Oracle® Communications Calendar Server

Security Guide

Release 7.0.5

E54936-01

February 2015

ORACLE®

Oracle Communications Calendar Server Security Guide, Release 7.0.5

E54936-01

Copyright © 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

4 Managing Domain Access Controls

5 Creating, Exporting, and Importing SSL Certificates

6 Configuring Client Authentication

7 Configuring GlassFish Server to Use a CA-Signed Certificate

Preface

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

Audience

This document is intended for system administrators or software technicians who work with Calendar Server.

Related Documents

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

- *Calendar Server Concepts*: Provides an overview of Calendar Server.
- *Calendar Server Installation and Configuration Guide*: Provides instructions for installing and configuring Calendar Server.
- *Calendar Server Release Notes*: Describes the new features, fixes, known issues, troubleshooting tips, and required third-party products and licensing.
- *Calendar Server Administrator's Guide*: Describes the tasks and concepts for administering Calendar Server.
- *Calendar Server WCAP Developer's Guide*: Describes how to use the Web Calendar Access Protocol 7.0 (WCAPbis) with Calendar Server.

Documentation Accessibility

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

Access to Oracle Support

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

1

Calendar Server Security Overview

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

Basic Security Considerations

The following principles are fundamental to using any application securely:

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

   `http://www.oracle.com/technetwork/topics/security/alerts-086861.html`

Understanding the Calendar Server Environment

When planning your Calendar Server implementation, consider the following:

- Which resources must be protected?

  For example:

  - Calendar Server front end
  - Calendar Server back end (MySQL Server or Oracle Database)
  - Document store (can be local, remote, or dbdocstore)

  - Dependent resources, such as GlassFish Server and Directory Server

- From whom am I protecting the resources?

  In general, resources must be protected from everyone on the Internet. But should the Calendar Server deployment be protected from employees on the intranet in your enterprise? Should your employees have access to all resources within the GlassFish Server environment? Should the system administrators have access to all resources? Should the system administrators be able to access all data? You might consider giving access to highly confidential data or strategic resources to only a few well trusted system administrators. On the other hand, perhaps it would be best to allow no system administrators access to the data or resources.

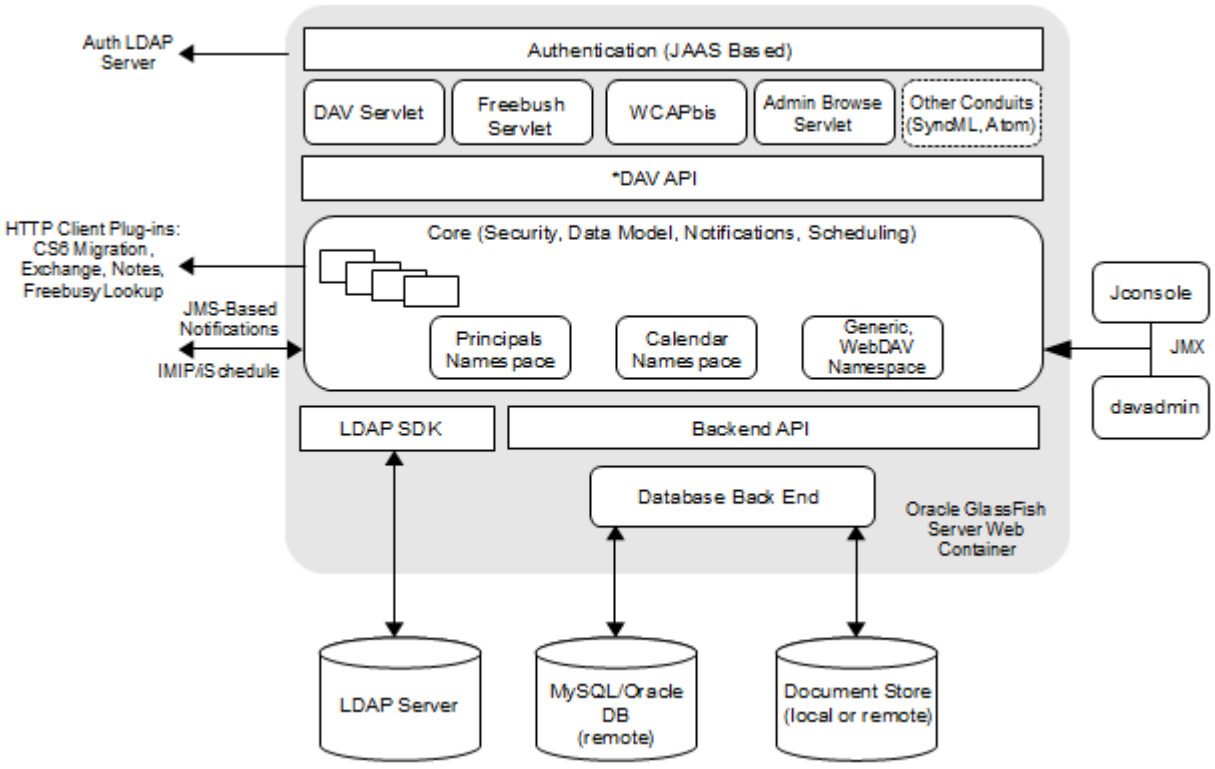
- What happens if protections on strategic resources fail?

  In some cases, a fault in your security scheme is easily detected and considered nothing more than an inconvenience. In other cases, a fault might cause great damage to companies or individual clients that use Calendar Server. Understanding the security ramifications of each resource help you protect it properly.

Overview of Calendar Server Security

Figure 1–1 shows all the various components that can comprise Calendar Server, including the components to which it connects. Each installed or integrated component requires special steps and configurations to ensure system security.

Figure 1–1 Calendar Server Components



Recommended Deployment Topologies

You can deploy Calendar Server on a single host or on multiple hosts, splitting up the components into multiple front-end hosts and multiple back-end hosts. You can also install the document stores onto separate hosts. For more information, see the topic on planning your installation in *Calendar Server Installation and Configuration Guide*.

The general architectural recommendation is to use the well-known and generally accepted Internet-Firewall-DMZ-Firewall-Intranet architecture. For more information on addressing network infrastructure concerns, see the *Determining Your Communications Suite Network Infrastructure Needs* documentation at:

https://wikis.oracle.com/display/CommSuite/Determining+Your+Communications+Suite+Network+Infrastructure+Needs

Operating System Security

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

Firewall Port Configuration

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

- MySQL Database port (default 3306)
- Oracle Database server port (default 1521)
- Calendar Server back-end remote document store port (default 8008)
- GlassFish Server administration server port (default 4848)
- Calendar Server access port (default 443)
- Notification mail server port (default 25)

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

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

Database Security

For more information about securing Oracle Database, see *Oracle Database Security Guide* and *Oracle Database Advanced Security Administrator's Guide*, at:

http://www.oracle.com/pls/db112/portal.portal_db?selected=25&frame=

For more information about securing MySQL Server, see *Security in MySQL*, at:

http://dev.mysql.com/doc/mysql-security-excerpt/5.6/en/index.html

Secure Communications

Secure connections between applications connected over the World Wide Web can be obtained by using protocols such as Secure Socket Layer (SSL) or Transport Layer Security (TLS). SSL is often used to refer to either of these protocols or a combination of the two (SSL/TLS). Due to a security problem with SSLv3, Calendar Server

recommends the use of only TLS. However, throughout this guide, secure communications may be referred to by the generic term SSL.

In a Calendar Server deployment, you can enable the use of TLS between the following components:

- GlassFish Server and client connections
- Calendar Server and Directory Server
- GlassFish Server and the JMX port used by Calendar Server administration utility
- Calendar Server and the back-end database
- Calendar Server and the document store

See "Implementing Calendar Server Security" for more information.

LDAP Security

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

2

Performing a Secure Calendar Server Installation

This chapter presents planning information for your Oracle Communications Calendar Server system and describes recommended deployment topologies that enhance security.

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

Installing Infrastructure Components Securely

Calendar Server is deployed within GlassFish Server. When installing and configuring GlassFish Server:

- Configure HTTPS and disable HTTP
- Configure the JMX Port for GlassFish Server to use SSL
- Configure GlassFish Server to prevent Denial of Service (DoS) attacks

To configure and administer GlassFish Server security, see *Oracle GlassFish Server Security Guide*.

Calendar Server can use either MySQL Server or Oracle Database as the database for storing contact information. For information on how to install and configure either MySQL Server or Oracle Database, see *Calendar Server Installation and Configuration Guide*.

