

**Oracle® Communications Unified  
Communications Suite**

Certificate Authentication Guide

Release 7.0.6

July 2015

**ORACLE®**

Oracle Communications Unified Communications Suite Certificate Authentication Guide, Release 7.0.6

Copyright © 2007, 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

---

## Contents

1. Certificate Based Authentication for Oracle Communications Unified Communications Suite . .	4
2. Introduction to Certificate-Based Authentication Technology . . . . .	6
3. Certificate-based Authentication for Connector for Microsoft Outlook 7.3 Update 1 . . . . .	9
4. How to Set Up Certificate-based Authentication for Convergence . . . . .	11
5. Certificate Based Authentication for Messaging Server . . . . .	23
6. How to Configure Oracle Communications Sun Calendar Server Client Authentication . . . . .	40
To Configure Oracle Communications Calendar Server Client Authentication . . . . .	47
Enabling the NSS Shared Database Feature in Oracle Communications Calendar Server . . .	54

# Chapter 1. Certificate Based Authentication for Oracle Communications Unified Communications Suite

---

## Certificate Based Authentication Feature for Oracle Communications Unified Communications Suite

Certificate-based authentication is available in Oracle Communications Calendar Server (formerly Sun Java System Calendar Server 6), Convergence, and Connector for Microsoft Outlook for Oracle Communications Unified Communications Suite 7.3 Update 1. (Messaging Server has supported client certificate authentication prior to version 5.2. The enhancement to support dynamic CRL updates was introduced in Messaging Server 7 Update 4 and was "back-ported" to Messaging Server 7 Update 3.)

The feature is not supported by Calendar Server 7, Communications Express, or Instant Messaging.

For a brief introduction to the technology, see [Introduction to Certificate-Based Authentication Technology](#).

Certificate-based authentication is enabled or disabled by default, depending upon the product. The following table shows the default setting for each of the Communications Suite products:

### Certificate Authentication Defaults

Product	Certificate-based Authentication Enabled by Default?
Calendar Server	No
Convergence	No
Messaging Server	No
Outlook Connector	Yes (and cannot be disabled)

The feature is introduced in the following patches:

### Introduction of Certificate Authentication in Patches

Product	Patches
Oracle Communications Calendar Server (formerly Sun Java System Calendar Server 6)	Oracle Solaris SPARC: 121657-43 Oracle Solaris x86 : 121658-43 Red Hat Linux : 121659-4
Convergence	137631-14 - Oracle Solaris SPARC 137632-14 - x86 Oracle Solaris 137633-14 - Red Hat Linux
Messaging Server	Feature available since version 5.2

For detailed information on how to enable and use certificate-based authentication for the Communications Suite products, refer to the following information:

- [How to Configure Oracle Communications Sun Calendar Server Client Authentication](#)
- [How to Set Up Certificate-based Authentication for Convergence](#)
- [Certificate Based Authentication for Messaging Server](#)
- [Certificate-based Authentication for Connector for Microsoft Outlook 7.3 Update 1](#)

# Chapter 2. Introduction to Certificate-Based Authentication Technology

---

## Introduction to Certificate-Based Authentication Technology Used by Unified Communications Suite

Topics:

- End User Authentication Types
- SSL/TLS Overview
- Certificate Authentication Overview
- Certificate and Key Storage Overview
- SSL/TLS/Certificate Standards Overview

### End User Authentication Types

There are three types of end-user authentication used by Communications Suite products:

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. CRAM-MD5/APOP/DIGEST-MD5 passwords are slightly better on the wire, but worse on the back end than those in the clear. Therefore, CRAM-MD5/ADOP/DIGEST-MD5 passwords are not recommended. HTTP-only WebSSO systems most commonly use passwords over SSL/TLS.

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. SSLv3 is also a legacy version which is still used, though it is less secure than TLS.

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, and 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.

### Certificate Authentication Overview

Certificate Authentication involves several components, including:

- 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, for example, 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 that 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 is a list of relevant certificate standards.

- 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 1.2 (RFC 5246).
- SSLv3 is a less secure and less extensible predecessor to TLS 1.0 and later.
- TLS Renegotiation Indication Extension (RFC 5746) fixes a security bug present in all versions of SSL and TLS prior to TLS 1.3. For patches, see [Sun Alert 273350](#).
- 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.
- 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 non-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.
- PKCS#11 is an industry standard that defines how a security library talks to software or hardware that can implement certificate and key storage as well as 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.
- 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.



# Chapter 3. Certificate-based Authentication for Connector for Microsoft Outlook 7.3 Update 1

---



## Certificate-based Authentication for Oracle Communications Connector for Microsoft Outlook 7.3 Update 1



This page contains information for Connector for Microsoft Outlook 8.0.1 and will not be updated in the future. For documentation beginning with Connector for Microsoft Outlook 8.0.2, see the Oracle Technology Network site at:

<http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html>

### Topics:

- [Overview](#)
- [Using Certificate-based Authentication with Outlook Connector](#)
- [Configuration](#)

## Overview

Certificate based authentication is one of the most secure authentication methods. It uses digital certificates which are stored locally in the system, (or in a smart card etc.) to authenticate a user, as opposed to using passwords.

Connector for Microsoft Outlook 7.3 Update 1 supports certificate based authentication for the following back-end services:

1. Mail access service : IMAP + SSL, IMAP + STARTTLS
2. Mail transport service: SMTP + SSL
3. Calendar service : WCAP + SSL
4. Address Book Service : WABP + SSL
5. Corporate Directory : LDAP + SSL
6. User preferences : LDAP + SSL

Connector for Microsoft Outlook supports smart-card certificates and software digital certificates.

## Using Certificate-based Authentication with Outlook Connector

When logged into Connector for Microsoft Outlook, Outlook determines whether the server has been

configured for certificate-based authentication based on the server's response to initiating a connection. If the server is configured for certificate-based authentication, Connector for Microsoft Outlook searches through the local system for eligible certificates with which to log into the server.

The eligible certificates are filtered by:

1. Certificates issued by "trusted CA" by the backend servers.
2. Unexpired certificates.
3. Certificates that can be used for SSL client authentication.

Connector for Microsoft Outlook prompts the user with the list of eligible certificates. The user selects the certificate with which to authenticate to back-end server. If certificate-based authentication fails, Outlook Connector falls back to password-based authentication and prompts user to enter a password to log in. This is also the case if the user doesn't select any certificates from the list.



#### **Note**

If a user's profile is configured by an administrator to only support certificate-based authentication, the user will never be prompted for a password.

If certificate-based authentication fails and password based authentication succeeds for the mail server, then certificate-based authentication will not be attempted for other back-end services, such as SMTP, CS, AB and LDAP. If certificate based authentication succeeds, then certificate-based authentication will be attempted with other back-end services first. If certificate-based authentication fails, password-based authentication will be attempted.

If a user mistakenly chooses a certificate different than the certificate used during the last login, the user is warned that a different certificate is being used to logon from the previous attempt and is given an option not to continue.



#### **Caution**

If both certificate-based and password-based authentication fails with the mail server, Outlook Connector will not proceed authenticating with other back-end services such as Addressbook, Calendar Server and LDAP server.

## **Configuration**

It is possible for an administrator to configure an Connector for Microsoft Outlook user profile such that an authentication certificate is used with the back-end services without the user having to choose the certificate.

A user can configure an Connector for Microsoft Outlook profile so that the certificate information is saved, thus allowing a user to log in without being prompted to select an authentication certificate.

# Chapter 4. How to Set Up Certificate-based Authentication for Convergence

---

## How to Set Up Certificate-based Authentication for Convergence

The following topics (with the exception of the introduction) are the tasks required to set up certificate-based authentication and should be performed in the given order.

