt**), which allows concurrent modification of the locally-stored certificates, keys, CRLs, and PKCS#11 modules without requiring that the server be shut down.

Modifying the Certificate Format

The aforementioned tools use the newer format by default.

If a site is using a legacy certificate format, the following commands upgrade to the new format. (Substitute appropriate path for your site to the configuration):

1. **cd /opt/sun/comms/messaging64/config**
2. Create file containing just the password for the key database with no extra newline. Call this **pwfile**.
3. **certutil --merge -d -Z SHA256 -g 2048 -f pwfile --source-dir . -@ pwfile**
4. Create a directory for the old format databases: **mkdir old_certdb**
5. **mv cert8.db key3.db secmod.db old_certdb**
6. Recompile the MTA configuration if you are running a compiled configuration and restart the Messaging Server.

```
imsimta cnbuild
stop-msg
start-msg
```

Note: Executing the latter two commands prevents the **certutil** and **pk12util** utilities from using the old databases, if you happen to forget to set the environment variable. Remove the old format databases once a successful migration is verified.

Checking the NSS version

The new certificate and key storage format was introduced in NSS version 3.12. To check the version of NSS, run the **imsimta version** command.

SSL/TLS Configuration

This section summarizes the configuration options related to the use of SSL/TLS with Messaging Server.

SSL-Related Settings

Table 8–1 provides a summary of the available SSL-related settings, which do not apply to the MMP unless explicitly mentioned.

Table 8–1 SSL-Related Settings

Unified Configuration	Legacy Configuration	Description
base.sslnicknames	**encryption.rsa.nssslpersonalityssl**	Sets the certificate nickname of the server certificate used by the Messaging Server by default.
base.defaultdomain	**service.defaultdomain**	Default domain for authentication if one not explicitly specified; including AUTH EXTERNAL.
http.sslnicknames **imap.sslnicknames** Not applicable **mta.sslnicknames**	**local.http.sslnicknames** **local.imap.sslnicknames** **local.smtp.sslnicknames** **local.imta.sslnicknames**	Specify a certificate nickname of a server certificate that will be used by a particular Messaging Server service.

Table 8–1 (Cont.) SSL-Related Settings

Unified Configuration	Legacy Configuration	Description
base.ugldapusessl	**local.ugldapusessl**	If set, use LDAPS (LDAP-over-SSL default port 636) when contacting an LDAP server.
base.proxyimapssl	**local.service.proxy.imapssl**	If set, use IMAPS (IMAP-over-SSL default port 993) when proxying to another IMAP back-end for shared folders. Defaults to 1 if IMAP port is 993.
http.sslusessl **imap.sslusessl** **pop.sslusessl**	**service.http.sslusessl** **service.imap.sslusessl** **service.pop.sslusessl**	Enable use of SSL/STARTTLS for these services.
http.sslport **imap.sslport** **pop.sslport**	**service.http.sslport** **service.imap.sslport** **service.pop.sslport**	Specify a non-default SSL port.
http.enablesslport **imap.enablesslport** **pop.enablesslport**	**service.http.enablesslport** **service.imap.enablesslport** **service.pop.enablesslport**	Enable use of SSL on a separate port for this service.
isc.sslusessl **fit.sslusessl** **isc_client.sslusessl**	Not applicable	Enable use of SSL/STARTTLS for these services.
http.plaintextmincipher **imap.plaintextmincipher** **pop.plaintextmincipher** **mta.plaintextmincipher**	**service.http.plaintextmincipher** **service.imap.plaintextmincipher** **service.pop.plaintextmincipher** **service.imta.plaintextmincipher**	If set to 1, these utilities require use of SSL or STARTTLS prior to use of a plaintext password. In previous versions of SMTP, this behavior had to be simulated with multiple channels, and the use of **tlsswitchchannel** and **saslswitchchannel**. The use of **tlsswitchchannel** is no longer necessary.
http.sslcachesize **imap.sslcachesize** **pop.sslcachesize** Not applicable	**service.http.sslcachesize** **service.imap.sslcachesize** **service.pop.sslcachesize** **service.imta.sslcachesize**	Specifies the number of SSL sessions to cache (a cached session reconnects clients faster).
base.ldaptrace	**local.ldaptrace**	Enables additional LDAP diagnostics.
base.ssladjustciphersuites	**local.ssladjustciphersuites**	Determines which SSL cipher suites are enabled for use.
base.sslcompress	**local.sslcompress**	Enables SSL/TLS compression (per RFC 3749). Default = 1 and is used by MMP.
base.ssldbpath	**local.ssldbpath**	Determines where SSL certificate and key databases are stored (defaults to configuration directory). Using another directory is discouraged.
base.ssldblegacy	**local.ssldblegacy**	When set to 0, enables use of the new certificate and key database format. Used by MMP.
base.ssldbprefix	**local.ssldbprefix**	A prefix used in the certificate and key database file names. Changing this is discouraged.
base.ssljkspath	Not applicable	Specifies the path of the keystore file.

Table 8–1 (Cont.) SSL-Related Settings

Unified Configuration	Legacy Configuration	Description
base.ssljkspassword	Not applicable	Specifies the password for the keystore file.
base.sslpkix	**local.sslpkix**	Setting this makes your system unsupported. It enables untested experimental code with unknown scalability, timeout, and failure handling characteristics to perform full RFC 3280 PKIX validation of certificates. Used by MMP.
base.sslrequiresafenegotiate	**local.sslrequiresafenegotiate**	Requires use of the SSL/TLS renegotiate extension (per RFC 5746). Default = 0 and is used by MMP.

Dispatcher SSL-Related Settings

When a dispatcher service configuration block includes **TLS_PORT**, the server will negotiate SSL/TLS immediately before starting the specified application. This is typically used to enable the standard submissions port 465 in the **SERVICE=SMTP_SUBMIT** section and that is presently a commented-out option in the default configuration (older versions used to incorrectly include that as an option in the **SERVICE=SMTP** section although it is rarely used for SMTP relay).

Messaging Transfer Agent (MTA) SSL-Related Channel Options

Table 8–2 shows SSL-related channel options.

Table 8–2 SSL-Related Channel Options

Option	Description
maytlsserver	Allow use of TLS connecting to the server for that channel
musttlsserver	Require use of TLS connecting to the server for that channel
tlsswitchchannel	Change channels if TLS is successfully negotiated
maytlsclient	Use TLS if available for outbound mail sent through that channel
musttlsclient	Require TLS for outbound mail sent through that channel
maysaslserver	Allow use of SMTP AUTH (including AUTH EXTERNAL)
mustsaslserver	Require use of SMTP AUTH (including AUTH EXTERNAL)
saslswitchchannel	Change channels if SASL authentication is successful

SMTP Channel Options

Table 8–3 shows the SMTP channel options.

Table 8–3 SMTP Channel Options

SMTP Channel Option	Description
IGNORE_BAD_CERT	The **maytls** channel keywords and SMTPS ignore errors with bad client and server certificates. Use the **musttls** channel keywords to control whether the SMTP client or STARTTLS command processor on the server will ignore bad certificates as follows: ■ Set bit 0 (value 1) to ignore bad client certificates. ■ Set bit 1 (value 2) to ignore bad server certificates. ■ Default setting is 3 (ignore bad certificates in SMTP).
SSL_CLIENT	Set this option to 1 to negotiate SSL/TLS on outbound client connections from this channel.
EXTERNAL_IDENTITY	Use to enable support for SASL AUTH EXTERNAL for outbound client connections. The value may be the empty string which is the identity the remote server implicitly associates with this client. When an SMTP Server's TLS implementation asks the Messaging Server SMTP client for a client certificate, the SMTP TLS client only supports providing the default server certificate for that Messaging Server installation.
AUTH_USERNAME	For outbound client SMTP connections, use this username with SASL PLAIN authentication.
AUTH_PASSWORD	For outbound client SMTP connections, use this password with SASL PLAIN authentication.
PORT_ACCESS SSL-Related Fields	The right hand side of PORT_ACCESS supports some additional fields on a successful match: {*$Yruleset* | *realm* | *tls-cert-nickname* Additional fields between *realm* and *tls-cert-nickname* are enabled by LOG_CONNECTION bit 4 and $D) While *ruleset* is not currently used, the other two fields are typically used to change SSL/TLS and authentication behavior based on the server IP address to which the client is connected. The *realm* is the default domain appended to an unqualified authentication ID, and the *tls-cert-nickname* is the nickname of an alternate TLS server certificate to use.
BURL_ACCESS Mapping Table SSL-Related Input Flags	■ **$:A** Test if SASL authentication complete ■ **$:T** Test if TLS is in use

MMP SSL-Related Settings

Table 8–4 shows MMP options that appear in **PopProxyAService.cfg**, **ImapProxyAService.cfg**, and/or **SmtpProxyAService.cfg**. All of these configuration files that specify **SSLEnable** *must* have the same settings. If different settings are used in different files, the MMP SSL subsystem will initialize with the settings from only one of the files. Use of non-default values is discouraged.