Credentials Needed to Install Calendar Server Components

The installation prompts for authentication credentials for the following:

- Database user
- GlassFish Server administrator
- Directory Server manager (bind DN and password)
- Calendar Server administrator

Post-Installation Configuration

After installation, configuring Calendar Server for a secure deployment involves a number of steps:

1. Make sure that HTTPS is configured correctly on the front-end GlassFish Server host:
   - Use a CA signed certificate
   - Set SSL port to default port of 443 to ease client configurations
   - Change the `fulluriprefix` configuration option
2. Disable HTTP on the front-end GlassFish Server host
3. Configure JMX port for GlassFish Server to use SSL
4. Enable LDAP SSL, if not previously done
5. Enable secure notification mail submission
6. Configure SSL on back ends
   - Set up secure communication to the Calendar Server database
   - Set up secure communications to the remote document store
7. Add LDAP access control for Calendar Server

See "Implementing Calendar Server Security" for more information.

3

Implementing Calendar Server Security

This chapter explains the security features of Oracle Communications Calendar Server and the following tasks:

- Configuring HTTPS on Front-End GlassFish Server Hosts
- Disabling SSLv3 on Front-End GlassFish Server Hosts
- Disabling HTTP on Front-End GlassFish Server Hosts
- Enabling LDAP SSL in Calendar Server
- Enabling Secure Notification Mail Submission
- Configuring and Using Authentication
- Configuring Calendar Server Proxy Authentication
- Configuring and Using Access Control
- Adding LDAP Access Control for Calendar Server Features
- Configuring SSL for the Database Back End and Document Store
- Making Calendar Server and GlassFish Server Secure
- Detecting Possible Security Issues

About System Security in Calendar Server

Security requirements arise from the need to protect data: first, from accidental loss and corruption, and second, from deliberate unauthorized attempts to access or alter that data. Secondary concerns include protecting against undue delays in accessing or using data, or even against interference to the point of denial of service. The global costs of such security breaches run up to billions of dollars annually, and the cost to individual companies can be severe, sometimes catastrophic.

The critical security features that provide these protections are:

- Authentication
- Access Control
- Secure Communications

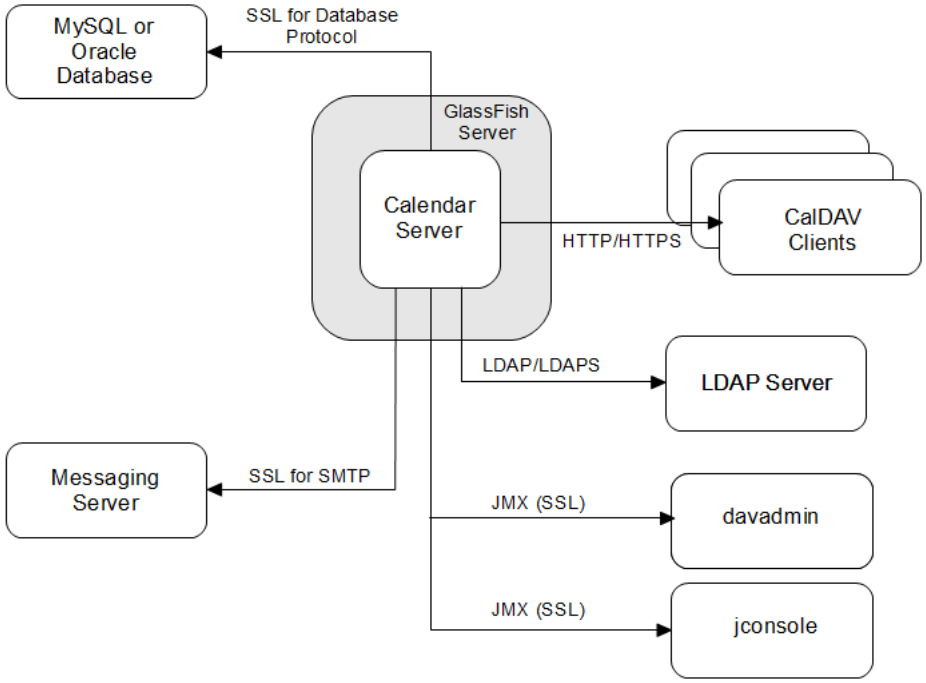
Authentication is the way in which an entity (a user, an application, or a component) determines that another entity is who it claims to be. An entity uses security credentials to authenticate itself. The credentials might be a user name and password, a digital certificate, or something else. Usually, servers or applications require clients to authenticate themselves. Additionally, clients might require servers to authenticate themselves. When authentication is bidirectional, it is called *mutual authentication*.

Access Control, also known as authorization, is the means by which users are granted permission to access data or perform operations. After a user is authenticated, the user's level of authorization determines what operations the user can perform.

Calendar Server supports LDAP authentication and the use of Access Control Instructions (ACIs) to grant end users permission to search the LDAP directory for other users and resources.

Figure 3–1 shows the protocols and communication flows used by Calendar Server that can be secured. The HTTPS, LDAPS, JMX, and SMTPS protocols must be secured by using SSL. SQL/JDBC connections between GlassFish Server and the database are also secured by using SSL.

Figure 3–1 Calendar Server Protocols Flow



In the preceding figure, HTTPS and SSL provide encryption of data between the server and the respective components. For information on securing SMTP for notifications, see the topic on configuration parameters in *Calendar Server System Administrator's Guide* for information on the **notification.dav.smtp.*** parameters. See "Enabling LDAP SSL in Calendar Server" for information on LDAPS. See "Configuring SSL for MySQL Server" or "Configuring SSL for the Oracle Database" for information on securing connections to the back-end database. For more information on designing a secure deployment, see the "Designing for Security" documentation at:

`https://wikis.oracle.com/display/CommSuite/Designing+for+Security`

Note: SSL is often used to refer to either SSL or TLS protocols or a combination of the two (SSL/TLS). Throughout this chapter, secure communications may be referred to by the generic term SSL.

Configuring HTTPS on Front-End GlassFish Server Hosts

To configure HTTPS on the front-end GlassFish Server host, perform the following tasks:

- Installing an Official Certificate
- Setting SSL Default Port to 443
- Changing the URL Prefix When Reconfiguring for SSL

Installing an Official Certificate

The default GlassFish Server installation comes with a self-signed certificate, which is incompatible with production usage. To install an official certificate, see "Configuring GlassFish Server to Use a CA-Signed Certificate."

Setting SSL Default Port to 443

Most clients assume that the GlassFish Server is running on the default SSL port number (443). If you did not set the GlassFish Server default SSL port to 443 during installation, use this task to do so.

To set the default SSL port to 443:

1. List all the HTTP listeners. For example:

```
./asadmin list-http-listeners
http-listener-1
http-listener-2
admin-listener
Command list-http-listeners executed successfully.
```

2. Note the SSL listener (which will have an attribute of **security-enabled=true**).

```
./asadmin get
server.network-config.protocols.protocol.http-listener-1.security-enabled
server.network-config.protocols.protocol.http-listener-1.security-enabled=false
Command get executed successfully.

./asadmin get
server.network-config.protocols.protocol.http-listener-2.security-enabled
server.network-config.protocols.protocol.http-listener-2.security-enabled=true
Command get executed successfully.

./asadmin get
server.network-config.protocols.protocol.admin-listener.security-enabled
server.network-config.protocols.protocol.admin-listener.security-enabled=false
Command get executed successfully.

./asadmin get
server.network-config.network-listeners.network-listener.http-listener-2.port
server.network-config.network-listeners.network-listener.http-listener-2.port=8
181
Command get executed successfully.
```

3. Set the port number to the correct value. For example:

```
./asadmin set
server.network-config.network-listeners.network-listener.http-listener-2.port=4
43
server.network-config.network-listeners.network-listener.http-listener-2.port=4
```

```
43
Command set executed successfully.
```

This change does not require you to restart GlassFish Server.

Changing the URL Prefix When Reconfiguring for SSL

The default installation of GlassFish Server enables both HTTP and HTTPS (using a self-signed certificate) on the server instance. During the Calendar Server configuration, you can choose to specify only the HTTP port, which sets the **davcore.uriinfo.fulluriprefix** parameter to **http://***host*:*http_port*. This results in all URLs constructed by Calendar Server, for example those pointing to attachments, to have the **http://** URL. When you reconfigure to use SSL, the host name part of this prefix should match the host name associated with the certificate.

To change the **fulluriprefix**:

Run the **davadmin config** command to set the **davcore.uriinfo.fulluriprefix** parameter to the desired host name and port. For example:

```
./davadmin config -u admin -o davcore.uriinfo.fulluriprefix -v
https://www.example.com:8181
```

Disabling SSLv3 on Front-End GlassFish Server Hosts

Identify the http-listener for the publicly accessible port that has SSL/TLS enabled (**security-enabled=true**) on which requests for Calendar Server are received. Ensure that SSLv3 is disabled for this listener by setting the option ssl3-enabled to **false**.

1. Identify the HTTP listeners that have SSL/TLS enabled (**security-enabled=true**) and verify whether SSLv3 is enabled on that listener (**ssl3-enabled=true**).