Topics:

- [Introduction to Certificates and SSL](#)
- [How to Enable SSL and Client Authentication for a Listener in Oracle GlassFish Server](#)
- [How to Obtain and Manage Certificates in the GlassFish Server Using Network Security Service \(NSS\) Tools](#)
- [How to Configure the Convergence Certificate Mapper](#)
- [How to Use the `iwadmin` CLI Tool to Enable Certificate Authentication Support](#)

## Introduction to Certificates and SSL

### About Digital Certificates

Digital certificates (or simply certificates) are electronic files that uniquely identify people and resources on the Internet. Certificates also enable secure, confidential communication between two entities.

There are different kinds of certificates, such as personal certificates, used by individuals, and server certificates, used to establish secure sessions between the server and clients through secure sockets layer (SSL) technology. For more information on SSL, see [About Secure Sockets Layer](#).

Certificates are based on public key cryptography, which uses pairs of digital keys (very long numbers) to encrypt or encode information so it can be read only by its intended recipient. The recipient then decrypts (decodes) the information to read it.

A key pair contains a public key and a private key. The owner distributes the public key and makes it available to anyone. But the owner never distributes the private key; it is always kept secret. Because the keys are mathematically related, data encrypted with one key can be decrypted only with the other key in the pair.

A certificate is like a passport: it identifies the holder and provides other important information. Certificates are issued by a trusted third party called a Certification Authority (CA). The CA is analogous to passport office: it validates the certificate holder's identity and signs the certificate so that it cannot be forged or tampered with. Once a CA has signed a certificate, the holder can present it as proof of identity and to establish encrypted, confidential communications.

Most importantly, a certificate binds the owner's public key to the owner's identity. Like a passport binds a photograph to personal information about its holder, a certificate binds a public key to information about its owner.

In addition to the public key, a certificate typically includes information such as:

- The name of the holder and other identification, such as the URL of the Web server using the certificate, or an individual's email address.
- The name of the CA that issued the certificate.

- An expiration date.

Digital Certificates are governed by the technical specifications of the X.509 format. To verify the identity of a user in the certificate realm, the authentication service verifies an X.509 certificate, using the common name field of the X.509 certificate as the principal name.

## About Certificate Chains

Web browsers are pre-configured with a set of root CA certificates that the browser automatically trusts. Any certificates from elsewhere must come with a certificate chain to verify their validity. A certificate chain is series of certificates issued by successive CA certificates, eventually ending in a root CA certificate.

When a certificate is first generated, it is a self-signed certificate. A self-signed certificate is one for which the issuer (signer) is the same as the subject (the entity whose public key is being authenticated by the certificate). When the owner sends a certificate signing request (CSR) to a CA, then imports the response, the self-signed certificate is replaced by a chain of certificates. At the bottom of the chain is the certificate (reply) issued by the CA authenticating the subject's public key. The next certificate in the chain is one that authenticates the CA's public key. Usually, this is a self-signed certificate (that is, a certificate from the CA authenticating its own public key) and the last certificate in the chain.

In other cases, the CA can return a chain of certificates. In this case, the bottom certificate in the chain is the same (a certificate signed by the CA, authenticating the public key of the key entry), but the second certificate in the chain is a certificate signed by a different CA, authenticating the public key of the CA to which you sent the CSR. Then, the next certificate in the chain is a certificate authenticating the second CA's key, and so on, until a self-signed root certificate is reached. Each certificate in the chain (after the first) thus authenticates the public key of the signer of the previous certificate in the chain.

About Secure Sockets Layer

Secure Sockets Layer (SSL) is the most popular standard for securing Internet communications and transactions. Web applications use HTTPS (HTTP over SSL), which uses digital certificates to ensure secure, confidential communications between server and clients. In an SSL connection, both the client and the server encrypt data before sending it, then decrypt it upon receipt.

When a Web browser (client) wants to connect to a secure site, an SSL handshake happens:

- The browser sends a message over the network requesting a secure session (typically, by requesting a URL that begins with https instead of http).
- The server responds by sending its certificate (including its public key).
- The browser verifies that the server's certificate is valid and is signed by a CA whose certificate is in the browser's database (and who is trusted). It also verifies that the CA certificate has not expired.
- If the certificate is valid, the browser generates a one time, unique session key and encrypts it with the server's public key. The browser then sends the encrypted session key to the server so that they both have a copy.
- The server decrypts the message using its private key and recovers the session key.

After the handshake, the client has verified the identity of the Web site, and only the client and the Web server have a copy of the session key. From this point forward, the client and the server use the session key to encrypt all their communications with each other. Thus, their communications are ensured to be secure.

The newest version of the SSL standard is called TLS (Transport Layer Security). The Enterprise Server supports the Secure Sockets Layer (SSL) 3.0 and the Transport Layer Security (TLS) 1.0 encryption protocols.

To use SSL, the Enterprise Server must have a certificate for each external interface, or IP address, that accepts secure connections. The HTTPS service of most Web servers will not run unless a digital certificate has been installed. Use the procedure described in [Generating a Certificate Using the keytool](#)

Utility to set up a digital certificate that your Web server can use for SSL.

## About Ciphers

A cipher is a cryptographic algorithm used for encryption or decryption. SSL and TLS protocols support a variety of ciphers used to authenticate the server and client to each other, transmit certificates, and establish session keys.

Some ciphers are stronger and more secure than others. Clients and servers can support different cipher suites. Choose ciphers from the SSL3 and TLS protocols. During a secure connection, the client and the server agree to use the strongest cipher they both have enabled for communication, so it is usually sufficient to enable all ciphers.

## Using Name-based Virtual Hosts

Using name-based virtual hosts for a secure application can be problematic. This is a design limitation of the SSL protocol itself. The SSL handshake, where the client browser accepts the server certificate, must occur before the HTTP request is accessed. As a result, the request information containing the virtual host name cannot be determined prior to authentication, and it is therefore not possible to assign multiple certificates to a single IP address.

If all virtual hosts on a single IP address need to authenticate against the same certificate, the addition of multiple virtual hosts probably will not interfere with normal SSL operations on the server. Be aware, however, that most browsers will compare the server's domain name against the domain name listed in the certificate, if any (applicable primarily to official, CA-signed certificates). If the domain names do not match, these browsers display a warning. In general, only address-based virtual hosts are commonly used with SSL in a production environment.

## How to Enable SSL and Client Authentication for a Listener in Oracle GlassFish Server

### HTTP Listeners

Each virtual server provides connections between the server and clients through one or more HTTP listeners. Each HTTP listener is a listen socket that has an IP address, a port number, a server name, and a default virtual server.

HTTP listeners must have a unique combination of port number and IP address. For example, an HTTP listener can listen on all configured IP addresses on a given port for a machine by specifying the IP address 0.0.0.0. Alternatively, the HTTP listener can specify a unique IP address for each listener, but use the same port.

Since an HTTP listener is a combination of IP address and port number, you can have multiple HTTP listeners with the same IP address and different port numbers (for example, 1.1.1.1:8081 and 1.1.1.1:8082), or with different IP addresses and the same port number (for example, 1.1.1.1:8081 and 1.2.3.4:8081, if your machine was configured to respond to both these addresses).

However, if an HTTP listener uses the 0.0.0.0 IP address, which listens on all IP addresses on a port, you cannot create HTTP listeners for additional IP addresses that listen on the same port for a specific IP address. For example, if an HTTP listener uses 0.0.0.0:8080 (all IP addresses on port 8080), another HTTP listener cannot use 1.2.3.4:8080.

Because the system running the Enterprise Server typically has access to only one IP address, HTTP listeners typically use the 0.0.0.0 IP address and different port numbers, with each port number serving a different purpose. If the system does have access to more than one IP address, each address can serve a different purpose.

By default, when the Enterprise Server starts, it has the following HTTP listeners:

- Two HTTP listeners named `http-listener-1` and `http-listener-2`, associated with the virtual server named `server`.
- An HTTP listener named `admin-listener`, associated with the virtual server named `__asadmin`.

All these listeners use the IP address 0.0.0.0 and the port numbers specified as the HTTP server port numbers during installation of the Enterprise Server. If the Enterprise Server uses the default port number values, `http-listener-1` uses port 8080, `http-listener-2` uses port 8181, and `admin-listener` uses port 48489.