Table 8–4 MMP SSL-Related Settings

MMP SSL-Related Option	Description
SSLCacheDir	The directory to search for certificate & key databases.
SSLCertPrefix	Filename prefix used when locating the **cert8.db** or **cert9.db**.
SSLKeyPasswordFile	Filename to use instead of **sslpassword.conf** for SSL key passwords.

Table 8–4 (Cont.) MMP SSL-Related Settings

MMP SSL-Related Option	Description
SSLKeyPrefix	Filename prefix used when locating the **key3.db** or **key4.db**.
SSLSecmodFile	Specifies alternate file name/path for **secmod.db**.

Table 8–5 shows the MMP options that can be included in **PopProxyAService.cfg**, **ImapProxyAService.cfg** and **SmtpProxyAService.cfg**. The options can also be included in **vdmap.cfg** and may have different settings for each service or virtual domain.

Table 8–5 Optional MMP SSL-Related Options

Optional MMP SSL-Related Options	Description
DebugKeys	A list of keywords used to enable additional diagnostics. The **TLS** keyword enables some additional TLS debugging features.
DefaultDomain	Default domain for authentication identities without an explicit domain, including AUTH EXTERNAL.
RestrictPlainPasswords	Requires the use of SSL/TLS prior to allowing plaintext password authentication.
SSLAdjustCipherSuites	Determines the availability of the SSL/TLS cipher suites.
SSLCertNicknames	Assigns certificate nickname(s) to use for server.
StoreAdmin	Determines which user identity to use when proxying to back-end server. Required to support client certificates with MMP.
StoreAdminPass	Assigns password for user identity to use when proxying to back-end server. Required to support client certificates with MMP.

Table 8–6 shows the MMP SSL-related options that may have different settings for each service.

Table 8–6 MMP SSL-Related Options that May Have Different Settings for Each Service

MMP SSL-Related Option with Different Settings Per Service	Description
CertMapFile	Names the certificate mapping file.
LdapUrl	Uses **LDAPS:** instead of **LDAP:** with SSL when talking to LDAP.
SSLBacksidePort	Use this port when communicating to the back-end over SSL only if the connection to the MMP is also over SSL.
SSLEnable	Enables STARTTLS and SSL for this service.
SSLPorts	Assigns one or more ports to negotiate SSL immediately. The assigned ports must also be listed in the appropriate ServiceList element in **AService.cfg**.
UserGroupDN	All user entries appear below this LDAP DN (Distinguished Name) in the LDAP DIT (Directory Information Tree). This is used both by the client certificate authentication mapping subsystem (regardless of schema level) and by schema 2.

Table 8–7 shows the MMP option that is only available in **vdmap.cfg**.

Table 8–7 MMP SSL-Related Option Only Available in vdmap.cfg

MMP SSL-Related Option Only Available in vdmap.cfg	Description
CertMap	Specifies which **certmap.conf** settings are used by default for this virtual domain.

certmap.conf Settings

For client certificate authentication to work, a client certificate must be translated to a Messaging Server user with an entry in LDAP. The **certmap.conf** configuration file is required to perform this function. The default installation creates a **certmap.conf.sample** file in the configuration directory which can be copied to **certmap.conf**. **certmap.conf** has named sections - a **default** section is mandatory - so different rules can be applied to different client certificate issuer DNs. Table 8–8 shows the available options.

Table 8–8 certmap.conf Options

certmap.conf Option	Description
DNComps	commented out - take the user's DN from the certificate as is empty - search the entire LDAP tree (**DN == suffix**) attr names - a comma separated list of attributes to form DN
FilterComps	commented out - set the filter to **objectclass=** empty - set the filter to **objectclass=** attr names - a comma separated list of attributes to form the filter
verifycert	The user's LDAP entry must have a **userCertificate;binary** field that matches the certificate used by the client.
CmapLdapAttr	If not empty, search the entire tree for an entry with the **CmapLdapAttr** attribute that matches the client certificate subject.

SSL/TLS Tasks

This section includes procedures to create, install, and test the different types of certificates and keys used by SSL/TLS.

How to Create and Install a Self-signed CA Certificate and Key

Using the following procedure, you can create a self-signed CA certificate used to sign client and server certificates. (In a production environment you would generate a certificate request (-R) and have that signed by a CA certificate.)

```
cd /opt/sun/comms/messaging64/config
certutil -d -g 2048 -Z SHA256 -N
```

This prompts for a password. Save that password to a file called **pwfile** with no trailing newline. Also save the password to a file called **sslpassword.conf** which should contain the single line:

```
Internal (Software) Token:password
```

Where the password selected is after the ':'. You may include a newline in this file.

Ensure that the certificate, key, and **pkcs11.txt** files are owned by the Messaging Server (typically this is the **mailsvr** user).

Then create a self-signed certificate by performing the following command:

```
certutil -S -d -g 2048 -Z SHA256 -n CA-Cert -s "cn=CA Cert for Messaging" -x -t CT
-f pwfile
```

How to Create and Install a CA-signed Server Certificate and Key

To create and install a CA-signed server certificate and key, perform the following:

```
certutil -S -d -g 2048 -Z SHA256 -n Server-Cert -s "cn=mail.example.com" -c
CA-Cert -f pwfile -t P
```

How to Create a CA-signed Client Certificate and Key

To create a CA-signed client certificate and key, perform the following:

```
certutil -S -d -g 2048 -Z SHA256 -m 1 -n client-cert -s "e=user@example.com" -c
CA-Cert -f pwfile -t u
```

How to Test a CA-signed Client Certificate and Key

To test a CA-signed client certificate and key, perform the following:

msconfig set imap.sslusessl 1 (Unified Configuration) or **configutil - o service.imap.sslusessl -v 1** (legacy configuration).

```
start-msg
/opt/sun/comms/messaging64/lib/sslconnect -r -c client-cert mail.example.com 143
```

Next, type the following command: **A AUTHENTICATE EXTERNAL =**

and the following appears:

A OK User logged in

How to Create and Install a CRL for a Client Certificate

To create and install a CRL for a client certificate, perform the following:

```
crlutil -d -G -n CA-Cert << EOF
update=20100510200000Z
addcert 1 20100510200000Z
EOF
```

The **1** in the **addcert** is the serial number of the client certificate, which was specified by the **-m** option used when creating the client certificate.

How to Test a CRL for a Client Certificate

To test a CRL for a client certificate, perform the following:

```
/opt/sun/comms/messaging64/lib/sslconnect -r -c client-cert mail.example.com 143
```

Next, type the following command: **A AUTHENTICATE EXTERNAL =**

and the following appears:

```
A NO Mechanism not Available
```

The following message should appear in the IMAP log:

```
[10/May/2010:20:29:20 -0700] nifty-silver imapd[12720]: General Notice: Bad
certificate from [127.0.0.1:64215]: errno -8180 (Peer's Certificate has been
revoked.)
```

How to Look Up Numeric SSL/TLS Error Codes

To look up numeric error codes related to SSL/TLS, see:

https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/SSL_functions/sslerr.html

String error text and numeric error codes are outputted to debug log files. Often, this site contains additional information that is not in the string error text that we output.

Sample Protocol Sessions with Client Certificate Authentication

This section inauthentication uses standard protocol.cludes example protocol transcripts where client certificate

IMAP (STARTTLS) default port 143

The following is how standard IMAP client certificate authentication executes:

```
S: * OK CommSuite7:CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS
CHILDREN BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT
THREAD=ORDEREDSUBJECT THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN
SASL-IR SEARCHRES XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1
X-UNAUTHENTICATE X-SUN-IMAP X-ANNOTATEMORE XUM1 ID STARTTLS IDLE XREFRESH
AUTH=PLAIN nifty-silver.west.example.com IMAP4 service (Oracle Communications
Messaging Server 7u5-0.01 32bit (built Apr 8 2010))

C: a STARTTLS

S: a OK Completed

...TLS-negotiation-with-client-cert...

C: b CAPABILITY

S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS CHILDREN
BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT THREAD=ORDEREDSUBJECT
THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN SASL-IR SEARCHRES
XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1 X-UNAUTHENTICATE
X-SUN-IMAP X-ANNOTATEMORE XUM1 ID IDLE XREFRESH AUTH=EXTERNAL AUTH=PLAIN

S: b OK Completed

C: c AUTHENTICATE EXTERNAL =

S: c OK User logged in
```

For a standard IMAP client certificate authentication to execute successfully, the following requirements must be met:

1. There has to be a valid server certificate installed.
2. The client has to connect to the fully-qualified domain name of the server using the fully-qualified hostname present in the server certificate.