   ```
   ./asadmin get configs.config.server-config.network-config.protocols.protocol.*
   | grep http-listener.*security-enabled=true
   configs.config.server-config.network-config.protocols.protocol.http-listener-2.
   security-enabled=true

   ./asadmin get
   configs.config.server-config.network-config.protocols.protocol.http-listener-2.
   ssl.ssl3-enabled
   configs.config.server-config.network-config.protocols.protocol.http-listener-2.
   ssl.ssl3-enabled=true
   Command get executed successfully.
   ```

2. Disable those HTTP listeners.

   ```
   ./asadmin set
   configs.config.server-config.network-config.protocols.protocol.http-listener-2.
   ssl.ssl3-enabled=false
   configs.config.server-config.network-config.protocols.protocol.http-listener-2.
   ssl.ssl3-enabled=false
   Command set executed successfully.
   ```

3. Restart GlassFish Server.

Disabling HTTP on Front-End GlassFish Server Hosts

Disable non-SSL HTTP access to prevent any unsecured communications with Calendar Server.

1. List all the HTTP listeners and note the ones that do not have security enabled. For example:

```
./asadmin get
configs.config.server-config.network-config.network-listeners.network-listener.
http-listener-1.enabled
configs.config.server-config.network-config.network-listeners.network-listener.
http-listener-1.enabled=true
Command get executed successfully.
```

2. Disable those HTTP listeners. For example:

```
./asadmin set
configs.config.server-config.network-config.network-listeners.network-listener.
http-listener-1.enabled=false
configs.config.server-config.network-config.network-listeners.network-listener.
http-listener-1.enabled=false
Command set executed successfully.
```

   This change does not require you to restart GlassFish Server.

Enabling LDAP SSL in Calendar Server

If you specified to use an LDAPS URL during the Calendar Server initial configuration, the changes described in this section were already performed by the **init-config** script.

You must have already enabled the back-end Directory Server for SSL, either with a CA-signed certificate or self-signed certificate. Also, copy the **cert8.db** and **key3.db** files to the *CalendarServer_home***/lib/** directory.

To configure Calendar Server to communicate with Directory Server over SSL:

1. Create a text file, for example, **usessl.txt**, with the following content:

```
base.ldapinfo.authldap.ldapport=port_number
base.ldapinfo.authldap.ldapusessl=true
base.ldapinfo.ugldap.ldapport=port_number
base.ldapinfo.ugldap.ldapusessl=true
```

   Change the *port_number* values to the LDAP SSL port value in your deployment.

2. Run the **davadmin config** command to make the configuration change. For example:

```
./davadmin config -u admin -f usessl.txt
```

   Warning messages appear to restart GlassFish Server.

3. Restart GlassFish Server.

Enabling Secure Notification Mail Submission

The Calendar Server notification mechanism relies on a messaging (email) server to deliver email notifications. By default, message submission is unsecured. You can secure this communication by using TLS/SSL transport.

To enable secure notification mail submission for Calendar Server:

1. Install either a CA-signed SSL certificate or self-signed certificate for your messaging server.

2. When installing a self-signed certificate, follow these guidelines:
   a. Import the certificate into the Java **trustStore** file by using the Java **keytool** command.
   b. Define the **javax.net.ssl.trustStore** and **javax.net.ssl.trustStorePassword** variables for the GlassFish Server JVM.
3. To use TLS, set the **notification.dav.smtpstarttls** configuration parameter to **`true`**.

   ```
   ./davadmin config modify -o notification.dav.smtpstarttls -v true
   ```

4. To use SSL, set the **notification.dav.smtpusessl** configuration parameter to **`true`** and set the **notification.dav.smtpport** configuration parameter to the SMTP SSL port, typically 465.

   ```
   ./davadmin config modify -o notification.dav.smtpusessl -v true
   ./davadmin config modify -o notification.dav.smtpport -v 465
   ```

5. To further enhance security, employ SMTP authentication:
   a. Set the **notification.dav.smtpauth** configuration parameter to **true**.

      ```
      ./davadmin config modify -o notification.dav.smtpauth -v true
      ```

   b. Specify the user and password of a valid mail user in your deployment by using the **notification.dav.smtpuser** and **notification.dav.smtppassword** configuration parameters.

      ```
      ./davadmin config modify -o notification.dav.smtpuser -v smtp_user
      ./davadmin config modify -o notification.dav.smtppassword -v password
      ```

Configuring and Using Authentication

For information on Calendar Server and LDAP authentication, see the topic on provisioning users in *Calendar Server System Administrator's Guide*.

Configuring Calendar Server Proxy Authentication

You can configure Calendar Server for proxy authentication to enable a calendar administrator to log in to Calendar Server on behalf of a calendar user. For a CalDAV client to do so, when providing the user name and credentials for HTTP basic authentication, give the user name *admin*;*user* instead of just *user*. For the password, use the administrator password. For WCAP clients, the login API defines a proxy authentication parameter. See the documentation on the **login.wcap** command in *Calendar Server WCAP Developer's Guide* for more information.

Configuring and Using Access Control

For information on configuring access control, see the topic on administering access in *Calendar Server System Administrator's Guide*.

Adding LDAP Access Control for Calendar Server Features

This section provides sample Access Control Instructions (ACIs) that show the attributes that Calendar Server needs for granting end users permission to search the LDAP directory for other users and resources. Tailor these samples to your individual site's security needs. Add an ACI to the user/group suffix in Directory Server by using the **ldapmodify** command. For more information on creating ACIs, see the ACI topic

in *Oracle Directory Server Enterprise Edition Administration Guide 11g Release 1 (11.1.1.5.0)* at:

http://docs.oracle.com/cd/E20295_01/html/821-1220/bcanc.html

Sample ACI configurations:

- The following sample ACI enables users to search for all other users and resources in the same domain for all domains hosted in the Directory Server, assuming a suffix of `o=isp`.

```
dn: o=isp
changetype: modify
add: aci
aci: (target="ldap:///($dn),o=isp")
 (targetattr!="userPassword")
 (targetfilter=(|(objectClass=icscalendaruser)(objectclass=icscalendarresource)
))
 (version 3.0; acl "CS User search access to all users and resources in own
domain - product=davserver,class=install,num=3,version=1";
 allow (read,search)
 userdn="ldap:///[$dn],o=isp??sub?(objectclass=icscalendaruser)";)
```

- To control domain access at a finer level, add individual ACIs instead of using the **$dn** macro. For example, to allow search for only one domain, set the following ACI on the domain organization entry.

```
dn: domainA.orgdn
changetype: modify
add: aci
aci: (targetattr!="userPassword")
 (targetfilter=(|(objectClass=icscalendaruser)(objectclass=icscalendarresource)
))
 (version 3.0; acl "CS User search access to all users and resources in domain
A - product=davserver,class=install,num=3,version=1";
 allow (read,search)
 userdn="ldap:///domainA.orgdn??sub?(objectclass=icscalendaruser)";)
```

  Replace *domainA.orgdn* with your organization DN.

When adding ACIs to enable users to search other domains, that is, cross-domain searches, keep the following in mind:

- For domain A users to be able to search for users in domain B, you would add an ACI on domain B's node to allow the search from users in domain A. For example, to enable users in **example.com** to search users in **varrius.com**, add the following ACI to the domain entry for **varrius.com domain o=varrius.com,o=isp** by using the **ldapmodify** command:

```
dn: o=varrius.com,o=isp
changetype: modify
add: aci
aci: (targetattr!="userPassword")
 (targetfilter=(|(objectClass=icscalendaruser)(objectclass=icscalendarresource)
))
 (version 3.0; acl "example.com users access in varrius.com -
product=davserver,class=install,num=3,version=1";
 allow (read,search)
 userdn="ldap:///o=example.com,o=isp??sub?(objectclass=icscalendaruser)";)
```

- You might also need to set Domain Access Control Lists (ACLs) to control operations that span multiple domains. For more information, see the topic on managing domain access in *Calendar Server System Administrator's Guide*.

Configuring SSL for the Database Back End and Document Store

This section describes how to configure the Secure Sockets Layer (SSL) protocol between the Calendar Server front ends and back ends, including the database and the remote document store.

Topics:

Configuring SSL for MySQL Server

You can enhance your security by configuring SSL communication between the Calendar Server front ends and back ends. To do so, first enable the back-end database servers for SSL by setting up the required **trustStore** files. Then configure the Calendar Server front ends to connect over SSL by making use of the stored certificates.

You can also configure SSL communication between the Calendar Server front ends and the remote document stores. For more information, see "Configuring SSL for the Remote Document Store."

To configure SSL for MySQL Server:

1. Create your own Certificate Authority and use it to sign a server certificate for the MySQL server instance and a client certificate for use with the MySQL Connector/J.