Each HTTP listener has a default virtual server. The default virtual server is the server to which the HTTP listener routes all request URLs whose host component does not match any of the virtual servers that are associated with the HTTP listener (a virtual server is associated with an HTTP listener by listing the HTTP listener in its `http-listeners` attribute).

In addition, specify the number of acceptor threads in the HTTP listener. Acceptor threads accept new connections and put them onto a connection queue. Session threads then pick up connections from the queue and service the requests. The server posts more session threads if required at the end of the request.

A set of request processing threads retrieves incoming HTTP requests from the connection queue and processes the requests. These threads parse the HTTP headers, select the appropriate virtual server, and run through the request processing engine to service the request. When there are no more requests to process, but the connection can be kept persistent (either by using HTTP/1.1 or sending a `Connection: keep-alive` header), the request processing thread assumes the connection to be idle and passes the connection to the Keep-Alive connection management subsystem.

The Keep-Alive subsystem periodically polls such idle connections and queues those connections with activity into the connection queue for future processing. From there, a request processing thread again retrieves the connection and processes its request. The Keep-Alive subsystem is multi-threaded, as it manages potentially tens of thousands of connections. Efficient polling techniques are used, by dividing the number of connections into smaller subsets, to determine which connections are ready with requests and which of those connections have idled for sufficient time to deem them closed (beyond a maximum permissible Keep-Alive timeout).

The HTTP listener's server name is the host name that appears in the URLs the server sends to the client as part of a redirect. This attribute affects URLs the server automatically generates; it does not affect the URLs for directories and files stored in the server. This name is normally the alias name if the server uses an alias. If a client sends a `Host:` header, that host name supersedes the HTTP listener's server name value in redirects.

Specify a redirect port to use a different port number from that specified in the original request. A redirect occurs in one of these situations:

- If a client tries to access a resource that no longer exists at the specified URL (that is, the resource has moved to another location), the server redirects the client to the new location (instead of returning a 404), by returning a designated response code and including the new location in the response's `Location` header.
- If a client tries to access a resource that is protected (for example, SSL) on the regular HTTP port, the server redirects the request to the SSL-enabled port. In this case, the server returns a new URL in the `Location` response header, in which the original insecure port has been replaced with the SSL-enabled port. The client then connects to this new URL.

Specify also whether security is enabled for an HTTP listener and what kind of security is used (for example, which SSL protocol and which ciphers).

## How to Enable How SSL and client authentication for listener in Oracle

## GlassFish server

1. Log into the GlassFish Administration Console.
2. Click on Configurations -> server-config -> HTTP Service -> HTTP Listener -> Click the listener where SSL is enabled -> Click on SSL tab in the right pane -> Select the check box "Client Authentication Enabled."
3. Save Changes.
4. Restart the GlassFish Application Server.

## Accessing a Web Application Deployed on GlassFish Server

To access a web application deployed on the GlassFish Server, use the URL <http://localhost:8080/> (or <https://localhost:8181/> if it is a secure application), along with the context root specified for the web application. To access the Admin Console, use the URL <https://localhost:4848/> or <http://localhost:4848/asadmin/> (its default context root).

Because a virtual server must specify an existing HTTP listener, and because it cannot specify an HTTP listener that is already being used by another virtual server, create at least one HTTP listener before creating a new virtual server. The above restriction applies only if the new virtual server has the same host or alias name as one of the virtual servers already bound to the existing HTTP listener.

### HTTP listener Sub-elements, Attributes and Properties

For detailed information about the available sub-elements, attributes and properties for the `http-listener` element, see [Tables 1-71 - 1-73](#).

## How to Obtain and Manage Certificates in the GlassFish Server Using Network Security Service (NSS) Tools

The following NSS tools are available to obtain and manage certificates in the GlassFish server:

- `certutil`, a command-line utility for managing certificates and key databases.
- `pk12util`, a command-line utility used to import and export keys and certificates between the certificate/key databases and files in PKCS12 format.
- `modutil`, a command-line utility for managing PKCS #11 module information within `secmod.db` files or within hardware tokens.

For detailed instructions and examples of how to use the aforementioned utilities, see [Using Network Security Service \(NSS\) Tools](#).

## Certificate Revocation Support in GlassFish Server

There are two types of Certificate Revocation Lists, static and dynamic. The following methods of certificate revocation are supported:

1. Static Certificate Revocation List (CRL) checking
2. Dynamic Certificate Revocation List (CRL) checking
3. Revocation Checking using OCSP (Online Certificate Status Protocol)

For detailed information on how to enable revocation checking, see [SSL and CRL Checking in Oracle GlassFish Server](#)

## How to Configure the Convergence Certificate Mapper

### Introduction to the Certificate Mapper

A certificate mapper is used to map a certificate presented by a client to the user in the directory that should be associated with that certificate. Certificate mapping determines how Convergence server scans user entries in the LDAP directory. Use the `certmap.conf` to configure how a client certificate is mapped to an LDAP entry. You can edit the `certmap.conf` file to add entries to match the structure of your LDAP directory and to list the certificates you want your users to have. Users can be authenticated based on attributes and their corresponding values used in the `subjectDN`.

The mapping file defines the following information:

- The LDAP tree location the server from where begins its search (Base DN)
- Certificate attributes the server should use as search criteria when searching for the entry in the LDAP directory
- Whether the server must go through an additional verification process

## Mapper file syntax

The file contains one or more named mappings, each applying to a different CA. **Note that only lower-case entries can be used in `<name>`.** A mapping has the following syntax:

```
certmap=<name1> , <name2> , <name3> . . .  
  
<name1> . <property1> = <value>  
  
<name1> . <property2> = <value>
```

For example:

```
certmap default , <name> , <name1>  
default . <property1> = <value>  
  
default . <property2> = <value>  
  
<name> . <property1> = <value>  
<name> . <property2> = <value>  
  
<name1> . <property1> = <value>  
<name1> . <property2> = <value>
```

The first line specifies a name(s) for the entry. You can use any name for the entry. If a mapper entry for a specific CA (as present in the `IssuerDN` in the client certificate) does not exist, the default configuration/mapping is used. Thus, it is recommended to configure default mappings.

## Properties and Values

The `certmap.conf` has the following properties:

### IssuerDN

The certificate authority `IssuerDN` of the certificates to which the map applies. (DN format as defined in RFC 2253). The `IssuerDN` must match the `IssuerDN` of the CA that issued the client certificate. For example, the following two `IssuerDN` lines have different space separating the attributes, but the server treats these as two separate entries



```
certmap sun1 ou=Sun Certificate Authority,o=Sun, c=US
certmap sun2 ou=Sun Certificate Authority, o=Sun, c=US
```

### Syntax

The following are the valid syntax for `IssuerDN`:

- empty - The corresponding map is invalid. The server will fall back to use the default map. (For example: `usps.IssuerDN=`)
- commented out/not set - The corresponding map is invalid. The server will fall back to use the default map.
- Valid RFC 2253 DN - This map will be used for all the certificates, whose `IssuerDN` matches this DN.

### DNComps

The `DNComps` property is used to configure the LDAP tree location from where the server begins its search (Base DN)

Specifies a comma separated list of relative distinguished name components of the base DN for an LDAP search to find the user entry matching the certificate. The components are taken from the subject DN of the certificate

The server gathers values for these attributes from the client certificate and uses the values to form an LDAP DN, which determines where the server starts its search in the LDAP directory. For example, if you set `DNComps` to use the `o` and `c` attributes of the DN, the server starts the search from the `o=<org>`, `c=<country>` entry in the LDAP directory, where `<org>` and `<country>` are replaced with values from the DN in the certificate.

Any attribute for which a mapping is defined but is not contained in the certificate subject is not included in the generated base DN.