3. There has to be a CA certificate trusted to sign client certificates installed.
4. **service.imap.sslusessl** must be enabled.
5. There must be a valid **certmap.conf** with correct permissions.
6. The client must supply a valid client certificate during the SSL exchange, and the **certmap** code must map the certificate to a valid mail user.

AUTH=EXTERNAL appears in the capability list when the above conditions are met.

This can also be used for administrative proxy authentication. If the client certificate is a store administrator who wants to authenticate as user **cnewman**, the command would be:

```
C: c AUTHENTICATE EXTERNAL Y25ld21hbg==

S: c OK User logged in

====
```

An attempt to proxy authenticate inappropriately:

```
====

c AUTHENTICATE EXTERNAL YWRtaW4=

c NO Not authorized to login as specified user
```

Submission (STARTTLS) Default port 587

```
S: 220 nifty-silver.west.example.com - Server ESMTP (Oracle Communications
Messaging Server 7u5-0.01 32bit (built Apr 20 2010))

C: EHLO nifty-silver.west.example.com

S: 250-nifty-silver.west.example.com

S: 250-8BITMIME

S: 250-PIPELINING

S: 250-CHUNKING

S: 250-DSN

S: 250-ENHANCEDSTATUSCODES

S: 250-EXPN

S: 250-HELP

S: 250-XADR

S: 250-XSTA

S: 250-XCIR

S: 250-XGEN

S: 250-XLOOP 8DA8D338B89F8CEC5FED34564D95F616
```

```
S: 250-STARTTLS

S: 250-AUTH PLAIN LOGIN

S: 250-AUTH=LOGIN PLAIN

S: 250-NO-SOLICITING

S: 250 SIZE 0

C: STARTTLS

S: 220 2.5.0 Go ahead with TLS negotiation.

...TLS-negotiation-with-client-cert...

C: EHLO nifty-silver.west.example.com

S: 250-nifty-silver.west.example.com

S: 250-8BITMIME

S: 250-PIPELINING

S: 250-CHUNKING

S: 250-DSN

S: 250-ENHANCEDSTATUSCODES

S: 250-EXPN

S: 250-HELP

S: 250-XADR

S: 250-XSTA

S: 250-XCIR

S: 250-XGEN

S: 250-XLOOP 8DA8D338B89F8CEC5FED34564D95F616

S: 250-AUTH EXTERNAL PLAIN LOGIN

S: 250-AUTH=LOGIN PLAIN

S: 250-NO-SOLICITING

S: 250 SIZE 0

C: AUTH EXTERNAL =

S: 235 2.7.0 EXTERNAL authentication successful.
```

For the authentication to execute successfully the following requirements must be met:

1. There must be a valid server certificate installed.

2. The client must connect to the fully-qualified domain name of the server using the fully-qualified hostname present in the server certificate.
3. There must be a CA certificate trusted to sign client certs installed.
4. The relevant channel must include the **maytlsserver/musttlsserver** channel option. For submission on port 587 with a factory configuration, the relevant channel is **tcp_submit** and has these settings by default.
5. The relevant channel must include the **maysaslserver/mustsaslserver** channel option.
6. The relevant channel typically includes **saslswitchchannel tcp_auth**.
7. There must be a valid **certmap.conf** file with correct permissions.
8. The client must supply a valid client certificate during the SSL exchange, and the **certmap** code has to successfully map that to a valid mail user.

The standards are not clear whether the **AUTH EXTERNAL** is required or not. Currently, administrative proxy authentication is allowed through standard protocol.

POP (STLS) default port 110

```
S: +OK nifty-silver.west.example.com POP3 service (Oracle Communications Messaging
Server 7u5-0.01 32bit (built Apr 8 2010))

C: STLS

S: +OK

C: CAPA

S: +OK list follows

S: TOP

S: PIPELINING

S: UIDL

S: RESP-CODES

S: AUTH-RESP-CODE

S: SASL EXTERNAL PLAIN

S: USER

S: IMPLEMENTATION POPD-7.5p0.01 Apr 28 2010

S: .

C: AUTH EXTERNAL =

S: +OK Maildrop ready
```

The standard POP (STLS) executes in the same way as the standard IMAP (STARTTLS) with the same requirements.

IMAPS typical port 993

```
...TLS-negotiation-with-client-cert...

S: * PREAUTH CommSuite7:CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE
UIDPLUS CHILDREN BINARY UNSELECT SORT CATENATE URLAUTH LANGUAGE ESEARCH ESORT
THREAD=ORDEREDSUBJECT THREAD=REFERENCES ENABLE CONTEXT=SEARCH CONTEXT=SORT WITHIN
SASL-IR SEARCHRES XSENDER X-NETSCAPE XSERVERINFO X-SUN-SORT ANNOTATE-EXPERIMENT-1
X-UNAUTHENTICATE X-SUN-IMAP X-ANNOTATEMORE XUM1 ID IDLE XREFRESH
nifty-silver.west.example.com IMAP4 service (Oracle Communications Messaging
Server
7u5-0.01 32bit (built Apr 8 2010))

====
```

The * **PREAUTH** means the client certificate supplied during SSL negotiation was valid, was successfully **certmapped** to a specific user, and the specific user has been logged in already. If the server sends * **OK** then this has failed and the client must proceed with standard password authentication.

Submissions typical port 465

```
...TLS-negotiation-with-client-cert...

S: 220 nifty-silver.west.example.com - Server ESMTP (Oracle Communications
Messaging Server 7u5-0.01 32bit (built Apr 20 2010))

C: EHLO nifty-silver.west.example.com

S: 250-nifty-silver.west.example.com

S: 250-8BITMIME

S: 250-PIPELINING

S: 250-CHUNKING

S: 250-DSN

S: 250-ENHANCEDSTATUSCODES

S: 250-EXPN

S: 250-HELP

S: 250-XADR

S: 250-XSTA

S: 250-XCIR

S: 250-XGEN

S: 250-XLOOP 28B98A3E76A6F4C19EDF069FBDC26E97

S: 250-AUTH EXTERNAL PLAIN LOGIN

S: 250-AUTH=LOGIN PLAIN

S: 250-NO-SOLICITING
```

```
S: 250 SIZE 0

C: AUTH EXTERNAL =

S: 235 2.7.0 EXTERNAL authentication successful.
```

As with standard submissions STARTTLS: clients do not know if the server accepted the client certificate unless the **EXTERNAL** mechanism is available. Some third-party clients fail to look for and use the EXTERNAL mechanism.

POPS typical port 995

```
...TLS-negotiation-with-client-cert...

S: +OK nifty-silver.west.example.com POP3 service (Oracle Communications Messaging
Server 7u5-0.01 32bit (built Apr 8 2010))

C: CAPA

S: +OK list follows

S: TOP

S: PIPELINING

S: UIDL

S: RESP-CODES

S: AUTH-RESP-CODE

S: SASL EXTERNAL PLAIN

S: USER

S: IMPLEMENTATION POPD-7.5p0.01 Apr 28 2010

S: .

C: AUTH EXTERNAL =

S: +OK Maildrop ready
```

This execution differs from standard IMAPS because POP3 cannot determine if client certificate authentication succeeded. (There is nothing equivalent to IMAP's *** PREAUTH** greeting.) The Messaging Server implementation uses the only standard protocol mechanism (AUTH EXTERNAL), in compliance with RFC 1939 section 4 (RFC 1939 section 4 declares a POP3 session starts in AUTHORIZATION state). Some third-party clients fail to use the EXTERNAL mechanism and assume the client certificate worked if the connection remains open.

SSL/TLS Best Practices

The following are best practices when using SSL/TLS with Messaging Server:

- Monitor Oracle Critical Patch Updates and update Messaging Server promptly if it appears.