   For example:

```
shell> cd /etc/mysql
# Create CA certificate
shell> openssl genrsa 2048 > ca-key.pem
shell> openssl req -new -x509 -nodes -days 3650 \
-key ca-key.pem -out ca-cert.pem
# Create server certificate, remove passphrase, and sign it
# server-cert.pem = public key, server-key.pem = private key
shell> openssl req -newkey rsa:2048 -days 3650 \
-nodes -keyout server-key.pem -out server-req.pem
shell> openssl rsa -in server-key.pem -out server-key.pem
shell> openssl x509 -req -in server-req.pem -days 3650 \
-CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out server-cert.pem
# Create client certificate, remove passphrase, and sign it
# client-cert.pem = public key, client-key.pem = private key
shell> openssl req -newkey rsa:2048 -days 3650 \
-nodes -keyout client-key.pem -out client-req.pem
shell> openssl rsa -in client-key.pem -out client-key.pem
shell> openssl x509 -req -in client-req.pem -days 3650 \
-CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out client-cert.pem
# Verify both the self-signed client and server cert
shell> openssl verify -CAfile ca-cert.pem server-cert.pem client-cert.pem
server-cert.pem: OK
client-cert.pem: OK
```

2. Enable SSL in the MySQL instance.

   a. Stop the MySQL instance, if running.

   b. In the [mysqld] section of the **my.cnf file**, add the following configuration parameters:

```
ssl-ca=/etc/mysql/ca-cert.pem
ssl-cert=/etc/mysql/server-cert.pem
ssl-key=/etc/mysql/server-key.pem
```

3. To verify that the MySQL instance is now enabled for SSL, run the **mysql** command-line tool to check the global variable **have_ssl**.

   For example:

```
shell> /opt/mysql/mysql/bin/mysql -uroot -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 10079
Server version: 5.5.28-enterprise-commercial-advanced MySQL Enterprise Server -
Advanced Edition (Commercial)
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show variables like 'have_ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_ssl | YES |
+---------------+-------+
1 row in set (0.00 sec)
mysql> quit
Bye
shell>
```

4. If you want to run the **mysql** command-line tool with SSL, use the **--ssl-ca** option.

   For example:

```
shell> ./mysql --ssl-ca=/etc/mysql/ca-cert.pem -uroot -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 10089
Server version: 5.5.28-enterprise-commercial-advanced MySQL Enterprise Server -
Advanced Edition (Commercial)
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> \s
--------------
./mysql Ver 14.14 Distrib 5.5.28, for solaris10 (sparc) using EditLine wrapper
Connection id: 10089
Current database:
Current user: root@localhost
SSL: Cipher in use is DHE-RSA-AES256-SHA
Current pager: stdout
Using outfile: ''
Using delimiter: ;
```

```
Server version: 5.5.28-enterprise-commercial-advanced MySQL Enterprise Server -
Advanced Edition (Commercial)
Protocol version: 10
Connection: Localhost via UNIX socket
Server characterset: utf8
Db characterset: utf8
Client characterset: latin1
Conn. characterset: latin1
UNIX socket: /tmp/mysql.sock
Uptime: 2 days 23 hours 55 min 2 sec
Threads: 25 Questions: 104093 Slow queries: 0 Opens: 58 Flush tables: 1 Open
tables: 51 Queries per second avg: 0.402
--------------
mysql> quit
Bye
shell>
```

The output from the **\s** command shows that SSL is in use with the cipher information.

5. For the MySQL Connector/J in GlassFish to communicate with the MySQL server in SSL, put your Certificate Authority and the client certificate into a JKS **trustStore** and **keystore** respectively, and use them in the JDBC connection pool setup.

   For example, while remaining in the **/etc/mysql** directory, perform the following commands:

   ```
   shell> keytool -importcert -file ca-cert.pem -keystore cacerts.jks -storepass
   secret -storetype JKS
   shell> keytool -importcert -file client-cert.pem -keystore keystore.jks
   -storepass secret -storetype JKS
   ```

6. Stop GlassFish Server.

7. Add the following parameters in the **domain.xml** file to the existing **caldavPool** and **ischedulePool** JDBC connection pool definition:

   ```
   <property name="useSSL" value="true"/>
   <property name="requireSSL" value="true"/>
   <property name="trustCertificateKeyStoreUrl"
   value="file:///etc/mysql/cacerts.jks"/>
   <property name="trustCertificateKeyStoreType" value="JKS"/>
   <property name="trustCertificateKeyStorePassword" value="secret"/>
   <property name="clientCertificateKeyStoreUrl"
   value="file:///etc/mysql/keystore.jks"/>
   <property name="clientCertificateKeyStoreType" value="JKS"/>
   <property name="clientCertificateKeyStorePassword" value="secret"/>
   ```

8. Optional: To log the SSL communication with GlassFish Server, add the following to the JVM options in the **domain.xml** before starting GlassFish Server.

   ```
   <jvm-options>-Djavax.net.debug=ssl</jvm-options>
   ```

   This setting causes messages such as the following to appear in the GlassFish Server **server.log** file:

   ```
   [#|2013-03-25T16:11:52.799-0700|INFO|sun-appserver2.1|javax.enterprise.system.s
   tream.out|_ThreadID=39;_ThreadName=Timer-8;|
   Timer-8, WRITE: TLSv1 Handshake, length = 163|#]

   [#|2013-03-25T16:11:52.846-0700|INFO|sun-appserver2.1|javax.enterprise.system.s
   ```

```
tream.out|_ThreadID=39;_ThreadName=Timer-8;|
Timer-8, READ: TLSv1 Handshake, length = 74|#]

[#|2013-03-25T16:11:52.846-0700|INFO|sun-appserver2.1|javax.enterprise.system.s
tream.out|_ThreadID=39;_ThreadName=Timer-8;|
*** ServerHello, TLSv1|#]
```

9. Start GlassFish Server.

Configuring SSL for the Oracle Database

You can enhance security by configuring SSL communication between the Calendar Server front and back ends. First, you enable the Oracle back-end database server for SSL by setting up the required Oracle wallet and Oracle Net Listener. Then, you configure the Calendar Server front ends to connect over SSL by setting the properties on the JDBC connection pool setting.

You can also configure SSL communication between the Calendar Server front ends and the remote document stores. For more information, see "Configuring SSL for the Remote Document Store."

For more information about configuring Oracle Database with SSL, see the SSL with Oracle JDBC Thin Driver documentation at:

http://www.oracle.com/technetwork/database/enterprise-edition/wp-oracle-jdbc-thin-ssl-130128.pdf

Installing the Database Server Certificate

You can install the Oracle Database server certificate in an Oracle wallet that is accessible by the Oracle Net listener:

1. Use the **orapki** tool to create the Oracle wallet and a server certificate signing request. For more information about creating wallets, see Managing Oracle Wallets with the orapki Utility in *Oracle Database Advanced Security Administrator's Guide 11g Release 1*. For specific commands, see {{orapki}} Usage Examples in *Oracle Database Advanced Security Administrator's Guide 11g Release 1*.
2. When the certificate is signed by a trusted certificate authority, add it to the wallet for the Oracle Net configuration. For more information about using orapki to add a certificate to a wallet, see "orapki Utility Syntax" in *Oracle Database Advanced Security Administrator's Guide 11g Release 1*.
3. If the certificate authority is not known to the GlassFish Server instance hosting the Calendar Server front end, import the certificate authority's certificate into the GlassFish Server JVM. Depending on the GlassFish Server version, this is located in one of the following:
   - In the JRE CA certificate store that is typically available at:

     javahome/jre/lib/security/cacerts

     where *javahome* is the location of the JDK used by GlassFish Server.
   - In the GlassFish Server instance **config /cacerts.jks** store:

     GlassFish_home/domains/domain1/config/cacerts.jks

Or, you can use a self-signed certificate:

1. Use the **orapki** tool to create the root and server certificates, and use the root certificate to sign the server certificate.

2. The Oracle wallet and the self-signed server certificate are used in the Oracle Net configuration.
3. On the GlassFish Server machine hosting the Calendar Server front end, import the root certificate into the GlassFish Server JVM as described in Step in the previous task.

Enabling the TCP/IP SSL Listener in Oracle Net Services

To enable the Oracle Net listener for SSL communication, you must modify three configuration files for Oracle Net Services as shown in the following examples.