### Syntax

The following are the valid syntax for `DNComps`:

- empty - The server performs a sub-tree search with base DN as the configured Base DN (`dcroot`) in convergence configuration. (For example: `default.DNComps=`)
- commented out - take the existing user's DN from the cert and this becomes the Base DN. (For example: `#default.DNComps=ou,o`)
- attribute names - a comma separated list of attributes to form DN. The attribute values are taken from the subject DN of the certificate.  
attribute substitutions - attribute and their corresponding LDAP attribute to be used as a substitute while constructing the Base DN. The attribute values are taken from the subject DN of the certificate. (For example: `default.DNComps=ou=OrgUnit,o`)

### Supported Attributes

The following specified attributes are supported in this mapper (as listed in RFC 2253 and RFC 5280):

Attribute Keyword	Description
cn	Common Name
l	location
street	Street
ou	Organization Unit
o	Organization
c	Country
uid	Unique ID
emailaddress	RFC 822 Email Address
s	State
serialnumber	Serial Number for a name
dnq/dnqqualifier	DN specific information
t	Person's title (rank)
surname	Last Name
givenname	First Name
initials	Initials
generation	denoting Jr. 'III', etc.
ip	IP address

### FilterComps

`FilterComps` specifies a comma separated list of attributes to form a filter for an LDAP search to find the user entry matching the certificate. The values for the filter are taken from the Subject DN of the client certificate.

If multiple attribute mappings are defined, then the server combines them with an AND search. For example, if two mappings are defined `cn` and `uid`, and the server is presented with a certificate having a subject of `UID=john.doe@example.com,CN=John Doe,O=Example Corp,C=US`, then it generates a search filter of `(&(cn=John Doe)(uid=john.doe@example.com))`.

Any attribute for which a mapping is defined but is not contained in the certificate subject is not included in the generated search filter. All attributes that can be used in generated search filters should have corresponding indexes in all back-end databases that can be searched by this certificate mapper. For the mapping to be successful, the generated search filter must match one user in the directory (within the scope of the base DNs for the mapper). If no users or multiple users match the generated criteria, the mapping fails.

### Syntax

The following are the valid syntax for `FilterComps`:

- empty - The generated search filter is `(objectclass=*)` (For example: `default.FilterComps=`)
- commented out - The generated search filter is `(objectclass=*)` (For example: `#default.DNComps=cn`)
- attribute names - a comma separated list of attributes to form search. The attribute values are taken from the subject DN of the certificate. If multiple attribute mappings are defined, then the

server combines them with an AND search.

- attribute substitutions - attribute and their corresponding LDAP attribute to be used as a substitute while constructing the search filter. The attribute values are taken from the subject DN of the certificate. (For example: `default.FilterComps=uid=employeeID,emailaddress=mail`)

### Supported Attributes

The following specified attributes are supported in this mapper (as listed in RFC 2253 and RFC 5280):

Attribute Keyword	Description
cn	Common Name
l	location
street	Street
ou	Organization Unit
o	Organization
c	Country
uid	Unique ID
emailaddress	RFC 822 Email Address
s	State
serialnumber	Serial Number for a name
dnq/dnqualifier	DN specific information
t	Person's title (rank)
surname	Last Name
givenname	First Name
initials	Initials
generation	denoting Jr. 'III', etc.
ip	IP address

### CmapLdapAttr

`CmapLdapAttr` specifies the name of the LDAP attribute in the directory containing the subject DN of the certificate.

Unless this attribute is empty, the `baseDN` is the convergence configured Base DN (DC root) with a search filter containing subject DN. For the mapping to be successful, the certificate mapper must match exactly one user (within the scope of the base DN's for the mapper). If no entries or multiple entries match, the server performs a search with the `baseDN` generated from the corresponding `DNComps` and the search filter from the corresponding `FilterComps`.

### Syntax

The following is valid syntax for `CmapLdapAttr`:

- attribute name - is a name for the attribute in the LDAP directory that contains subject DN's from all certificates

belonging to the user. (For example: `default.CmapLdapAttr=certSubjectDN`)

## VerifyCert

`VerifyCert` tells the server whether it should compare the client's certificate with the certificate found in the LDAP directory. It takes two values: **on** and **off**. This feature is useful to ensure your end-users have a valid, un-revoked certificate. However, you should only use this property if your LDAP directory contains certificates.

The default behavior is the same as **off**, meaning client certificates are not checked to be valid and not revoked.

## Syntax

The following is the valid syntax for `VerifyCert`:

- **on** - compare's client certificate with the certificate found in the `userCertificate` attribute in user's LDAP entry  
(For example: `default.VerifyCert=on`)

## Sample Mappings for the `certmap.conf`

The `certmap.conf` file should have at least one entry. The following examples illustrate the different ways you can use the `certmap.conf` file.

### Example 1

This example represents a `certmap.conf` file with only one default mapping:

```
certmap=default
default.DNComps=ou, o, c
default.FilterComps= emailaddress=mail, uid
default:verifycert=on
default.issuerDN=default
```

Using this example, the server starts its search at the LDAP branch point containing the entry `{{ou=<orgunit>, o=<org>, c=<country>}}` where the text within `<>` is replaced with the values from the subject's DN in the client certificate.

The server then uses the values for email address and userid from the certificate to search for a match in the LDAP

directory. The search filter would be `&(mail=<email>)(uid=<uid>)`. When it finds an entry, the server verifies the certificate by comparing the one the client sent to the one stored in the directory.

### Example 2

The following example file has two mappings, one for default and another for the US Postal Service (USPS):

```
certmap=default,usps
default.DNComps=
default.FilterComps=emailaddress, uid
usps.IssuerDN=ou=United States Postal Service, o=usps, c=US
usps.DNComps=ou,o,c
usps.FilterComps=emailaddress=mail
usps.verifycert=on
default.issuerDN=default
```

When the server gets a certificate from someone other than the USPS, it uses the default mapping, which starts at the configured base DN (DC root) and searches for an entry matching the client's user ID and email address. If the certificate is from the USPS, the server starts its search at the base DN containing the organizational unit and searches for matching email addresses. Note that if the certificate is from the USPS, the server verifies the certificate; other certificates are not verified.



### CAUTION

The `IssuerDN` in the certificate must be identical to the `IssuerDN` listed in the first line of the mapping. In the previous example, a certificate from an issuer DN that is `o=United States Postal Service,c=US` will not match because there is no space between the `o` and the `c` attributes.

### Example 3

The following example uses `CmapLdapAttr` to scan the LDAP database for the `certSubjectDN` attribute whose value exactly matches the entire subject DN taken from the client certificate.

```
certmap=myco
myco.IssuerDN=ou=My Company Inc, o=myco, c=US
myco.CmapLdapAttr=certSubjectDN
myco.DNComps=o, c
myco.FilterComps=emailaddress=mail, uid
myco.verifycert=on
```

If the client certificate subject is:

```
uid=Walt Whitman, o=LeavesOfGrass Inc, c=US
```

the server first searches for entries that contain the following information:

```
certSubjectDN=uid=Walt Whitman, o=LeavesOfGrass Inc, c=US
```

If a matching entry is found, the server verifies the entry. If no matching entries are found, the server uses

`DNComps` and `FilterComps` to search for matching entries. In this example, the server would search for `uid=Walt Whitman` in all entries under `o=LeavesOfGrass Inc, c=US`.

## How to Use the `iwadmin` CLI Tool to Enable Certificate Authentication Support

To enable certificate authentication support using the `iwadmin` CLI Tool, set the `auth.cert.enable` option to `=true`. For details, see [CLI tool Overview](#).



### Note

Beginning with GlassFish Server 2.1.1 Patch 6, if certificate authentication support is enabled when a user accesses Convergence without certificate authentication, then certificate authentication support will fall back to form-based authentication. To ensure proper fallback, add the `<property name="com.sun.grizzly.ssl.auth" value="want" />` to the `http-listener` on your GlassFish Server.

Once you have enabled certificate authentication support, restart the GlassFish Server.