- Use a 2048-bit RSA certificate with a SHA256 signature.
- Enable use of SSL between end-user clients and the server whenever possible.
- Enable use of SSL between any geographically-disparate back-end servers. For example, if you have an off-site failover LDAP master configured, enable SSL when talking to LDAP.
- Set **RestrictPlainPassword** and **plaintextmincipher** whenever possible for your deployed clients. If you can identify just the clients lacking this support, put them on a separate MMP virtual domain.
- Enable the separate SSL-only ports (993, 995, 465) which can be used in addition to SSL/TLS on any of the regular ports (143,110, 587). With the **RestrictPlainPassword** and **plaintextmincipher** options, there is no significant security difference between the regular and SSL-only ports.
- If possible, disable weaker cipher suites with **ssladjustciphersuites** (particularly the RC4 cipher suites).

Client Certificate SSL/TLS Best Practices

Follow these client certificate SSL/TLS best practices:

- Set **IGNORE_BAD_CERT** to 0, at least for submission service.
- If using CRLs, use the new format certificate database.
- If using CRLs, make sure the CRLs stored in the certificate database are updated periodically, perhaps through **cron** or similar mechanism.

Messaging Server and SSL/TLS: Known Limitations

The following is not an exhaustive list of product limitations. There are additional limitations in ancillary utilities outside the core product feature set.

Administrative Proxy with a Certificate

Except for the **IMAPS** service, all of the Messaging Server servers (IMAP, POP3, SMTP Submission) support administrative proxy using a client certificate. If the administrator provides a certificate that maps to a known store administrator identity using **certmap**, that administrator can authenticate as another user through the AUTH EXTERNAL mechanism. However, the feature is not presently available on the MMP.

Proxy IMAP Authentication Limitations

When Messaging Server's **mshttpd** or **IMAPD** daemons contact an IMAP back-end server, they only support administrative proxy using plain text passwords, which are optional over the SSL port. Support for administrative proxy through a client certificate is not currently supported. STARTTLS is not currently supported either.

The MTA IMAP URL resolver used by the MTA's BURL feature does not support SSL.

Proxy MMP (IMAP/POP/SMTP-Submission) Authentication Limitations

The MMP's server components support client certificate authentication but do not support administrative proxy authentication using a mechanism other than plaintext passwords. The MMP's client components only support password replay or administrative proxy through plaintext passwords (optionally over SSL if the client also used SSL). They do not support client certificate authentication to the back-end and also do not support STARTTLS.

Internal Protocols Lacking Support for SSL and/or Authentication

The LMTP server only supports authentication using the PORT_ACCESS mapping to filter based on IP address. Basic SSL/TLS (without client certificates) should work using standard MTA channel keywords.

Disabling Passwords-Over-SSL

You cannot disable passwords-over SSL.

Hosting Multiple Domains with SSL

The MMP's IMAP, POP, and Submission proxies can be used to host multiple SSL domains with different certificates as long as each domain has a separate IP address and the appropriate **vdmap.cfg** settings. The MTA's **PORT_ACCESS** mapping supports standard submissions with **STARTTLS**, although standard submissions through the MTA does not have this feature. The back-end IMAP and POP servers do not have this support.

CRL Updates and OCSP

The default certificate validation algorithm checks a client certificate against CRLs stored in the local NSS certificate database. There is no automated procedure to update those CRLs and fetch CRLs. OCSP is not supported.

To use up-to-date CRLs for SSL, you must use a new certificate format (**cert9.db**, **key4.db**, **pkcs11.txt**) or a third-party PKCS#11 module that supports concurrent read/write access, as well write your own **cron** or equivalent script to fetch and update this information.

Time Delay for Updates to CRLs or New Certificates

When a CRL is updated in the new certificate format (**cert9.db**, **key4.db**, **pkcs11.txt**), it can take up to 10 minutes for running processes to notice the database has changed and update their internal cache of certificates and CRLs. If a CRL change must take effect immediately, the relevant servers must be restarted.

References

Technical Documentation for New Database Format in NSS:

`https://wiki.mozilla.org/NSS_Shared_DB`

NSS Shared DB Howto (Primarily for Firefox/Thunderbird):

`https://wiki.mozilla.org/NSS_Shared_DB_Howto`

Documentation for **certutil**:

`https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/tools/NSS_Tools_certutil`

Documentation for **crlutil**:

`https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/tools/NSS_Tools_crlutil`

Error Codes Returned by Mozilla APIs:

`https://developer.mozilla.org/en-US/docs/Mozilla/Errors`

9 Configuring Messaging Server and Solaris Cryptographic Framework

This chapter describes how to configure Sun Java System Messaging Server 6.3 for SSL and the Solaris Cryptographic Framework (SCF). This is a three-step process where you configure Messaging Server for SSL (optional), configure the Solaris Cryptographic Framework, then configure Messaging Server to use SCF.

About the Solaris Cryptographic Framework

The Solaris Cryptographic Framework (SCF) provides a common store of algorithms and PKCS#11 libraries to handle cryptographic requirements. The PKCS#11 libraries are implemented according to the cryptography standard created by RSA Security Inc., PKCS#11 Cryptographic Token Interface (Cryptoki). See Chapter 8, Introduction to the Solaris Cryptographic Framework, in *Oracle Solaris Security for Developers Guide* for more information. The Solaris Cryptographic Framework is available in the Solaris 10 Operating System and Solaris Express releases.

A PKCS#11 module (also called a cryptographic module or a cryptographic service provider) manages cryptographic services such as encryption and decryption through the PKCS#11 interface. PKCS#11 modules can be thought of as drivers for cryptographic devices that can be implemented in either software or hardware. A PKCS#11 module always has one or more slots, which can be implemented as physical hardware slots in some form of physical reader (for example, for smart cards) or as conceptual software slots. Each slot for a PKCS#11 module can in turn contain a token, which is the hardware or software device that actually provides cryptographic services and optionally stores certificates and keys. A hardware token is a PKCS#11 token implemented in physical devices, such as hardware accelerators and smart cards. A software token is a PKCS#11 token implemented entirely in software.

Messaging Server is configured to use the NSS built-in soft token for its cryptographic needs. Any PKCS#11 module that supports PKCS#11 can be used with NSS libraries, so the Solaris Cryptographic Framework can be used as the cryptographic service provider for Messaging Server.

Configuring Messaging Server for SSL

This section describes the procedures you use to create the secmod, key3, and cert8 databases, obtain a certificate, and install the certificate for the NSS built-in soft token. Completing such procedures enables SSL on your Messaging Server deployment.

If you already have a Messaging Server deployment configured for SSL, skip this section and proceed to "Configuring Individual Messaging Processes for SSL".

About the pk12util Command

The **pk12util** command is the primary mechanism for managing certificates. It allows you to generate a certificate request, add a certificate to the certificate database, list certificates in the certificate database, and so on. The **pk12util** command, like all Messaging Server commands, runs as **mailsrv**. Thus, even if you execute **pk12util** as root, it executes as the Messaging Server user. The **pk12util** utility is located in the *MessagingServer_home*/**bin** directory, where *MessagingServer_home* is the location of the Messaging Server installation, by default **/opt/sun/comms/messaging64**.

For detailed information about the **pk12util** command, type:

```
MessagingServer_home/bin/pk12util --help
```

Creating the Certificate Database and Add Certificate/Key Pairs

The following procedure generates **secmod**, **key3**, and **cert8** databases, and also creates the **sslpassword.conf** file. By default, certificates are generated in the *MessagingServer_home*/**config** directory.

You can also specify the location and prefix of the certificate databases by using the following **local.ssldbprefix** configuration parameter.

The Messaging Server user (for example, **mailsrv**) should be able to read and write to local.ssldbpath.

To create the certificate database and add certificate/key pairs:

1. Change directories to the Messaging Server **bin** directory.

   ```
   cd MessagingServer_home/bin
   ```

2. Run the following command:

   ```
   pk12util generate-certDB
   ```

   This utility prompts you for a password to protect the keys and certificates in the certificate database.

3. Choose the Certificate Database password. (The password is not echoed on the screen.)
4. Confirm the Certificate Database password. (The password is not echoed on the screen.)
5. Confirm that the required databases and the **sslpassword.conf** file were created.