1. Manually edit the **sqlnet.ora**, **tnsnames.ora** and **listener.ora** files in **$ORACLE_HOME/network/admin**. In the following examples, replace values for host name, wallet location, and port numbers to match your configuration:

   - **sqlnet.ora**

```
# sqlnet.ora Network Configuration File:
# Generated by Oracle configuration tools.
SQLNET.AUTHENTICATION_SERVICES= (BEQ, TCPS)
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER= (SHA1, MD5)
SSL_VERSION = 0
TRACE_LEVEL_CLIENT = SUPPORT
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
SSL_CLIENT_AUTHENTICATION = FALSE
TRACE_LEVEL_SERVER = SUPPORT
SQLNET.CRYPTO_SEED = 'kjlkwebflk090kj92hjkky9hsjkhdhhhwjk'
SQLNET.ENCRYPTION_TYPES_SERVER= (3DES168, AES256, RC4_256, AES192, AES128,
RC4_128, 3DES112)
WALLET_LOCATION =
(SOURCE =
(METHOD = FILE)
(METHOD_DATA =
(DIRECTORY = /local/oracle/wallet/server)
)
)
SSL_CIPHER_SUITES= (SSL_RSA_WITH_AES_256_CBC_SHA, SSL_RSA_WITH_AES_128_CBC_
SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_RSA_WITH_RC4_128_SHA, SSL_RSA_WITH_
RC4_128_MD5, SSL_RSA_WITH_DES_CBC_SHA)
ADR_BASE = /u01/app/oracle
```

   - **tnsnames.ora**

```
# tnsnames.ora Network Configuration File:
# Generated by Oracle configuration tools.
LISTENER_ORCL =
(ADDRESS = (PROTOCOL = TCP)(HOST = dbhost.example.com)(PORT = 1521))
(ADDRESS = (PROTOCOL = TCPS)(HOST = dbhost.example.com)(PORT = 2484))
ORCL =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCPS)(HOST = dbhost.example.com)(PORT = 2484))
(ADDRESS = (PROTOCOL = TCP)(HOST = dbhost.example.com)(PORT = 1521))
)
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = orcl.example.com)
)
)
```

   - **listener.ora**

```
# listener.ora Network Configuration File:
# Generated by Oracle configuration tools.
SSL_CLIENT_AUTHENTICATION = FALSE
WALLET_LOCATION =
(SOURCE =
(METHOD = FILE)
(METHOD_DATA =
(DIRECTORY = /local/oracle/wallet/server)
)
)
LISTENER =
(DESCRIPTION_LIST =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
)
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = dbhost.example.com)(PORT = 1521))
)
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCPS)(HOST = dbhost.example.com)(PORT = 2484))
)
)
ADR_BASE_LISTENER = /u01/app/oracle
TRACE_LEVEL_LISTENER = SUPPORT
```

2. Restart the listener using the **lsnrctl** command.

Or, if you prefer, you can use the Oracle Net Manager graphical user interface (GUI) tool to configure the Oracle Net Services values for profile, name service, and listener. Refer to the examples above when entering values in the GUI.

1. Run **netmgr** from the command line.
2. Expand Oracle Net Configuration, and select **Profile** under the local configuration icon.
3. Under the Oracle Advanced Security pull-down menu, click the **SSL** tab.
4. Type the path to the Oracle wallet for the server. You can also add various cipher suites for use in the SSL negotiation.
5. Click the **Encryption** tab. Choose an encryption method for your SSL configuration.
6. Click the **Integrity** tab. Choose a data integrity method for your SSL configuration.
7. Under Oracle Net Configuration, expand Service Naming and select the **local service** icon.
8. Under Address Configuration, click the **Address 1** tab, choose the TCP/IP with SSL protocol, and type the host name and port number.
9. To add the additional TCP/IP with SSL address to the listener, under Oracle Net Configuration, expand Listeners and select the **listener** icon.
10. Click the **Address 3** tab, choose the TCP/IP with SSL protocol, and type the host name and port number.
11. From the **File** menu, choose **Save Network Configuration**.

Completing Installation and Modifying JDBC Connection Pool Settings

1. Install and configure Oracle Communications Calendar Server to communicate with the Oracle database using the non-SSL port.

2. Modify the **caldavPool** and **iSchedulePool** JDBC connection pool settings to use the TCPS protocol to communicate in SSL.

   Change the URL property in the JDBC connection pool setting by editing the **HOST**, **PORT**, and **SERVICE_NAME** according to your deployment.

   For example:

   ```
   jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=dbhost.example.com
   )(PORT=2484))(CONNECT_DATA=(SERVICE_NAME=orcl.example.com)))
   ```

Configuring SSL for the Remote Document Store

You can use SSL to secure transmission of data between Calendar Server and the remote document store. Configuring SSL between Calendar Server and the document store consists of the following high-level steps:

1. Create a self-signed or Certificate Authority (CA) signed certificate.
2. Create or update the SSL **keyStore** file on the document store host.
3. Configure the document store to accept SSL connections.
4. Install the certificate on the Calendar Server host, if necessary.
5. Make Calendar Server configuration changes to support SSL for the document store.

Creating an SSL Certificate

To create a certificate, see "Creating a Self-Signed Certificate."

Creating or Updating the SSL Keystore on the Document Store Server Host

After creating either a self-signed or CA-signed certificate, you must import that certificate into a **KeyStore** file on the document store host. If you created the certificate in a **KeyStore** on the server host, you can use it "as is." If you created the certificate on some other host, or obtained a CA-signed certificate, you must import it into a **KeyStore** on the document store host as described in "Creating, Exporting, and Importing SSL Certificates."

Making SSL Configuration Changes on the Document Store

To update the document store configuration with the location of the **KeyStore** file, and to enable SSL, edit the **config/ashttpd.properties** file. This file was previously created by the **configure-as** script when you initially configured the remote document store. The full path to the **ashttpd.properties** file is displayed when the **configure-as script** finishes.

To update the **asshttpd.properties** file for SSL:

1. Open the **ashttpd.properties** file.
2. Set the **store.usessl** property to **true**.

   ```
   store.usessl=true
   ```

3. Set the **store.sslkeystorepath** property to the **KeyStore** file's path. For example:

   ```
   store.sslkeystorepath=/myconfig/mykeystore.jks
   ```

4. Run the **davadmin passfile** command to set the document store SSL passwords, if you have not done so previously.

```
./davadmin passfile modify -O
```

5. When prompted by the command, type the document store and **KeyStore** (certificate) passwords.
6. Stop then restart the document store for the changes to take effect.

```
./stop-as
./start-as
```

Installing a Certificate on the Calendar Server Host

If you are using a CA-signed certificate, you do not need to install the certificate on the Calendar Server host, as long as the instance of GlassFish Server you are using contains the root certificate of that CA.

If you are using a self-signed certificate, you must import the certificate into the **trustStore** file that is used by GlassFish Server on the Calendar Server host. For more information, see "Installing the Self-Signed Certificate on the Client."

Configuring Calendar Server to Use SSL for the Document Store

To configure Calendar Server to use SSL for the document store:

1. On the Calendar Server host, set the **store.document.usessl** configuration parameter to **true**.

```
./davadmin config modify -o store.document.usessl -v true
```

2. Restart GlassFish Server.

Making Calendar Server and GlassFish Server Secure

Topics in this section:

Preventing Denial of Service Attacks on GlassFish Server

Using GlassFish Server, you can prevent a Denial of Service (DoS) attack against the server by:

- Limiting the size of a POST request
- Specifying a request timeout value
- Creating a blacklist of host names and/or IP addresses

For more information, see the topic on DoS prevention in *Calendar Server System Administrator's Guide*.

Configuring JMX Port for GlassFish Server to Use SSL

GlassFish Server does not enable the JMX port with SSL by default. If you want to make the JMX communications secure, you need to enable security for GlassFish, either through the GlassFish Administration Console, or through the **asadmin** command.

The davadmin command uses the JMX protocol to connect to GlassFish Server. This section describes how to create secure communications between the **davadmin**

command and the Calendar Server host over SSL. To do so, you need to create a **trustStore** file for **davadmin**. If you are using SSL for communicating with GlassFish Server, it is mandatory to also configure JMX to use SSL

To create secure communications between **davadmin** and Calendar Server:

1. Export the server certificate that GlassFish Server is using for SSL.

   Depending on the GlassFish Server version, you might use the Java **keytool** command or the NSS **certutil** command to export the certificate.

   - Example **keytool** command:

     ```
     keytool -exportcert -keystore keystore.jks -storetype JKS -alias s1as -rfc
     -file /tmp/s1as.txt
     ```

   - Example **certutil** command:

     ```
     /usr/sfw/bin/certutil -L -d . -n s1as -a > /tmp/s1as.txt
     ```

     In these examples, the current directory is the GlassFish Server configuration directory and the certificate is named **s1as**.

2. Import the GlassFish Server certificate into a Java **keystore** for use by the **davadmin** command with the **-s** option.

   For example:

   ```
   keytool -importcert -alias s1as -file /tmp/s1as.txt -keystore
   /var/opt/sun/comms/davserver/config/davtruststore -storetype JKS
   ```

3. Modify the **/var/opt/sun/comms/davserver/config/davadmin.properties** file to reflect the new **trustStore** file created in the previous step.

   Add the following line:

   ```
   secure=/var/opt/sun/comms/davserver/config/davtruststore
   ```

   Alternately, you could specify the explicit path to the **trustStore** file in the **davadmin** command with the **-s** option.