### **How To Enable Fallback to Form-based Authentication with Certificate-based Authentication Disabled**

If the form-based option is set to true, and certificate-based authentication is disabled, when the user accesses Convergence without certificate authentication, a login page is displayed. Additionally, Convergence can be accessed on another port. (This feature will be supported in future releases of GlassFish).

To enable form-based authentication, set `auth.cert.enablefallback` to true.

# Chapter 5. Certificate Based Authentication for Messaging Server

---

## Certificate Based Authentication for Messaging Server

Topics:

- Introduction: SSL/TLS, Client Certificates and CRLs
- SSL/TLS Tools Available in Communications Suite Installer
- Certificate and Key Storage
- SSL/TLS Configuration
- SSL/TLS Tasks
- Sample Protocol Sessions with Client Certificate Authentication
- SSL/TLS Best Practices
- Messaging Server and SSL/TLS: Known Limitations
- Appendix A. References

### Introduction: SSL/TLS, Client Certificates and CRLs

The following discusses issues related to configuring Messaging Server to use client certificates with CRLs with 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/DIGEST-MD5 passwords are slightly better on the wire, but worse on the back-end than those in the clear. Therefore, CRAM-MD5/ADOP/DIGEST-MD5 passwords are not recommended. (DIGEST-MD5 has been removed from Messaging Server 7 Update 4). 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. SSLv3 is also a legacy version which is still used, though it is less secure than TLS.

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 (e.g., 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.

## Certificate Authentication Overview

Certificate Authentication involves several components:

- a public key (information shared with other parties)
- an associated private key (secret information)
- a certificate authority (an authority that asserts someone in possession of a particular private key has a certain identity)
- a 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 certificate 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 7 Update 4:


- 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 1.2 (RFC 5246).
- SSLv3 is a less secure and less extensible predecessor to TLS 1.0 and later.
- TLS Renegotiation Indication Extension (RFC 5746) fixes a security bug present in all versions of SSL and TLS prior to TLS 1.3. For patches, see [Sun Alert 273350](#)
- 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 in order 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.

 `imap`s 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.

 "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 non-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 `mshhttpd` 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.
- PKCS#11 is an industry standard that defines how a security library talks to software or hardware that can implement certificate and key storage as well as 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 Communications Suite Installer

`SUNwtlsu`

The `SUNWt1su` (Solaris) and `sun-nss` (Linux) packages installed by the Communications Suite Installer include utilities that can be used to manage keys and certificates for Messaging Server. On Solaris, the utilities are installed in `/usr/sfw/bin` by default. On Linux they are installed in `/opt/sun/private/bin` by default. Make sure to supply the option `-d /var/opt/sun/comms/messaging64/config` (substituting the appropriate path for your installation) to use these tools with the Messaging Server's configuration directory. If you run these utilities as `root`, you must manually "chown" the appropriate output files (`cert8.db`, `key3.db`, `secmod.db` or `cert9.db`, `key4.db`, `pkcs11.txt`) so that they are owned by the Messaging Server user.

## Utilities Used to Manage Certificates

- 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 as well as 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 available in the Messaging Server's bin 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 which 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

When Messaging Server 7 Update 4 (or earlier) is used with the default PKCS#11 module (`cert8.db`, `key3.db`, `secmod.db`), modifications to the certificate and key databases are not permitted while the server is running. Such changes can corrupt internal data structures and cause servers to crash. The command `configutil -o local.ssldblegacy -v 0` 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 only if the administrator modifies his environment as follows:

```
~/bashrc: export NSS_DEFAULT_DB_TYPE="sql"  
~/cshrc: setenv NSS_DEFAULT_DB_TYPE "sql"
```

If a site is using a legacy certificate format, the following commands will 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 sql:. -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
```

**i** Executing the latter two commands prevent the `certutil` and `pk12util` utilities from using the old databases, if you happen 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. Starting with Messaging Server 7.0.5.31.0, the NSS version is displayed by the `imsimta` version command. To check the version of NSS on older releases, do the following:

For Solaris: `pkgparam -v SUNWt1s SUNW_PRODVERS`  
 For Linux: `rpm -q qf='%{N}%{V}%{R}.%{arch}\n' sun-nss`

It is recommended that you upgrade to Messaging Server 7.0.5.31.0 or later (NSS 3.16 or later).

## SSL/TLS Configuration

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

### `configutil` SSL-related Settings

The following is re is a summary of the available SSL-related settings, which do not apply to the MMP unless explicitly mentioned.

- `encryption.rsa.nssslpersonalityssl` - Sets the certificate nickname of the server certificate used by the Messaging Server by default.
- `local.defdomain` - Default domain for authentication if one not explicitly specified; including AUTH EXTERNAL.
- `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.
- `local.ldapusessl` - If set, use `ldaps` (LDAP-over-SSL default port 636) when contacting an LDAP server.
- `local.service.proxy.imapssl` - If set, use `imaps` (IMAP-over-SSL default port 993) when proxying to another IMAP backend for shared folders. Defaults to 1 if IMAP port is 993.
- `service.http.sslusessl`, `service.imap.sslusessl`, `service.pop.sslusessl` - Enable use of SSL/STARTTLS for these services.

- `service.http.sslport`, `service.imap.sslport`, `service.pop.sslport` - Specify a non-default SSL port.
- `service.http.enablenesslport`, `service.imap.enablenesslport`, `service.pop.enablenesslport` - Enable use of SSL on a separate port for this service.
- `service.http.plaintextmincipher`, `service.imap.plaintextmincipher`, `service.pop.plaintextmincipher`, `service.imta.plaintextmincipher` (new in Messaging Server 7 Update 4) - 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, as well as the use of `tlsswitchchannel` and `saslswitchchannel`. The use of `tlsswitchchannel` is no longer necessary.
- `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.)
- `local.ldapcheckcert` - Verifies that the LDAP server's certificate is valid when using LDAPSSL (default = 1). Used by MMP in Messaging Server 7 Update 4.
- `local.ldaptrace` - Enables additional LDAP diagnostics. (For Messaging Server 7 Update 3, logs LDAP connection creation/deletion.) In Communications Suite 7 update 1, enables detailed protocol logging for LDAP.
- `local.ssladjustciphersuites` - Determines which SSL cipher suites are enabled for use.
- `local.sslcompress` (new in Messaging Server 7 Update 4) - Enables SSL/TLS compression (per RFC 3749). Default = 1 and is used by MMP.
- `local.ssldbpath` - Determines where SSL certificate and key databases are stored (defaults to configuration directory). Using another directory is discouraged.
- `local.ssldblegacy` (new in Messaging Server 7 Update 4) - When set to 0, enables use of the new certificate and key database format. Used by MMP.
- `local.ssldbprefix` - A prefix used in the the certificate and key database files names. Changing this is discouraged.
- `local.sslpkix` (new in Messaging Server 7 Update 4, restricted use) - 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.
- `local.sslrequiresafenegotiate` (new in mirasol) - Requires use of the SSL/TLS renegotiate extension (per RFC 5746). Default = 0. Used by MMP.
- `local.sslv3enable` (new in new in Messaging Server 7 Update 4) - When set to 0, `local.sslv3enable` disables support for SSLv3. Default = 1. Used by MMP.
- `service.imap.indexer.sslusessl` (new in Messaging Server 7.0.5.31) - `imap.indexer.sslusessl` in Unified Configuration - If set to 1, the IMAP server uses SSL to authenticate with the Indexing and Search Service using the `service.imap.indexer.port` option (Unified Configuration) or `service.imap.indexer.port` option (legacy configuration).

## Dispatcher SSL-related Settings

When a dispatcher service configuration block includes "TLS\_PORT", the server will negotiate SSL/TLS

immediately prior to starting the specified application. This is typically used to enable the non-standard `smtps` submission 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's rarely used for SMTP relay).

## Messaging Transfer Agent (MTA) SSL-related Channel Keywords

`imta.cnf` Channel Keywords

Keyword	Description
<code>maytlserver</code>	Allow use of TLS connecting to the server for that channel
<code>musttlserver</code>	Require use of TLS connecting to the server for that channel
<code>tlswitchchannel</code>	Change channels if TLS successfully negotiated
<code>maytlsclient</code>	Use TLS if available for outbound mail sent via that channel
<code>musttlsclient</code>	Require TLS for outbound mail sent via that channel
<code>maysaslserver</code>	Allow use of SMTP AUTH (including AUTH EXTERNAL)
<code>mustsaslserver</code>	Require use of SMTP AUTH (including AUTH EXTERNAL)
<code>saslswitchchannel</code>	Change channels if SASL authentication successful

SMTP channel options:

`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 cert 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:

```
{$Y<ruleset>|<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 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

The following MMP options appear in PopProxyAService.cfg, ImapProxyAService.cfg and/or SmtProxyAService.cfg. Note that 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.