   ```
   # ls -lrt ../config/*.db ../config/sslpassword.conf
   -rw-------   1 mailsrv  mail       32768 Nov 16 04:40 ../config/secmod.db
   -rw-------   1 mailsrv  mail       65536 Nov 16 04:40 ../config/cert8.db
   -rw-------   1 mailsrv  mail       32768 Nov 16 04:40 ../config/key3.db
   -rw-r-----   1 mailsrv  mail          36 Nov 16 04:40
   ../config/sslpassword.conf
   # cat ../config/sslpassword.conf
   Internal (Software) Token:12345678
   ```

Obtaining a Certificate

By default, the **pk12util** command creates a self-signed-certificate with the nickname Server-Cert. You can either remove this or use the self-signed-certificate.

The following example shows how to remove this default self-signed-certificate:

```
pk12util remove-cert -W MessagingServer_home/config/sslpassword Server-Cert
```

1. Request a CA-signed server certificate.

   To the **pk12util request-cert** command, specify the cert request to be in ASCII format (the default is binary). The resulting certificate request is a PKCS#10 certificate request in Privacy Enhanced Mail (PEM) format. PEM is format specified by RFCs 1421 through 1424 (RFC 1421) and used to represent a base64-encoded certificate request in US-ASCII characters.

   For example:

   ```
   # echo "12345678" > MessagingServer_home/config/sslpassword
   ```

   > **Note:** This is the same password you used to generate the certificate database.

   ```
   pk12util request-cert  -W MessagingServer_home/config/sslpassword
   --name "foobar.siroe.com" --org "Development" --org-unit "Comms" --city
   Santaclara --state California --country us -F ascii -o /tmp/MyCertRequest
   ```

   The content of the Certificate Signing Request (CSR) looks similar to this:

   ```
   # cat /tmp/MyCertRequest
   -----BEGIN NEW CERTIFICATE REQUEST-----
   MIIBvzCCASgCAQAwfzELMAkGA1UEBhMCdXMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
   EzARBgNVBAcTClNhbnRhY2xhcmExDjAMBgNVBAsTBWNvbW1zMRQwEgYDVQQKEwtE
   ZXZlbG9wbWVudDEgMB4GA1UEAxMXYmlvdGl0ZS5yZWQuaXBsYW5ldC5jb20wgZ8w
   DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALFCDfmu1uYFy3DtHAo/kJUFqXF6utq2
   ga+Tow2PNEyuX+70SqyZ0vFwiL8di9b1mLMHLp3WBDPmXjVfNkANHk6Q38RlfyzT
   7iYpvhi6+4OVthzC65FaqwnEEiodZ7z7yx8vRhj4lVxeoNJpJexGr1MHuYr8tobe
   ljgEmrLP17aNAgMBAAGgADANBgkqhkiG9w0BAQQFAAOBgQANzqjEwcnnmdXjo/KH
   buolHi1hNEYoDsXWIlTi78xkH+7gtfCPkymFzTy5mS58PzAqyWm81MZKXj39C+Eq
   DOCJmZYRt3lG6wo0M8nMoQLHsVSHfGZxOyppupzYsmVfszhczr0EEHJP66itDPW2
   /jHGRbhXfeJNhKsisscd/YYkEQ==
   ```

2. Transmit the CSR to your Certificate Authority, according to its procedures.

   The process for obtaining your Certificate Authority certificate differs depending on the certificate authority you use. Some commercial CAs provide a web site that enables you to automatically download the certificate. Other CAs email the certificate to you upon request. After you have sent your request, you must wait for the CA to respond with your certificate.

3. Save the certificate you receive back from the Certificate Authority.

   You should back up your certificates in a safe location. If you ever lose the certificates, you can reinstall them by using your backup file. You can save the certificates as text files. The PKCS#11 certificate in PEM format looks similar to the following:

   ```
   -----BEGIN CERTIFICATE-----
   MIIDmTCCAwKgAwIBAgIBZjANBgkqhkiG9w0BAQQFADCBhjELMAkGA1UEBhMCVVMx
   EzARBgNVBAgTCkNhbGlmb3JuaWExDzANBgNVBAoTBlNTRS1TVzEPMA0GA1UECxMG
   UG9ydGFsMRgwFgYDVQQDEw9WZWVyYSBOYXRhcmFqYW4xJjAkBgkqhkiG9w0BCQEW
   F3ZlZXJhLm5hdGFyYWphbkBzdW4uY29tMB4XDTA2MTExNjA3MDIyN1oXDTA3MTEx
   NjA3MDIyN1owfzELMAkGA1UEBhMCdXMxEzARBgNVBAgTCkNhbGlmb3JuaWExEzAR
   BgNVBAcTClNhbnRhY2xhcmExFDASBgNVBAoTC0RldmVsb3BtZW50MQ4wDAYDVQQL
   EwVjb21tczEgMB4GA1UEAxMXYmlvdGl0ZS5yZWQuaXBsYW5ldC5jb20wgZ8wDQYJ
   KoZIhvcNAQEBBQADgY0AMIGJAoGBALFCDfmu1uYFy3DtHAo/kJUFqXF6utq2ga+T
   ```

```
ow2PNEyuX+70SqyZ0vFwiL8di9b1mLMHLp3WBDPmXjVfNkANHk6Q38RlfyzT7iYp
vhi6+4OVthzC65FaqwnEEiodZ7z7yx8vRhj4lVxeoNJpJexGr1MHuYr8tobeljgE
mrLP17aNAgMBAAGjggEbMIIBFzAJBgNVHRMEAjAAMDUGCWCGSAGG+EIBDQQoFiZT
dW5PTkUtUFRTIFBvcnRhbDogT3BlblNTTCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQU
wkhhgKwbt1P8SXrRHpesVuhel0gwgbMGA1UdIwSBqzCBqIAUIALOde3lgiZiUwXo
PTiN/YaJKoihgYykgYkwgYYxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9y
bmlhMQ8wDQYDVQQKEwZTU0UtU1cxDzANBgNVBAsTBlBvcnRhbDEYMBYGA1UEAxMP
VmVlcmEgTmF0YXJhamFuMSYwJAYJKoZIhvcNAQkBFhd2ZWVyYS5uYXRhcmFqYW5A
c3VuLmNvbYIBADANBgkqhkiG9w0BAQQFAAOBgQAdsOxEygD/Rbj4NWHhTrAZcn2B
mWv40MFS1oAgJSMc5BPTBGHcSnnLEh0ZApFLfWknVro4ubW3mb5ByaHoR3sOsxAO
4705avDUgX2g+4V80ef2CVOo5AoZRNMgVMt4Ju3D1PDZsDWQstbfV3PTeMyAzy/7
NZh1+adCuh8J+Rhl4Q==
-----END CERTIFICATE-----
```

4. Follow the same steps described previously to obtain additional certificates.

Adding Certificates to the NSS Software Token

To add certificates to the NSS software token:

1. Save the signed certificate into a temporary location, for example:

   ```
   /tmp/caissuedFirstCert
   ```

   For this example, a second signed certificate was also obtained using the previous procedure and saved as the following:

   ```
   /tmp/caissuedSecondCert
   ```

2. Install the CA-signed server certificates.

   ```
   pk12util add-cert -W MessagingServer_home/config/sslpassword Server-Cert
   /tmp/caissuedFirstCert
   pk12util add-cert -W MessagingServer_home/config/sslpassword
   SolCrypto-Framework /tmp/caissuedSecondCert
   ```

3. Repeat Steps 1 and 2 for other certificates you have obtained.
4. Verify that the certificates have been successfully installed.

   ```
   pk12util list-certs -W MessagingServer_home/config/sslpassword
   ...
   2 certificates found
   ```

Listing the Default NSS Certificates

For newer certificate formats (**cert9.db** and **pkcs11.txt**), to list the default NSS certificates, you need to first add the **libnssckbi.so** default certificate library to the configuration.

To list the default built-in certificates:

1. Change directories to the Messaging Server's configuration directory.

   ```
   cd /opt/sun/comms/messaging64/config
   ```

2. Run the following command to add the **libnssckbi.so** default certificate library to the configuration directory:

   ```
   modutil -dbdir sql:. -add "NSS certificates" -libfile
   /opt/sun/comms/messaging64/lib/libnssckbi.so
   ```

3. Run the following command to list the default NSS certificates:

```
certutil -L -d sql:. -h "NSS certificates"
```

Configuring Individual Messaging Processes for SSL

This section describes the procedures you use to configure the various Messaging Server processes for SSL.

Configuring MMP for SSL

To configure the MMP for SSL:

1. Set the SSL certificate nickname.

   Edit the **ImapProxyAService.cfg** and **PopProxyAService.cfg** files as follows, modifying the line default:SSLCertNickNames to "Server-Cert".