   For example:

   ```
   ./davadmin db backup -k /tmp/backup_file -O -A docstore_host.example.com:8008
   -s /my_home/my_truststore -u mysql
   ```

4. In the GlassFish Server Administration Console, enable secure JMX.

   a. Navigate to **Configurations**.

   b. Navigate to **server-config**.

   c. Navigate to **Admin Service**.

   d. On the **JMX Connector** tab, select the **Enabled box for Security**.

Detecting Possible Security Issues

You can use the Calendar Server log files to look for security problems. This section lists a few security-related log messages.

Login errors resemble the following:

```
INFO [2014-01-08T11:20:44.529-0600] <...LDAPLoginModule.lookupUser> Error while
retrieving user info for user <user>: No results found
```

If you have the logging level set to FINEST, then you see messages resembling the following when a login error occurs:

```
FINEST [2014-01-08T11:36:56.304-0600] <...WCAPServlet.service> failed login or
session timeout
```

If a user is trying to bypass the data parsing, you see warnings such as the following:

```
FINE [2014-01-08T11:39:53.426-0600] <...RESTServlet.service> Got a non standard
condition: failed to parse - Error at line 4:Illegal property [BEGI]
```

An unusually high activity of requests (REQ) from the same IP address shows up as the following in the commands log file:

```
Sample request log entry....
[2014-01-07T03:39:05.454-0700] <3887> DavServlet [REQ] REPORT
/davserver/principals/jsmith IP_address server:port
```

4

Managing Domain Access Controls

This chapter describes how Domain Access Control Lists (ACLs) control calendar operations that span multiple domains. Oracle Communications Calendar Server combines domain ACLs with the calendar and scheduling ACLs to grant or deny levels of access to these operations. All operations within a single domain rely strictly on the calendar and scheduling ACLs.

Topics:

Domain ACLs Overview

In Calendar Server, domain ACLs act as a gateway from one domain to another. When a user in one domain attempts to access a calendar in another domain, Calendar Server first checks the target domain's ACL. If that ACL allows access to the target domain then Calendar Server checks the calendar ACL. The domain ACL itself can also restrict access to the target calendar. For example, if the domain ACL only allows "read" access but the calendar ACL allows "write" access, the request is limited to "read" only.

Calendar Server domain ACLs are in the same form as WCAP calendar and scheduling ACLs. They consist of one or more Access Control Entries (ACEs) separated by semi-colons. The ACE is made up of a "who" part and a "privilege level" part in the form of *who:privilege*. A "who" can be a domain name in the form of *@domain* or **@** to designate the privilege level for **all**. The privilege level consists of a single letter.

In general, a domain-level ACL and a user-level ACL behave the same, as they both perform implicit denies and give precedence to more specific ACLs. The only difference occurs when the value is blank. If the domain-level ACL is blank, the server is allowed to continue its check of the user-level ACL. If the user-level ACL is blank, access is explicitly denied.

Authenticated Access

The two LDAP attributes used with domain ACLs are **icsDomainNames** and **icsDomainAcl**. Each domain can have zero or more **icsDomainNames**, which indicate the other domains that are known to this domain. The **icsDomainAcl** attribute, if it exists, contains the ACL for the domain.

icsDomainNames Attribute Overview

Each **icsDomainNames** attribute consists of the name of a domain, such **example.com**. This attribute indicates what other domains a particular domain is aware of. It is used for search operations across domains.

The search operation gets a list of target domains from the **icsDomainNames** attribute of the requester's domain. The operation first searches the requester's own domain then the domains in the list. As each domain in the list of target domains is processed, the **icsDomainAcl** of each domain and each calendar's ACL is checked, as described in the next section.

For a search in each of the domains specified in **icsDomainNames** to succeed, the corresponding domain nodes should have the correct LDAP ACIs and **icsDomainAcl** setting allowing search from this domain. See the topic on adding LDAP access control in *Calendar Server Installation and Configuration Guide* for more information on setting LDAP ACIs for domains. See the next section for more information on **icsDomainAcl** settings.

icsDomainAcl Attribute Overview

If the target domain does not have an **icsDomainAcl** attribute, the operation continues and the following checks are done by using the calendar and scheduling ACL to see if the operation can be completed.

1. If the **icsDomainAcl** attribute exists, the ACL is checked for the requesting user's domain. If the ACL has an ACE that contains the requesting user's domain, the privileges given to that domain in the ACE are returned and used to restrict what is available from the calendar and scheduling privileges. If the ACL contains both the domain ACE and the @ ACE, the domain ACE is always used.
2. If the **icsDomainAcl** attribute exists and does not contain an ACE for the requesting user's domain but it does contain the @ ACE, the privilege level of the @ ACE is used.
3. If the **icsDomainAcl** attribute exists but does not contain an ACE for the requesting user's domain and also does not contain the @ ACE, access is denied for that user.

Table 4–1 provides more detail on **icsDomainAcl** conditions and results. The Domain Privileges Returned column describes the rights that are available at the domain level and how these are used to restrict the rights allowed at the calendar and scheduling level.

Table 4–1 icsDomainAcl Conditions and Results

1.	icsDomainAcl	Domain in ACL	Privilege Level Versus Requested Level	Domain Privileges Returned
U1	Null or empty	Not applicable	Not applicable	All
U2	Exists	Yes	Domain privilege equal to or higher	Based on level in domain ACE
U3	Exists	Yes	Domain privilege lower	None
U4	Exists	Yes plus @	Domain privilege lower and @ privilege is higher	None (Always use domain ACE)
U5	Exists	No and no @	Not applicable	None

Table 4–1 (Cont.) icsDomainAcl Conditions and Results

1.	icsDomainAcl	Domain in ACL	Privilege Level Versus Requested Level	Domain Privileges Returned
U6	Exists	No but @ exists	@ privilege equal to or higher	Based on level in @ ACE
U7	Exists	No but @ exists	@ privilege lower	None

Authenticated Access Examples

This section provides examples that show different levels of cross-domain calendar access.

Example 1

Table 4–2 shows an example that enables **userA** in domain **a.com** to be able to read **userB**'s calendar in **domain b.com**.

Table 4–2 Example 1 Cross-Domain Access

Domain a.com	Domain b.com
LDAP entry = **icsDomainNames: b.com**	No **icsDomainAcl** defined **userB**'s calendar has **acl=userA@a.com:r**

Example 2

Table 4–3 shows an example that enables **userA@a.com** to write to the calendar for **userB@b.com**.

Table 4–3 Example 2 Cross-Domain Access

Domain a.com	Domain b.com
LDAP entry = **icsDomainNames: b.com**	LDAP entry = **icsDomainAcl: @a.com:w** **userB**'s calendar has **acl=userA@a.com:w**

Example 3

Table 4–4 shows an example that blocks **userA@a.com** from writing to **userB@b.com**'s calendar.

Table 4–4 Example 3 Cross-Domain Access

Domain a.com	Domain b.com
LDAP entry = **icsDomainNames: b.com**	LDAP entry = **icsDomainAcl: @a.com:r** **userB**'s calendar has **acl=userA@a.com:w**

Anonymous Access Overview

If there is no **icsDomainAcl** attribute for the target domain, all rights are returned and further checks are then done by using the calendar and scheduling ACLs.

If the target domain has an **icsDomainAcl**, but that ACL does not contain the @ ACE, access is denied.

If the target domain has an **icsDomainAcl** and the ACL contains the @ ACE and the privilege level of that ACE is higher than the requested privilege, the ACE's privilege level is returned and processing continues with the calendar and scheduling ACLs.

Table 4–5 provides more detail on anonymous access conditions and results. The Domain Privileges Returned column describes the rights that are available at the domain level and how these are used to restrict the rights allowed at the calendar and scheduling level.

Table 4–5 Anonymous Conditions and Results

1.	icsDomainAcl	@ in icsDomainAcl	Privilege Level Versus Requested Level	Domain Privileges Returned
A1	Null or empty	Not applicable	Not applicable	All
A2	Exists	No	Not applicable	None
A3	Exists	Yes	@ privilege equal to or higher	Based on level in @ ACE
A4	Exists	Yes	@ privilege lower	None

5

Creating, Exporting, and Importing SSL Certificates

This chapter describes how to create, export, and import SSL certificates, including both self-signed and CA (certificate authority) certificates.

Topics:

Overview of Self-Signed and Certificate Authority Certificates

An SSL certificate is necessary for transmission of encrypted data between a client and a server. A self-signed certificate is one that you create for your server, in the server's KeyStore. You then export and import the exported certificate into the client's TrustStore. A certificate authority (CA) certificate (or CA-signed certificate) is a certificate that has been issued by a trusted third party. To obtain a CA-signed certificate, you create a request file from your self-signed certificate and send it to a certificate authority for approval. You then import this CA-signed certificate into the server's KeyStore, replacing the self-signed certificate.

One advantage to using a CA-signed certificate instead of a self-signed certificate is that you do not need to import the CA-signed certificate into the client's TrustStore. The client already has a trusted root certificate for that CA either in the Glassfish Server instance or in the browser itself.