SSLCacheDir - The directory to search for cert & 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.

SSLKeyPrefix - Filename prefix used when locating the key3.db or key4.db.

SSLSecmodFile - Specifies alternate file name/path for secmod.db.

The following MMP options can be included in PopProxyAService.cfg, ImapProxyAService.cfg and SmtProxyAService.cfg. The options can also be included in vdmapp.cfg and may have different settings for each service or virtual domain:

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.

The following MMP options may have different settings for each service:

`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) as well as by schema 2.

The following MMP option is only available in `vdmap.cfg`:

`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. The following are the available options:

`DNComps`

commented out - take the user's DN from the cert 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**

### **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 sql:. -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.

Make sure that the certificate, key, and **pkcs11.txt** files are owned by the messaging server user by performing the following:

```
% certutil -S -d sql:. -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 sql:. -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 sql:. -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:

```
% configutil -o service.imap.sslusessl -v 1
% 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 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 sql:. -G -n CA-Cert << EOF
update=20100510200000Z
addcert 1 20100510200000Z
EOF
```

Note that the "1" in the `addcert` above 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:

<http://www.mozilla.org/projects/security/pki/nss/ref/ssl/sslerr.html>. String error text and numeric error codes are outputted in order 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 includes example protocol transcripts where client certificate authentication is used, using both standard and non-standard protocols.

### Standard 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.sun.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 are required:

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 certs 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.

Note that 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
====

```

## Standard Submission (STARTTLS) Default port 587

```

====
S: 220 nifty-silver.west.sun.com – Server ESMTP (Oracle Communications Messaging Server 7u5-0.01
32bit (built Apr 20 2010))
C: EHLO nifty-silver.west.sun.com
S: 250-nifty-silver.west.sun.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.sun.com
S: 250-nifty-silver.west.sun.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 is required:

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 a CA certificate trusted to sign client certs installed.
4. The relevant channel must include the `maytlssserver/musttlssserver` channel keyword. 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 keyword.
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 via standard protocol.

## Standard POP (STLS) default port 110

```

=====
S: +OK nifty-silver.west.sun.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 above executes in the same way as the standard IMAP (STARTTLS) with the same requirements.

### Non-standard `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.sun.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 that specific user has been logged in already. If the server sends "\*" OK" then this failed and the client must proceed with standard password authentication.

### Non-standard `smtps` typical port 465

```

=====
...TLS-negotiation-with-client-cert...
S: 220 nifty-silver.west.sun.com – Server ESMTP (Oracle Communications Messaging Server 7u5-0.01
32bit (built Apr 20 2010))
C: EHLO nifty-silver.west.sun.com
S: 250-nifty-silver.west.sun.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 SMTP STARTTLS: clients do not know if the server accepted the client certificate unless the "EXTERNAL" mechanism is available. Some third-party clients fail too look for and use the EXTERNAL mechanism.

## Non-standard pops typical port 995

```
====
...TLS-negotiation-with-client-cert...
S: +OK nifty-silver.west.sun.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 non-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 blindly 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:

- Make sure that the installed NSS version is 3.16 or later.
- Use a 2048-bit RSA certificate with a SHA256 signature. Do not use the deprecated `msgcert` tool to generate a server certificate as it does not support these two options.
- 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. Make sure `local.ldapcheckcert = 1` when earlier versions of Messaging Server.
- Set `RestrictPlainPassword` and `plaintextmncipher` 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) these can be used in addition to SSL/TLS on any of the regular ports (143, 110, 587). With the `RestrictPlainPassword` and `plaintextmncipher` options, there is no significant security difference between the regular and SSL-only ports.
- Use `ssladjustciphersuites` to enable `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_CBC_!`
- Set `tlsv12enable` to 1.

- If possible, disable weaker cipher suites with `ssladjustciphersuites` (particularly the RC4 cipher suites).
- Disable SSLv3 by setting `local.sslv3enable` to 0.

## 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 via 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.

### Protocols with Unspecified Client Certificate Behavior

Both IMAP4 and POP3 have well documented STARTTLS standards that cover client certificate behavior (published in June 1999). Messaging Server interoperates with standards compliant clients. Due to IMAP4's standard PREAUTH feature, Messaging Server's client certificate support on the undocumented `imaps` port is likely to interoperate with standards-compliant IMAP4 clients implementing `imaps`. However, administrative proxy authentication is made unavailable on that port as a result). Neither SMTP nor `pop3s` has well defined standards for client certificates.

### Administrative Proxy with a Certificate

Except for the "imaps" service, all of Communications Suite servers (IMAP, POP3, SMTP Submission) support administrative proxy using a client certificate. If the administrator provides a certificate that maps to a known store admin identity using `certmap`, that administrator can authenticate as another user via 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 backend server, they only support administrative proxy using plain text passwords, which are optional over the SSL port. Support for administrative proxy via 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.

### LDAP Client Authentication Limitations

The Messaging Server LDAP client only supports plaintext password authentication optionally over the SSL port for LDAP (636). The client does not support LDAP STARTTLS, client certificate authentication to LDAP or other authentication mechanisms for LDAP.

### 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 via 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. The ENS server only supports authentication using `service.ens.domainallowed` and `service.ens.domainnotallowed`, where filtering is based on the IP address.

## 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 **`vdmmap.cfg`** settings. The MTA's `PORT_ACCESS` mapping supports standard SMTP submission with `STARTTLS`, although non-standard `smtps` via 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.

In order 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.

## Appendix A. References

[Technical Documentation for new database format in NSS](#)

[NSS Shared DB Howto \(Primarily for Firefox/Thunderbird\)](#)

[Documentation for certutil](#)

[Documentation for crlutil](#)

[Key Manager \(add on for Firefox to provide GUI key management for NSS\)](#)

[Numeric SSL/TLS Error Codes](#)

# Chapter 6. How to Configure Oracle Communications Sun Calendar Server Client Authentication

---

## How to Configure Oracle Communications Sun Calendar Server Client Authentication

This information discusses how to configure Oracle Communications Sun Calendar Server (formerly Sun Java System Calendar Server, or Calendar Server 6) to use client certificates with CRLs with SSL/TLS.

Topics:

- [Authentication Technology Overview](#)
- [SSL/TLS Overview](#)
- [Certificate Authentication Overview](#)
- [Certificate and Key Storage Overview](#)
- [SSL/TLS/Certificate Standards Overview](#)
- [SSL/TLS Tools Available in Communications Suite Installer](#)
- [Utilities Used to Manage Certificates](#)
- [Certificate and Key Storage](#)
- [SSL/TLS Configuration](#)
- [How to Create and Install a Self-signed CA Certificate and Key](#)
- [How to Create and Install a CA-signed Server Certificate and Key](#)
- [How to Create a CA-signed Client Certificate and Key](#)
- [How to Create and Install a CRL for a Client Certificate](#)
- [Client Certificate SSL/TLS Best Practices](#)

## Authentication Technology Overview

Calendar 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.

By default, only passwords in the clear are enabled for Calendar 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. SSLv3 is also a legacy version which is still used, though it is less secure than TLS.

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.

## 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. Calendar 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 that are no longer valid, signed by the relevant CA.