   ```
   # cd MessagingServer_home/data/config
   ImapProxyAService.cfg: default:SSLCertNicknames "Server-Cert"
   PopProxyAService.cfg: default:SSLCertNicknames "Server-Cert"
   ```

2. To enable SSL and IMAP, edit the **ImapProxyAService.cfg** file and uncomment the relevant SSL settings.

   A sample **ImapProxyAService.cfg** file resembles the following:

   ```
   # SSL configuration
   #
   #  Enable SSL from client to MMP with this:
   default:SSLEnable         yes
   default:SSLPorts         993
   default:SSLCertNicknames  Server-Cert
   #  Password File for SSL server keys:
   default:SSLKeyPasswdFile Messaging_Server_Root/config/sslpassword.conf
   #  Where SSL session cache, secmod, cert, and key files are located:
   default:SSLCacheDir      Messaging_Server_Root/config
   #  Customizable SSL security module database file name:
   default:SSLSecmodFile     secmod.db
   #  Customizable SSL cert7.db and key3.db file prefixes:
   default:SSLCertPrefix     ""
   default:SSLKeyPrefix      ""
   #  Use SSL on this port when talking to the back-end server (0 = do not use
   SSL)
   default:SSLBacksidePort   993
   default:ServiceList
   /opt/sun/comms/messaging64/lib/ImapProxyAService@1143|1993
   /opt/sun/comms/messaging64/lib/PopProxyAService@1110|1995
   ```

3. To enable SSL and POP, edit the **PopProxyAService.cfg** file and uncomment the relevant SSL settings.

4. Edit the **AService.cfg** file. For SSL and POP, add |1995 after the 1110 in the ServiceList setting. For SSL and IMAP, add |1993 after 1143.

   The **AService.cfg** file should resemble the following:

   ```
   default:ServiceList
   /opt/sun/comms/messaging64/lib/ImapProxyAService@1143|1993
   /opt/sun/comms/messaging64/lib/PopProxyAService@1110|1995
   ```

Configuring IMAP for SSL

To configure IMAP for SSL:

1. Use the **configutil** command to set the following configuration parameters to enable SSL:

   ```
   configutil -o service.imap.enablesslport -v yes
   configutil -o service.imap.enable -v 1
   configutil -o service.imap.sslport -v 993
   configutil -o service.imap.sslusessl -v yes
   ```

2. Use the **configutil** command to set the SSL certificate nickname:

   ```
   configutil -o encryption.rsa.nssslpersonalityssl -v "Server-Cert"
   ```

Configuring POP for SSL

To configure POP for SSL:

1. Use the **configutil** command to set the following configuration parameters to enable SSL:

   ```
   configutil -o service.pop.enable -v 1
   configutil -o service.pop.enablesslport -v yes
   configutil -o service.pop.sslport -v 995
   configutil -o service.pop.sslusessl -v yes
   ```

2. Use the **configutil** command to set the SSL certificate nickname:

   ```
   configutil -o encryption.rsa.nssslpersonalityssl -v "Server-Cert"
   ```

Configuring HTTP for SSL

To configure HTTP for SSL:

1. Use the **configutil** command to set the following configuration parameters to enable SSL:

   ```
   configutil -o service.http.enable -v 1
   configutil -o service.http.enablesslport -v yes
   configutil -o service.http.sslport -v 443
   configutil -o service.http.sslusessl -v yes
   ```

2. Use the **configutil** command to set the SSL certificate nickname:

   ```
   configutil -o encryption.rsa.nssslpersonalityssl -v "Server-Cert"
   ```

Configuring SMTP for SSL

To configure SMTP for SSL:

1. Use the **configutil** command to set the SSL certificate nickname:

   ```
   configutil -o encryption.rsa.nssslpersonalityssl -v "Server-Cert"
   ```

2. To enable SSL encryption for outgoing messages, modify the channel definitions to include the TLS channel options such as **maytls**, **musttls**, and so on.
3. Uncomment the **TLS_PORT** entry (in the **dispatcher.cnf** file located in the *MessagingServer_home*/**config** directory) if you want to support SSL on an alternate port.

   ```
   port 465
   ```

```
TLS_PORT=465
```

Note: You can also use a per-service configuration setting for the SSL certificate nicknames. The configuration parameters, which have the same meaning (that is, the nickname) and override the **encryption.rsa.sslpersonalityssl** setting, are:

- **local.imta.sslnicknames** (for the SMTP and Submit Servers)
- **local.imap.sslnicknames** (for the IMAP Server)
- **local.pop.sslnicknames** (for the POP Server)
- **local.http.sslnicknames** (for the HTTP Server)

Verifying the SSL Configuration

To verify the SSL configuration:

1. Use the **netstat** command to verify that the service is running.

   ```
   netstat -an | grep service.service.sslport
   ```

   where *service* is a keyword **mmp**, **imap**, **pop**, **http**, or **smtp**.

2. Check for errors in the Messaging Server log files.

   Log files are located in the *MessagingServer_home*/**log** directories. For example, check the imap log to ensure that there are no SSL initialization errors (ASockSSL_Init errors).

Configuring the Solaris Cryptographic Framework (SCF)

Because the Solaris Cryptographic Framework software token contains private information, you use the **pktool(1)** command to set a password on the token. This command initializes the user's default keystore by logging in to the system as the application owner (the Messaging Server user).

Setting Up the SCF Software Token Pin

Running the **pktool setpin** command initializes the soft token data store in the **$HOME/.sunw/pkcs11_softtoken/** directory. These files are created to be accessible only to the owner, to protect their contents. This also means that you must perform the initialization as the same user that is used to run Messaging Server (that is, **mailsrv**), so this user has access to the right data store.

To set up the SCF software token pin:

1. Become the **mailsrv** user.

   ```
   su mailsrv
   ```

2. Run the **id** command.

   ```
   id
   uid=207023(mailsrv) gid=6(mail)
   ```

3. Run the **pktool** command.

   ```
   pktool setpin
   ```

4. Create the new passphrase. (The passphrase is not echoed on the screen.)

   You use this passphrase to import the certificate/key pair into the SCF token.

5. Reenter the passphrase. (The passphrase is not echoed on the screen.)

   You are notified that the passphrase is changed.

6. Change to the **/export/mailsrv** directory.

   ```
   cd /export/mailsrv
   ```

7. Check permissions.

   ```
   ls -alrR
   ..:
   total 6
   drwx------   3 mailsrv  mail         512 Nov 16 17:21 .sunw
   drwxr-xr-x   4 root     sys          512 Oct 31 12:31 ..
   drwxrwxrwx   3 root     root         512 Nov 16 17:21 .

   ../.sunw:
   total 6
   drwx------   4 mailsrv  mail         512 Nov 16 17:21 pkcs11_softtoken
   drwxrwxrwx   3 root     root         512 Nov 16 17:21 ..
   drwx------   3 mailsrv  mail         512 Nov 16 17:21 .

   ../.sunw/pkcs11_softtoken:
   total 10
   drwx------   2 mailsrv  mail         512 Nov 16 17:21 public
   drwx------   2 mailsrv  mail         512 Nov 16 17:21 private
   -rw-------   1 mailsrv  mail         103 Nov 16 17:21 objstore_info
   drwx------   3 mailsrv  mail         512 Nov 16 17:21 ..
   drwx------   4 mailsrv  mail         512 Nov 16 17:21 .

   ../.sunw/pkcs11_softtoken/public:
   total 4
   drwx------   4 mailsrv  mail         512 Nov 16 17:21 ..
   drwx------   2 mailsrv  mail         512 Nov 16 17:21 .

   ../.sunw/pkcs11_softtoken/private:
   total 4
   drwx------   4 mailsrv  mail         512 Nov 16 17:21 ..
   drwx------   2 mailsrv  mail         512 Nov 16 17:21 .
   ```

Administering the Cryptographic Framework by Using cryptoadm

The **cryptoadm** utility displays cryptographic provider information for a system, configures the mechanisms for each provider, and installs or uninstalls a cryptographic provider. The Solaris Cryptographic Framework supports three types of providers: a user-level provider (a PKCS#11 shared library), a kernel provider (a loadable kernel software module), and a kernel hardware provider (a cryptographic hardware device). The **cryptoadm** utility provides subcommands to enable and disable the metaslot's features, list metaslot's configuration, and also configure the metaslot's mechanisms policy.