Creating a Self-Signed Certificate

In general, you use the Java **keytool** command to create a self-signed certificate on the same server where the KeyStore is located. If you create the self-signed certificate on another server, you need to transfer it from that server to the server where it will be used to create the KeyStore.

1. Decide on a certificate name.

   At a minimum, the certificate file should have **.jks** as its extension.

2. Determine if the KeyStore file already exists on the server.

   ```
   keytool -list -v -keystore path_to_keystore_file
   ```

3. Generate the self-signed certificate and place it in the KeyStore.

   ```
   keytool -genkeypair -alias alias_name -keyalg RSA -validity #_of_days -keysize
   2048 -keystore path_to_keystore_file
   ```

   where:

   - *alias_name*: Specifies a word of your choice, for example, the fully qualified domain name of the server host.
   - *#_of_days*: Specifies the number of days that the certificate is to be valid.
   - **2048** is the recommended key size.
   - *path_to_keystore_file*: Specifies the path to the KeyStore. This file or its parent directory must exist.

4. If you are going to have the certificate signed by a CA, you can skip to "Creating a CA-Signed Certificate Request". However, while you are waiting for the CA to return your certificate, you can use your self-signed certificate by continuing with the next steps.

5. Export the certificate needs to a certificate file.

   ```
   keytool -export -alias alias_name -keystore path_to_keystore_file -rfc -file
   path_to_certificate_file
   ```

   where:

   - *alias_name*: Specifies the same alias that was used to generate the certificate.
   - *path_to_keystore_file*: Specifies the same KeyStore path that was used to generate the certificate.
   - *path_to_certificate_file*: Specifies the exported certificate file, often given an extension of **.cert**.

Installing the Self-Signed Certificate on the Client

The certificate file generated in the previous step must be transferred to the client host and imported into the truststore of the client.

To import the certificate into the truststore of your client:

```
keytool -importcert -alias alias_name -file path_to_certificate_file -keystore
truststore_file
```

where:

- *alias_name*: Specifies a word of your choice, usually the same word you used when generating the certificate.
- *path_to_certificate_file*: Specifies path to where you stored the certificate file.
- *glassfish_truststore_file*: Specifies the TrustStore file used by your client.

Creating a CA-Signed Certificate Request

To generate a CA certificate request using **keytool**:

```
./keytool -certreq -alias alias_name -keystore path_to_keystore_file -file
request_file
```

where:

- *alias_name*: Specifies the alias that you gave to the self-signed certificate when you created it.
- *path_to_keystore_file*: Specifies path to the KeyStore file that holds your self-signed certificate.
- *request_file*: Specifies path to the request file output. This file is sent to the CA.

Submit the certificate request and get the certificate approved by the CA. As there are many ways to have your certificate request approved, this step is left up to you.

Importing a CA-Signed Certificate

When you receive the certificate from the CA it can be imported into your KeyStore on the server.

```
./keytool -import -trustcacerts -alias alias_name -file certificate_file -keystore
keystore_file
```

where:

- *alias_name*: Specifies the alias that you gave to the self-signed certificate when you created it.
- *certificate_file*: Specifies the path to the CA-signed certificate file.
- *keystore_file*: Specifies the path to the server KeyStore.

6

Configuring Client Authentication

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

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

Setting 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 the OpenSSL website at:

   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
   ..........................................................................
   ....+++
   ..............................+++
   writing 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
```

Enabling SSL and Client Authentication for a Listener

For this step, refer to "How to Enable SSL and Client Authentication for a Listener in Oracle GlassFish Server" at:
https://wikis.oracle.com/display/CommSuite/How+to+Set+Up+Certificate-based+Authentication+for+Convergence#HowtoSetUpCertificate-basedAuthenticationforConvergence-HowtoEnableSSLandClientAuthenticationforaListenerinOracleGlassFishServer

Generating 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 to use SSL client authentication.

4. Modify HTTPS listener, typically **http-listener-2**, to use your **TestSSLCert**, that is, changing it from **s1as** to your own **TestSSLCert**.

5. Add the following line to the **http-listener-2** property list to use the fallback feature if client authentication fails. If you do not want to fallback, change **want** to **need**.

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

Creating 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
   ```

```
.....................................................................
.....................................+++
.....................+++
writing 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 the preceding steps as often as required.

Configuring 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 the topic about certificates at:

   https://wikis.oracle.com/display/CommSuite/How+to+Set+Up+Certificate-based+Authentication+for+Convergence

4. Restart Glassfish Server.

Testing Certificate Authentication

1. In a browser in which 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 use.

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:
...
```

Installing the Client Certificate in Connector for Microsoft Outlook

To use certificate authentication to Calendar Server with Connector for Microsoft Outlook, import the client certificate by using Internet Explorer. This makes the certificate available to Connector for Outlook. See the topic on certificate-based authentication for more information at:

https://wikis.oracle.com/display/CommSuite/Certificate+based+Authentication+for+Connector+for+Microsoft+Outlook+7.3+Update+1

7

Configuring GlassFish Server to Use a CA-Signed Certificate

This chapter describes how to configure GlassFish Server to use a certificate issued by a Certification Authority (CA) to establish secure sessions through secure sockets layer (SSL) technology.

For complete instructions on how to request and install a certificate for GlassFish Server, refer to the official GlassFish Security documentation. Also see the Mozilla certutil page for more information on the **certutil** command at:

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

To configure GlassFish Server to use a CA signed certificate:

1. Change to the *GlassFish_home*/**lib** directory and run the **certutil** command to generate the certificate request.

   In the following example:

   - The Organization (O) field is required and must be filled in with organization name exactly (without commas or periods).
   - The Country (C) field is required and must be filled in with the two-letter code for the country in which the server resides, for example,**US**.

     VeriSign does not accept CSR's with a country code of **U**. Instead you must use the country code **GB**.
   - The State (ST) field is required and should be filled in with the full name, not the abbreviated name, of the state or province in which the server resides.
   - The Common name (CN) field is required and should be filled in with the fully qualified domain name (FQDN) of the server. A fully qualified domain name means the full name of the server host. For example, **demoserver.central.example.com** is a fully qualified domain name, while **demoserver.central** is not. DNS must be able to resolve the FQDN.
   - The Locality (L) field is optional, but if filled in appears in the certificate. Use the name of the city in which the server resides.
   - The Organization Unit (OU) field is optional, but if filled in appears in the certificate. You can use it to differentiate between multiple SSL server instances running on the same host. If you do not need to use it then leave it blank.
   - **-o** specifies the file to be created.
   - **-d** specifies the **config** directory of the GlassFish Server domain for which the certificate is being requested.

- **-a** specifies ASCII output.

```
Sample command to create a request:
./certutil -R -s "CN=demoserver.central.example.com,OU=Demo, O=Example
Inc,L=San Jose,ST=California,C=US" -o /export/tmp/democert-app-server.req -d
/opt/SUNWappserver/domains/domain1/config -a
Enter Password or Pin for "NSS Certificate DB": <This is the GlassFish
Enterprise Server's administrative password.>
A random seed must be generated that will be used in the
creation of your key. One of the easiest ways to create a
random seed is to use the timing of keystrokes on a keyboard.
To begin, type keys on the keyboard until this progress meter
is full. DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!
Continue typing until the progress meter is full:
|************************************************************|
Finished. Press enter to continue:
Generating key. This may take a few moments...
```

The Certificate Request resembles the following:

```
Certificate request generated by Netscape certutil
Phone: (not specified)
Common Name: <>
Email: (not specified)
Organization: <>
State: <>
Country: <>
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBzDCCATUCAQAwgYsxCzAJBgNVBAYTAklOMRIwEAYDVQQIEwlLYXJuYXRha2Ex
EjAQBgNVBAcTCUJhbmdhbG9yZTEeMBwGA1UEChMVU3VuIE1pY3Jvc3lzdGVtcyBJ
bmMuMQ4wDAYDVQQLEwVDb21tczEkMCIGA1UEAxMbY29tcy0xNTJ4LTE2OC5pbmRp
YS5zdW4uY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCf95F4RGUJYLHg
HGAEP4g2T45Wlf7soXfOBqVbani25NZXjQNKGKsPqvr8f6ata6OLFMgYzTzIkpI5
qkRWY/OIpEAfm3fxUuv+PWDxmdnQYqSc/t7+OdAqeeNk5Vlw0MI2PneOsXtm5hia
hOTKMAZPrQbnbtefwiKp58zKcwMFbwIDAQABoAAwDQYJKoZIhvcNAQEEBQADgYEA
AEmToTq5gDx2oOrYI8glXG6JbSpHJcf6AyW7TDTHPYRTSdx7N63LVG2IjkwYLoU8
nmj+RYp5srw/WatDF/Mm5lN9pjs6KP4fbu2HqI4XHDdJMjot9DPmAVTdqQwto1+e
quA85Lp7x0eGlbjoyYR51gDAGdDcfWcM51TZx0FGwwM=
-----END NEW CERTIFICATE REQUEST-----
```