## Certificate and Key Storage Overview

Certificates and keys must be stored somewhere. By default, Calendar 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 that 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 Calendar Server 6.3:

- 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 1.2 (RFC 5246).
- SSLv3 is a less secure and less extensible predecessor to TLS 1.0 and later.
- TLS Renegotiation Indication Extension (RFC 5746) fixes a security bug present in all versions of SSL and TLS prior to TLS 1.3. For patches, see [Sun Alert 273350](#)
- HTTPS (RFC 2818) is a de-facto standard for use of SSL/TLS on the "https" port (443). The `cshttpd` daemon that runs on port 443 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.
- PKCS#11 is an industry standard that defines how a security library talks to software or hardware that can implement certificate and key storage as well as 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. Calendar 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. Calendar Server does not support this protocol.

## SSL/TLS Tools Available in Communications Suite Installer

### `SUNWt1su`

The `SUNWt1su` (Solaris) and `sun-nss` (Linux) packages installed by the Communications Suite Installer include utilities that can be used to manage keys and certificates for Calendar Server. On Solaris, the utilities are installed in `/usr/sfw/bin` by default. On Linux they are installed in `/opt/sun/private/bin` by default. Make sure to supply the option `-d /opt/sun/comms/calendar/SUNWics5/cal/config` (substituting the appropriate path for your installation) to use these tools with Calendar Server's configuration directory. If you run these utilities as root, you must manually "chown" the appropriate output files (`cert8.db`, `key3.db`, `secmod.db` or `cert9.db`, `key4.db`, `pkcs11.txt`) so that they are owned by the Calendar Server user.

## Utilities Used to Manage Certificates

- The `certutil` tool is used to manage certificates. When creating a certificate or certificate request, include `-Z SHA1 -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 as well as 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.

## Certificate and Key Storage

When Calendar Server is used with the default PKCS#11 module (`cert8.db`, `key3.db`, `secmod.db`), modifications to the certificate and key databases are not permitted while the server is running. Such changes can corrupt internal data structures and cause servers to crash. The `configlocal.ssldblegacy` set to "0" 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 only if the administrator modifies the environment as follows:

```
~/.bashrc: export NSS_DEFAULT_DB_TYPE="sql"  
~/.cshrc: setenv NSS_DEFAULT_DB_TYPE "sql"
```

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

1. `cd /opt/sun/comms/calendar/SUNWics5/cal/config`
2. Create file containing just the password for the key database with no

- ```
extra newline. Call this {{pwfile.certutil --merge -d sql:. -f pwfile
--source-dir . -@ pwfile
```
3. Create a directory for the old format databases. `mkdir old_certdb`
  4. `mv cert8.db key3.db secmod.db old_certdb`

Executing steps 3 and 4 prevent the `certutil` and `pk12util` utilities from using the old databases, if you happen 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, do the following:

For Solaris: `pkgparam -v SUNWtls SUNW_PRODVERS`

For Linux: `rpm -q sun-nss`

It is recommended that you upgrade to NSS 3.12.5 or later due to the SSL/TLS renegotiation vulnerability (See Sun Alert 273350).

## SSL/TLS Configuration

For client certificate authentication to work, a client certificate must be translated to a Calendar 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` file in the `config-templates` directory which can be copied to `certmap.conf`. `certmap.conf` has named sections ñ a "default" section is mandatory ñ so different rules can be applied to different client certificate issuer DN's. The following are the available options:

`DNComps`

- commented out - take the user's DN from the cert 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

### 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/calendar/SUNWics5/cal/config
% certutil -d sql:. -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.

Make sure that the certificate, key and `pkcs11.txt` files are owned by the calendar server user by performing the following:

```
% certutil -S -d sql:. -g 2048 -Z SHA1 -n CA-Cert -s "cn=CA Cert for
Calendar" -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 sql:. -g 2048 -Z SHA1 -n Server-Cert -s
"cn=cal.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 sql:. -g 2048 -Z SHA1 -m 1 -n client-cert -s
"e=user@example.com" -c CA-Cert -f pwfile -t u
```

## 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 sql:. -G -n CA-Cert << EOF
update=20100510200000Z
addcert 1 20100510200000Z
EOF
```

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

For the authentication to execute successfully the following is required:

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 a CA certificate trusted to sign client certs installed.
4. There must be a valid `certmap.conf` file with correct permissions.
5. 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.

## SSL/TLS Best Practices

The following are best practices when using SSL/TLS with Calendar Server:

- Make sure that the installed NSS version is 3.12.5 or later.
- Use a 2048-bit RSA certificate with a SHA1 signature.
- Enable use of SSL between end-user clients and the server whenever possible.
- Disable SSLv3 by setting `local.sslv3enable` to 0.

## Client Certificate SSL/TLS Best Practices

- 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 via cron or similar mechanism.

## How to Enable Certificate-based Authentication for the Calendar Server Client

Certificate-based Authentication is disabled by default. To enable the authentication functionality, perform the following steps:

1. `cd <install_root>/SUNWics5/cal/config/`
2. `cp ../lib/config-templates/certmap.conf .`
3. Change the ownership of the `certmap.conf` file to the calendar runtime user/group (`icsuser/icsgroup`)
4. Modify the `certmap.conf` file per your site's requirements.

## Calendar Server and SLL/TLS: Known Limitations

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.

In order 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 own `cron` or equivalent script to fetch and update this information.

## Appendix A. References

[Technical Documentation for new database format in NSS](#)

[NSS Shared DB Howto \(Primarily for Firefox/Thunderbird\)](#)

[Documentation for certutil](#)

Documentation for `crlutil`

Key Manager (add on for Firefox to provide GUI key management for NSS)

Sun Alert 273350

# To Configure Oracle Communications Calendar Server Client Authentication

## To Configure Oracle Communications Calendar Server Client Authentication



This page contains information for Calendar Server 7.0.4.15.0 and will not be updated in the future. For documentation beginning with Calendar Server 7.0.5.17.0, see the Oracle Technology Network site at:

<http://www.oracle.com/technetwork/documentation/oracle-communications-185806.html>

Starting with version 7 Update 2, Oracle Communications Calendar Server (also known as Calendar Server 7 and formerly known as Oracle Communications Calendar Server for CALDAV Clients) supports certificate-based authentication. In certificate-based authentication, clients request access to a protected resource, such as the Calendar Server. The server presents its certificate to the client, which the client verifies. If the verification succeeds, the client then sends its certificate to the server and the server verifies the client's credentials. As long as the client's credentials are verified, the server grants access to the protected resource.

Topics:

- [Overview of Setting Up Certificate Authentication](#)
- [To Set up Your Certificate Authority \(CA\)](#)

### Overview of Setting Up Certificate Authentication

Setting up certificate authentication for Calendar Server involves the following high-level steps:

1. Obtaining your certificates from a Certificate Authority (CA), or setting up your own CA for testing purposes
2. Enabling SSL and client authentication for a listener
3. Generating a certificate request and importing into GlassFish Server
4. Creating an SSL client
5. Configuring Calendar Server
6. Testing the certificate authentication
7. Installing the client certificate for use by Connector for Outlook

### To Set up Your Certificate Authority (CA)

**Note**

In a production environment, you would not set up your own Certificate Authority and generate certificates with it, as is described in this section. You would most likely purchase certificates from a commercial Certificate Authority, such as VeriSign. This section is purely for test purposes.

1. If you haven't done so already, download the OpenSSL toolkit from <http://www.openssl.org/>. You use the `openssl` command to perform a variety of cryptographic functions.
2. Create directories to hold your certificate authority (CA) keys, your server keys, and your client keys.

For example, you could use directories called `ssl/ca`, `ssl/server`, and `ssl/client`.