To administer the Cryptographic Framework:

1. You can list all the service providers and their cryptographic mechanisms in the Solaris Cryptographic Framework by running the **cryptoadm list -m** command. Verify that **ncp** and **pkcs11_softtoken.so** are available as cryptographic providers.

Because NCP is a kernel provider, **pkcs11_kernel.so** should appear before **pkcs11_softtoken.so** in the output.

Following is a partial output of this command.

```
# cryptoadm list -m
User-level providers:
=====================
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Mechanisms:
CKM_DSA
CKM_RSA_X_509
CKM_RSA_PKCS

Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
Mechanisms:
CKM_DES_CBC
CKM_DES_CBC_PAD
CKM_DES_ECB
CKM_DES_KEY_GEN
[.... so on ...]
CKM_DSA
CKM_DSA_SHA1
CKM_DSA_KEY_PAIR_GEN
[.... so on ...]
CKM_TLS_MASTER_KEY_DERIVE_DH
CKM_SSL3_KEY_AND_MAC_DERIVE
CKM_TLS_KEY_AND_MAC_DERIVE
CKM_TLS_PRF

Kernel software providers:
==========================
des: CKM_DES_ECB,CKM_DES_CBC,CKM_DES3_ECB,CKM_DES3_CBC
aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR
[.... so on ...]
sha1: CKM_SHA_1,CKM_SHA_1_HMAC,CKM_SHA_1_HMAC_GENERAL
md5: CKM_MD5,CKM_MD5_HMAC,CKM_MD5_HMAC_GENERAL
rsa:CKM_RSA_PKCS,CKM_RSA_X_509,CKM_MD5_RSA_PKCS,CKM_SHA1_RSA_PKCS,CKM_SHA256_
RSA_PKCS,
CKM_SHA384_RSA_PKCS,CKM_SHA512_RSA_PKCS
swrand: No mechanisms presented.

Kernel hardware providers:
==========================
ncp/0: CKM_DSA,CKM_RSA_X_509,CKM_RSA_PKCS
```

2. Disable the use of the following user-level mechanisms, forcing them to be performed by the NCP.

```
# cryptoadm disable provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so \
mechanism=CKM_SSL3_PRE_MASTER_KEY_GEN,\
CKM_SSL3_MASTER_KEY_DERIVE,CKM_SSL3_KEY_AND_MAC_DERIVE,CKM_SSL3_MASTER_KEY_
DERIVE_DH,\
CKM_SSL3_MD5_MAC,CKM_SSL3_SHA1_MAC
```

3. Verify that the user-level mechanisms are disabled.

```
# cryptoadm list -p provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_SSL3_SHA1_MAC,CKM_SSL3_MD5_MAC,CKM_SSL3_MASTER_KEY_DERIVE_DH,CKM_
SSL3_KEY_AND_MAC_DERIVE,
CKM_SSL3_MASTER_KEY_DERIVE,CKM_SSL3_PRE_MASTER_KEY_GEN. random is enabled.
```

Configuring the SCF Provider

NSS uses **secmod.db** to keep track of the PKCS#11 modules available. You can use the security Module Database Tool **modutil**, a CLI that comes with NSS to manage PKCS#11 module information within **secmod.db** files. The Security Module Database Tool enables you to add and delete PKCS#11 modules, change passwords, set defaults, list module contents, and enable or disable slots. The **modutil** CLI is bundled with the Messaging Server software and is located in the *MessagingServer_home*/**bin** directory. This example assumes that you are running **modutil** from the *MessagingServer_ Home*/**lib** directory and **cert8.db**, **secmod.db**, and **key3.db** are located under the **config** directory, that is *MessagingServer_home*/**config**.

1. List all the available PKCS#11 modules.

   Using **modutil**, you can list all the available PKCS#11 modules. By default, NSS has an internal PKCS#11 module.

   ```
   modutil -dbdir ../config -nocertdb -list
   Using database directory ../config...
   Listing of PKCS #11 Modules
   -----------------------------------------------------------
     1. NSS Internal PKCS #11 Module
        slots: 2 slots attached
        status: loaded

        slot: NSS Internal Cryptographic Services
        token: NSS Generic Crypto Services

        slot: NSS User Private Key and Certificate Services
        token: NSS Certificate DB
   ```

2. List the contents of the default NSS soft token.

   The file **sslpassword** contains the password to the Certificate Database.

   ```
   pk12util list-certs -W MessagingServer_home/config/sslpassword
   Alias Valid from Expires on Self-signed?Issued by Issued to
   Server-Cert           2006/11/15 23:02     2007/11/15 23:02   n
   CN=CA,OU=test,O=authority,ST=California,C=US
   CN=foobar.siroe.com,OU=comms,O=Dev,L=Santaclara,ST=California,C=us
   SolCrypto-Framework  2006/11/16 00:14  2007/11/16 00:14        n
   CN=CA,OU=test,O=authority,ST=California,C=US
   CN=SCF,OU=comms,O=Dev,L=Santaclara,ST=California,C=us
   2 certificates found
   ```

Adding the Solaris Cryptographic Framework as a Service Provider

Messaging Server is configured to use the NSS built-in soft token for its cryptographic needs, which employs PKCS#11 to access cryptography. You can modify the Messaging Server configuration to use the User-Level Cryptographic Framework of the Solaris Cryptographic Framework, out of the box, by linking to the **/usr/lib/libpkcs11.so** library to get direct access to the PKCS#11 functionality. That is, you register the Solaris Cryptographic Framework as a PKCS#11 module.

To administer the Cryptographic Framework as a service provider:

1. Register the **/usr/lib/libpkcs11.so** PKCS#11 library with the Messaging Server software and enable the slot named **Sun Metaslot**.

   ```
   modutil -dbdir ../config/ -nocertdb -add "Solaris Crypto Framework" -libfile
   /usr/lib/libpkcs11.so -mechanisms RSA
   WARNING: Performing this operation while the browser is running could cause
   ```

```
corruption
of your security databases. If the browser is currently running,
you should exit browser before continuing this operation. Type 'q <enter>' to
abort,
or <enter> to continue:

Using database directory ../config...

Module "Solaris crypto Framework" added to database.
```

2. Continue with "Enabling the Slot Named Sun Metaslot".

Enabling the Slot Named Sun Metaslot

To enable the slot named **Sun Metaslot**:

1. Run the following **modutil** command.

```
modutil -dbdir ../config/ -nocertdb -disable "Solaris Crypto Framework"

WARNING: Performing this operation while the browser is running
could cause corruption of your security databases.
If the browser is currently running,
you should exit browser before continuing this operation.
Type 'q <enter>' to abort, or <enter> to continue:
Using database directory ../config...

Slot "Sun Metaslot" disabled.
Slot "ncp/0 Crypto Accel Asym 1.0" disabled.
```

2. Run the following **modutil** command.

```
modutil -dbdir ../config/ -nocertdb -enable "Solaris Crypto Framework" -slot
"Sun Metaslot"
WARNING: Performing this operation while the browser is running
could cause corruption of your security databases.
If the browser is currently running,
you should exit browser before continuing this operation.
Type 'q <enter>' to abort, or <enter> to continue:

Using database directory ../config...
Slot "Sun Metaslot" enabled.
```

3. Run the following **modutil** command to verify that the Solaris Crypto Framework is successfully added.

```
modutil -dbdir ../config/ -nocertdb -list
Using database directory ../config...

Listing of PKCS #11 Modules
-----------------------------------------------------------
  1. NSS Internal PKCS #11 Module

     slots: 2 slots attached
     status: loaded

     slot: NSS Internal Cryptographic Services
     token: NSS Generic Crypto Services

     slot: NSS User Private Key and Certificate Services
     token: NSS Certificate DB
  2. Solaris crypto Framework

     library name: /usr/lib/libpkcs11.so

     slots: 2 slots attached
     status: loaded

     slot: Sun Metaslot
     token: Sun Metaslot

     slot: ncp/0 Crypto Accel Asym 1.0
     token: ncp/0 Crypto Accel Asym 1.0
-----------------------------------------------------------
```

Exporting the Certificate/Key Pairs From the NSS Soft Token

This task describes how to export the certificate/key pairs from the NSS soft token (as PKCS#12 formatted files) to be imported in to the SCF software token. The following shows how to export two certificates found in the internal token.