2. Submit the certificate request and get the certificate approved by the Certificate Authority (CA).

   As there are many ways to have your certificate request approved, this step is left up to you. The approved certificate, in **pem** format, resembles the following:

```
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 3 (0x3)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=<>, ST=<>, L=<>, O=<>, OU=<>, CN=<>
Validity
Not Before: Sep 24 11:47:45 2009 GMT
Not After : Sep 24 11:47:45 2010 GMT
Subject: C=<>, ST=<>, O=<>, OU=<>, CN=<>
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:9f:f7:91:78:44:65:09:60:b1:e0:1c:60:04:3f:
88:36:4f:8e:56:95:fe:ec:a1:77:ce:06:a5:5b:6a:
```

```
78:b6:e4:d6:57:8d:03:4a:18:ab:0f:aa:fa:fc:7f:
a6:ad:6b:a3:8b:14:c8:18:cd:3c:c8:92:92:39:aa:
44:56:63:f3:88:a4:40:1f:9b:77:f1:52:eb:fe:3d:
60:f1:99:d9:d0:62:a4:9c:fe:de:fe:39:d0:2a:79:
e3:64:e5:59:70:d0:c2:36:3e:77:8e:b1:7b:66:e6:
18:9a:84:e4:ca:30:06:4f:ad:06:e7:6e:d7:9f:c2:
22:a9:e7:cc:ca:73:03:05:6f
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
Netscape Cert Type:
SSL Server
X509v3 Key Usage:
Digital Signature, Non Repudiation, Key Encipherment
Netscape Comment:
OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
A0:0B:AC:87:D6:29:DA:AD:1C:EC:82:85:33:C3:BC:09:E0:25:B4:2B
X509v3 Authority Key Identifier:
keyid:10:C9:2C:31:AC:4A:A5:F1:08:0B:28:15:96:3F:1D:1A:71:33:E7:47
DirName:/C=<>/ST=<>/L=<>/O=<>/OU=<>/CN=<>
serial:CA:0F:64:A4:89:4F:2C:21
Signature Algorithm: md5WithRSAEncryption
51:5e:8b:08:bc:fa:9d:21:be:c6:1e:6b:30:d4:7d:a7:ef:86:
28:1b:6f:e4:66:c0:69:64:14:19:07:e9:5d:ad:a0:bb:ce:1a:
3c:27:81:30:3e:65:46:57:60:4c:a6:8c:76:a2:2e:14:4f:12:
35:a2:04:e9:36:31:2b:4e:c1:63:be:89:db:30:b8:01:78:c8:
39:0d:7d:2c:87:c9:cb:72:5d:1e:88:87:e7:ce:0f:b8:45:8a:
d7:66:a1:5a:d0:bf:3a:67:bd:a2:b9:65:21:f5:e5:db:8b:cf:
c0:39:18:66:96:79:7e:96:b3:21:00:c5:4a:24:bb:42:ad:52:
d4:f1
-----BEGIN CERTIFICATE-----
MIID7jCCA1egAwIBAgIBAzANBgkqhkiG9w0BAQQFADCBpTELMAkGA1UEBhMCSU4x
EjAQBgNVBAgTCUthcmFudGFrYTESMBAGA1UEBxMJQmFuZ2Fsb3JlMR8wHQYDVQQK
ExZTdW4gTWljcm9zeXN0ZW1zIEluZGlhMRQwEgYDVQQLEwtDb21tc1FBLUlFQzET
MBEGA1UEAxMKQ0EtQ29tbXNRQTEiMCAGCSqGSIb3DQEJARYTYWRtaW5AaW5kaWEu
c3VuLmNvbTAeFw0wOTA5MjQxMTQ3NDVaFw0xMDA5MjQxMTQ3NDVaMHcxCzAJBgNV
BAYTAklOMRIwEAYDVQQIEwlLYXJuYXRha2ExHjAcBgNVBAoTFVN1biBNaWNyb3N5
c3RlbXMgSW5jLjEOMAwGA1UECxMFQ29tbXMxJDAiBgNVBAMTG2NvbXMtMTUyeC0x
NjguaW5kaWEuc3VuLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAn/eR
eERlCWCx4BxgBD+INk+OVpX+7KF3zgalW2p4tuTWV40DShirD6r6/H+mrWujixTI
GM08yJKSOapEVmPziKRAH5t38VLr/j1g8ZnZ0GKknP7e/jnQKnnjZOVZcNDCNj53
jrF7ZuYYmoTkyjAGT60G527Xn8IiqefMynMDBW8CAwEAAaOCAVkwggFVMAkGA1Ud
EwQCMAAwEQYJYIZIAYb4QgEBBAQDAgZAMAsGA1UdDwQEAwIF4DAsBglghkgBhvhC
AQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFKAL
rIfWKdqtHOyChTPDvAngJbQrMIHaBgNVHSMEgdIwgc+AFBDJLDGsSqXxCAsoFZY/
HRpxM+dHoYGrpIGoMIGlMQswCQYDVQQGEwJJTjESMBAGA1UECBMJS2FyYW50YWth
MRIwEAYDVQQHEwlCYW5nYWxvcmUxHzAdBgNVBAoTFlN1biBNaWNyb3N5c3RlbXMg
SW5kaWExFDASBgNVBAsTC0NvbW1zUUEtSUVDMRMwEQYDVQQDEwpDQS1Db21tc1FB
MSIwIAYJKoZIhvcNAQkBFhNhZG1pbkBpbmRpYS5zdW4uY29tggkAyg9kpIlPLCEw
DQYJKoZIhvcNAQEEBQADgYEAUV6LCLz6nSG+xh5rMNR9p++GKBtv5GbAaWQUGQfp
Xa2gu84aPCeBMD5lRldgTKaMdqIuFE8SNaIE6TYxK07BY76J2zC4AXjIOQ19LIfJ
y3JdHoiH584PuEWK12ahWtC/Ome9orllIfXl24vPwDkYZpZ5fpazIQDFSiS7Qq1S
1PE=
-----END CERTIFICATE-----
```

3. Once you have obtained your CA signed and approved SSL server certificate, install it by using the **certutil** command.

```
./certutil -A -n TestSSLCert -t "P,u,u" -d GlassFish_home/domain-config
```

```
directory -i /space/smime/ssl-certs/certs/<>.pem
```

TestSSLCert is an example of the certificate nickname that you need to provide in the GlassFish Enterprise Server configuration.

4. Verify that the certificate was installed.

```
 ./certutil -L -d GlassFish_home/domain-config
(The output is similar to the following.)
verisignclass1ca T,c,c
thawtepersonalpremiumca T,c,c
baltimorecodesigningca T,c,c
TestSSLCert T,c,c
verisignclass2g2ca T,c,c
verisignclass3g3ca T,c,c
entrustglobalclientca T,c,c
entrustsslca T,c,c
verisignclass3g2ca T,c,c
thawtepremiumserverca T,c,c
entrust2048ca T,c,c
valicertclass2ca T,c,c
gtecybertrust5ca T,c,c
equifaxsecureebusinessca1 T,c,c
verisignclass1g3ca T,c,c
godaddyclass2ca T,c,c
thawtepersonalbasicca T,c,c
verisignclass1g2ca T,c,c
verisignclass2g3ca T,c,c
equifaxsecureca T,c,c
entrustclientca T,c,c
verisignserverca T,c,c
geotrustglobalca T,c,c
equifaxsecureebusinessca2 T,c,c
s1as u,u,u
sslCACert T,c,c
verisignclass3ca T,c,c
verisignclass2ca T,c,c
sslcert1 pu,pu,pu
gtecybertrustglobalca T,c,c
entrustgsslca T,c,c
thawtepersonalfreemailca T,c,c
thawteserverca T,c,c
baltimorecybertrustca T,c,c
starfieldclass2ca T,c,c
equifaxsecureglobalebusinessca1 T,c,c
TestSSLCert P,u,u
```

5. Log in to the GlassFish Server Administration Console and change the SSL certificate nickname. (This example uses **TestSSLCert**.) If you want the JMX connector to also use the new certificate, perform the following:
   a. From Configuration, select **server-config**.
   b. Select **admin service**.
   c. Select **system**.
   d. Select the **SSL** tab.
   e. Change the SSL certificate nickname to be the new one you want to use.
6. Run any **asadmin** command to prompt you to accept the new certificate:

```
cd GlassFish_home/bin
./asadmin list-jms-hosts
Do you trust the above certificate y? yes
```

Accepting the certificate updates your **.asadmintruststore** file.

7. Restart the GlassFish Server domain.
8. If you have installed the CA certificate after running the **init-config** command, copy the **.asadmintruststore** file under the root directory to Calendar Server's **config** directory, by default **/var/opt/sun/comms/davserver/config**.