```
cd /var/tmp
mkdir -p ssl/ca ssl/client ssl/server
```

3. Use the `openssl` command to create a private key and certificate request for your own CA. For example:



```

openssl req -new -newkey rsa:2048 -nodes -out ssl/ca/ca.csr -keyout
ssl/ca/ca.key
Generating a 2048 bit RSA private key
.....
new private key to 'ssl/ca/ca.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]: US
State or Province Name (full name) [Some-State]: CA
Locality Name (eg, city) []: Santa Clara
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Siroe
Organizational Unit Name (eg, section) []: Corporate
Common Name (eg, YOUR name) []: Sam Smith
Email Address []:sam.smith@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
# Create your CA's self-signed certificate.
The default setting is one year. You might want to increase this
setting by increasing the number of days.openssl x509 -trustout
-signkey ssl/ca/ca.key -days 365 -req -in ssl/ca/ca.csr -out
ssl/ca/ca.pem
Signature ok
subject=/C=US/ST=CA/L=Santa Clara/O=Siroe/OU=Corporate/CN=Sam
Smith/emailAddress=sam.smith@example.com
Getting Private key

```

4. Copy the `ca.pem` file to `ca.crt` and edit the file so that the strings "TRUSTED CERTIFICATE" read "CERTIFICATE."  
You can then import your CA certificate into your trusted root certificate's store.
5. Create a file to hold your CA's serial numbers. This file starts with the number 2.  
For example:

```
echo "02" > ssl/ca/ca.srl
```

#### To Enable SSL and Client Authentication for a Listener

Refer to [How to Enable SSL and Client Authentication for a Listener in Oracle GlassFish Server](#) for this step.

#### To Generate a Certificate Request and Import into GlassFish Server

1. Use the `certutil` command to generate a certificate request.

For example:

```
cd /opt/SUNWappserver/lib
./certutil -R -s "CN=host1.example.com, OU=Corporate, O=Siroe,
L=Santa Clara, ST=CA, C=US" -o /var/tmp/ssl/server/host1.csr -d
/opt/SUNWappserver/domains/domain1/config -a
# Use the previously created CA to sign this certificate request.
For example:cd /var/tmp
openssl x509 -CA ssl/ca/ca.pem -CAkey ssl/ca/ca.key -CAserial
ssl/ca/ca.srl -req -in ssl/server/host1.csr -out
ssl/server/host1.crt -days 365
```

2. Import your signed server certificate into your server NSS keystore.

For example:

```
cd /opt/SUNWappserver/lib
./certutil -A -n "TestSSLCert" -t "p,p,p" -d
/opt/SUNWappserver/domains/domain1/config -i
/var/tmp/ssl/server/host1.crt
```

3. Import your CA certificate into your server NSS keystore.

For example:

```
./certutil -A -n "TestCACert" -t "T,c,c" -d
/opt/SUNWappserver/domains/domain1/config -i /var/tmp/ssl/ca/ca.crt
```

The next step is necessary if you want to use SSL client authentication.

4. Modify HTTPS listener, typically `http-listener-2`, to use your `TestSSLCert`, that is, changing it from `slas` to your own `TestSSLCert`.
5. Add the following line to the `http-listener-2` property list if you want to use the fallback feature if client authentication fails. If you don't want to fallback, change `want` to `need`.

```
<property name="com.sun.grizzly.ssl.auth" value="want"/>
```

### To Create an SSL Client

1. Use the `openssl` command to create a client certificate request.  
The specified email address must be for an existing LDAP user in the Directory Server. For example:

```

cd /var/tmp
openssl req -new -newkey rsa:2048 -nodes -out
ssl/client/samsmith.req -keyout ssl/client/samsmith.key
Generating a 2048 bit RSA private key
.....
new private key to 'ssl/client/samsmith.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]: US
State or Province Name (full name) [Some-State]: CA
Locality Name (eg, city) []: Santa Clara
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Siroe
Organizational Unit Name (eg, section) []: Corporate
Common Name (eg, YOUR name) []:Sam Smith
Email Address []:sam.smith@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

```

2. Use the `openssl` command to have the CA sign the client certificate.

```

openssl x509 -CA ssl/ca/ca.pem -CAkey ssl/ca/ca.key -CAserial
ssl/ca/ca.srl -req -in ssl/client/samsmith.req -out
ssl/client/samsmith.pem -days 365
Signature ok
subject=/C=US/ST=CA/L=Santa Clara/O=Siroe/OU=Corporate/CN=Sam
Smith/emailAddress=sam.smith@example.com
Getting CA Private Key
# Use the {{openssl}} command to generate a PKCS12 file containing
your client certificate.
For example:openssl pkcs12 -export -clcerts -in
ssl/client/samsmith.pem -inkey ssl/client/samsmith.key -out
ssl/client/samsmith.p12 -name "sam_smith_cert"
Enter Export Password:
Verifying - Enter Export Password:

```

3. Import the PKCS12 file into your web browser to use as your client certificate.
4. Repeat Steps 1 through 4 as often as required.

### To Configure Calendar Server

1. Decide how you want to configure certificate authentication.  
The Calendar Server configuration parameters to enable Certificate Authentication are:

- `davcore.auth.cert.enable` (default is `false`)
  - `davcore.auth.cert.fallback` (default is `true`)
2. Use the `davadmin` command to configure the parameters that enable certificate authentication. For example:

```
davadmin config modify -u admin -o davcore.auth.cert.enable -v true
```

If you want to use certificate authentication exclusively, disable the fallback option by setting the `davcore.auth.cert.fallback` parameter to `false`. Otherwise, you can use the standard login mechanism if certificate authentication fails. The fallback option has meaning only if certificate authentication is enabled.

3. Set up the `certmap.conf` file to map the subject in the client certificate to an LDAP user. For example:

```
certmap=default,testca
default.IssuerDN=default

testca.IssuerDN=CN=TestCA,OU=Corporate,O=Siroe,L=Santa
Clara,ST=CA,C=US
testca.DNComps=
testca.FilterComps=emailaddress=mail
testca.VerifyCert=off
```

For more information, see [How to Configure the Convergence Certificate Mapper](#).

4. Restart Glassfish Server.

```
/opt/SUNWappserver/bin/asadmin stop-appserv
/opt/SUNWappserver/bin/asadmin start-appserv
```

### To Test Certificate Authentication

1. In a browser in which the the client certificate installed has been installed, connect to the Calendar Server SSL port. For example:

```
https://host1.example.com:8181/davserver/browse/home/samsmith/
```

If certificate authentication is working properly, a pop-up dialog prompts you to select the certificate to us.

2. Click OK. You should see the Calendar Server browser page if certificate authentication is working properly.
3. Check the `errors.0` log file for a message similar to the following for certificate authentication taking place.

```
INFO      [2011-06-10T10:44:15.623-0700]
<...X509CertificateLoginModule.login> Performing certificate
authentication with these details:
...
```

### To Install the Client Certificate in Connector for Microsoft Outlook

To use certificate authentication to Calendar Server 7 Update 2 with Connector for Outlook, import the client certificate by using Internet Explorer. This makes the certificate available to Connector for Outlook. See [Certificate-based Authentication for Connector for Microsoft Outlook](#) for more information.

# Enabling the NSS Shared Database Feature in Oracle Communications Calendar Server

## How to Enable the NSS Shared Database Feature in Oracle Communications Calendar Server



### NOTE:

Apply the 3.12.6 version of the NSS shared component prior to enabling the new feature. See Special Instructions in the patch 6948652 README for more details.

The shared database feature is disabled by default. To enable the feature, perform the following steps:

1. Set `local.sslldblegacy` in the `ics.conf` file to "0". This turns off the legacy database mode.
2. Navigate to the directory where the `certdb` files are located: `cd <install_root>/SUNWics5/cal/config`
3. Create a file named `pwfile` containing the password for the key database. Do not include an extra newline: `certutil --merge -d sql:. -f pwfile --source-dir . -@ pwfile`
4. Create the new directory: `mkdir old_certdb`
5. `mv cert8.db key3.db secmod.db old_certdb`
6. Change the ownership of the newly created database. (`cert9.db`, `key4.db` and `pkcs11.txt`) to calendar runtime user/group (`icsuser/icsgroup`).