To export the certificate/key pairs from the NSS soft token:

1. Choose the PKCS#12 file password and copy it to a file called **/tmp/pkcs12password**.

   ```
   # echo "pkcspassword" > /tmp/pkcs12password
   ```

2. Run the following commands to export the two certificates found in the Internal Token in to PKCS#12 formatted files.

   ```
   pk12util export-cert -W MessagingServer_home/config/sslpassword -o
   /tmp/Server-Certpk12 -O /tmp/pkcs12password Server-Cert

   # file /tmp/Server-Certpk12

   /tmp/Server-Certpk12:   data
   ```

3. Continue with "Importing the Key/Certificate Pairs to the Sun Metaslot (SCF)".

Importing the Key/Certificate Pairs to the Sun Metaslot (SCF)

To import the key/certificate pairs to the SCF:

1. Run the following **pk12util** command.

   ```
   $ pk12util -i /tmp/Server-Certpk12 -d ../config/ -h "Sun Metaslot"
   Enter Password or Pin for "Sun Metaslot":
   { This is the same password you entered, when running pktool setpin )
   Enter password for PKCS12 file:
   {PKCSpassword : password used to export certificates from the Internal
   Software Token }
   pk12util: PKCS12 IMPORT SUCCESSFUL
   ```

2. Run the following **pk12util** command.

   ```
   $  pk12util -i /tmp/SCFpk12 -d ../config/ -h "Sun Metaslot"
   Enter Password or Pin for "Sun Metaslot":
   { This is the same password you entered, when running pktool setpin )
   Enter password for PKCS12 file:
   {PKCSpassword : password used to export certificates from the Internal
   Software Token }
   ```

```
pk12util: PKCS12 IMPORT SUCCESSFUL
```

3. Continue with "Verifying the Successful Importation of the Certificate/Key Pairs".

Verifying the Successful Importation of the Certificate/Key Pairs

Use this task to verify that the certificate/key pairs were successfully imported to the token. You must be logged in as **mailsrv** (that is, the Messaging Server user).

To verify that the certificate/key pairs were successfully imported:

1. Run the following **certutil** command.

```
$ certutil -L -d ../config/ -h "Sun Metaslot"
Enter Password or Pin for "Sun Metaslot":
( This is the same password you entered, when running pktool setpin. )
Sun Metaslot:Server-Cert                                       u,u,u
Sun Metaslot:SolCrypto-Framework                               u,u,u
```

2. Run the following **certutil** command.

```
$ certutil -K -d ../config/ -h "Sun Metaslot"
Enter Password or Pin for "Sun Metaslot":
( This is the same password you entered, when running pktool setpin. )
<0> Server-Cert
<1> SolCrypto-Framework
```

3. Proceed to the next section to configure Messaging Server.

Configuring Messaging Server to Use the External Token

This section describes the procedures you use to configure the various Messaging Server processes to use the external token.

Configuring Messaging Server Processes to Use the External Token

To configure Messaging Server processes to use the external token:

1. Configure MMP by editing the **ImapProxyAService.cfg** and **PopProxyAService.cfg** files as follows.

```
# cd MessagingServer_home/data/config
ImapProxyAService.cfg: default:SSLCertNicknames  "Sun Metaslot:Server-Cert"
PopProxyAService.cfg:  default:SSLCertNicknames   "Sun Metaslot:Server-Cert"
```

2. Configure the IMAP server by setting the following configuration parameters to use the external token.

```
configutil -o encryption.rsa.nssslpersonalityssl -v "Sun Metaslot:Server-Cert"
```

OR

```
configutil -o local.imap.sslnicknames -v "Sun Metaslot:Server-Cert"
```

3. Configure the POP server by setting the following configuration parameters to use the external token.

```
configutil -o encryption.rsa.nssslpersonalityssl -v "Sun Metaslot:Server-Cert"
```

OR

```
configutil -o local.pop.sslnicknames -v "Sun Metaslot:Server-Cert"
```

4. Configure the HTTP server by setting the following configuration parameters to use the external token.

```
configutil -o encryption.rsa.nssslpersonalityssl -v "Sun Metaslot:Server-Cert"
```

OR

```
configutil -o local.http.sslnicknames -v "Sun Metaslot:Server-Cert"
```

5. Configure the SMTP server by setting the following configuration parameters to use the external token.

Use the **configutil** command to set the SSL certificate nickname:

```
configutil -o encryption.rsa.nssslpersonalityssl -v "Sun Metaslot:Server-Cert"
```

OR

```
configutil -o local.imta.sslnicknames -v "Sun Metaslot:Server-Cert"
```

Note: As shown previously, you can also use a per-service configuration setting for the SSL certificate nicknames. The configuration parameters, which have the same meaning (that is, the nickname) and override the **encryption.rsa.sslpersonalityssl** setting, are:

- **local.imta.sslnicknames** (for the SMTP and Submit Servers)
- **local.imap.sslnicknames** (for the IMAP Server)
- **local.pop.sslnicknames** (for the POP Server)
- **local.http.sslnicknames** (for the HTTP Server)

6. Save the "Sun Metaslot" password in to the **sslpassword.conf** file.

The "Sun Metaslot" is protected by a password. The server prompts for a password every time it starts up. Instead of entering the password every time, Messaging Server reads the password from the **sslpassword.conf** file located in the *MessagingServer_home*/**config** directory. Edit this file as follows.

```
# cat MessagingServer_home/data/config/sslpassword.conf
Sun Metaslot:secret
( "secret" : This is the same password you entered, when running pktool setpin
)
```

Starting and Debuging Messaging Server Services

This task explains how to restart the Messaging Server services, check for errors, and verify the operational SCF environment.

To start and debug Messaging Server services:

1. Restart the Messaging Server services.

```
# MessagingServer_home/bin/start-msg
Connecting to watcher ...
Launching watcher ... 27351
Starting ens server ... 27352
Starting store server .... 27353
Checking store server status .... ready
```

```
Starting imap server .... 27354
Starting pop server .... 27355
Starting http server .... 27356
Starting sched server ... 27357
Starting dispatcher server .... 27359
Starting job_controller server .... 27365
```

2. Ensure that there are no SSL_init errors in the log files, and no ASockSSL_init errors in any of the tcp_smpt, default, imap, pop, and http log files.

   You see something like the following when there is a problem.

```
http:[31/Nov/2006:11:36:21 -0800]
biotite httpd[27356]: General Error: SSLinitialization error:
ASockSSL_Init: couldn't open slot Metaslot (-8127)
imap:[31/Nov/2006:11:36:20 -0800]
biotite imapd[27354]: General Error: SSLinitialization error:
ASockSSL_Init: couldn't open slot Metaslot (-8127)
pop:[31/Nov/2006:11:36:21 -0800]
biotite popd[27355]: General Error:SSL initialization error:
ASockSSL_Init: couldn't open slot Metaslot (-8127)
tcp_smtp_server.log-0J8000L01MGM3Z00:[31/Nov/2006:11:36:22 -0800]
biotite [27363]: General Error:SSL initialization error:
ASockSSL_Init: couldn't open slot Metaslot (-8127)
tcp_smtp_server.log-0J8000L03MGM3Z00:[31/Nov/2006:11:36:22 -0800]
biotite [27364]: General Error:SSL initialization error:
ASockSSL_Init: couldn't open slot Metaslot (-8127)
```

3. Verify that the SSL ports are listening.

```
# netstat -an | grep 995
  *.995              *.*              0      0 49152      0 LISTEN
# netstat -an | grep 443
*.443              *.*              0      0 49152      0 LISTEN
# netstat -an | grep 465
*.465              *.*              0      0 49152      0 LISTEN
```

4. Once the application is operational on the Solaris Cryptographic Framework, use the **kstat** command to display the number of RSA public key decryptions performed using NCP since the last system boot.

   The number of RSA public key decryptions are shown as the **rsapublic** value in the **kstat** output. An incremental and a positive increase in the value of **rsapublic** shows that NCP is operational.

```
# kstat -n ncp0 | grep rsa
rsprivate                     X
rsapublic                     X
```

   In this output:

   - **rsaprivate** -- Total number of jobs submitted to the device for RSA private key operations.
   - **rsapublic** -- Total number of jobs submitted to the device for RSA public key operations.

   By using a browser to connect to the HTTP SSL port and log in, you see how the count increases. For example:

```
<Log in to https://host1.red.example.com>
# kstat -n ncp0 | grep rsa
rsaprivate   35
```

```
rsapublic    146
# kstat -n ncp0 | grep rsa
rsaprivate   38
rsapublic    149
```