Oracle® Communications Contacts Server

Security Guide

Release 8.0

E50288-05

May 2021

ORACLE

Oracle Communications Contacts Server Security Guide, Release 8.0

E50288-05

Contents

A Secure Deployment Checklist

Preface

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

Audience

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

Related Documents

For more information, see the following documents in the Contacts Server Release 8.0 documentation set:

- *Contacts Server System Administrator's Guide*: Provides instructions for administering Contacts Server.
- *Contacts Server Release Notes*: Describes the known issues and required third-party products and licensing.
- *Contacts Server Installation and Configuration Guide*: Describes the requirements for installing Contacts Server, installation procedures, and post-installation tasks.
- *Contacts Server RESTful Protocol Guide*: Describes the RESTful protocol that enables HTTP clients to fetch, add, and edit address book related data that is stored by Contacts Server.

Nomenclature

The following nomenclature is used throughout the document.

Convention	Meaning
Application Server	The term **Application Server** or **application server** is used in this document to refer to either GlassFish Server or WebLogic Server. **Supported Application Server**: Oracle Communications Contacts Server 8.0.0.4.0 and previous releases were deployed on GlassFish Server, which is no longer supported by Oracle. For that reason, Contacts Server 8.0.0.5.0 and beyond are only supported on Oracle WebLogic Server. Oracle strongly recommends that you upgrade your Contacts Server environments to release 8.0.0.5.0 or higher and migrate to WebLogic Server to receive full Oracle support.

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

Contacts Server Security Overview

This chapter provides an overview of Oracle Communications Contacts 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 Contacts Server Installation" for more information.
5. **Learn about and use Contacts Server security features.** See "Implementing Contacts 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 website at:

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

Understanding the Contacts Server Environment

When planning your Contacts Server implementation, consider the following:

- Which resources must be protected?

  For example:

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

  - Dependent resources, such as the application server and Oracle Directory Server Enterprise Edition

- From whom am I protecting the resources?

  In general, resources must be protected from everyone on the Internet. But should the Contacts Server deployment be protected from employees on the intranet in your enterprise? Should your employees have access to all resources within the application 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?

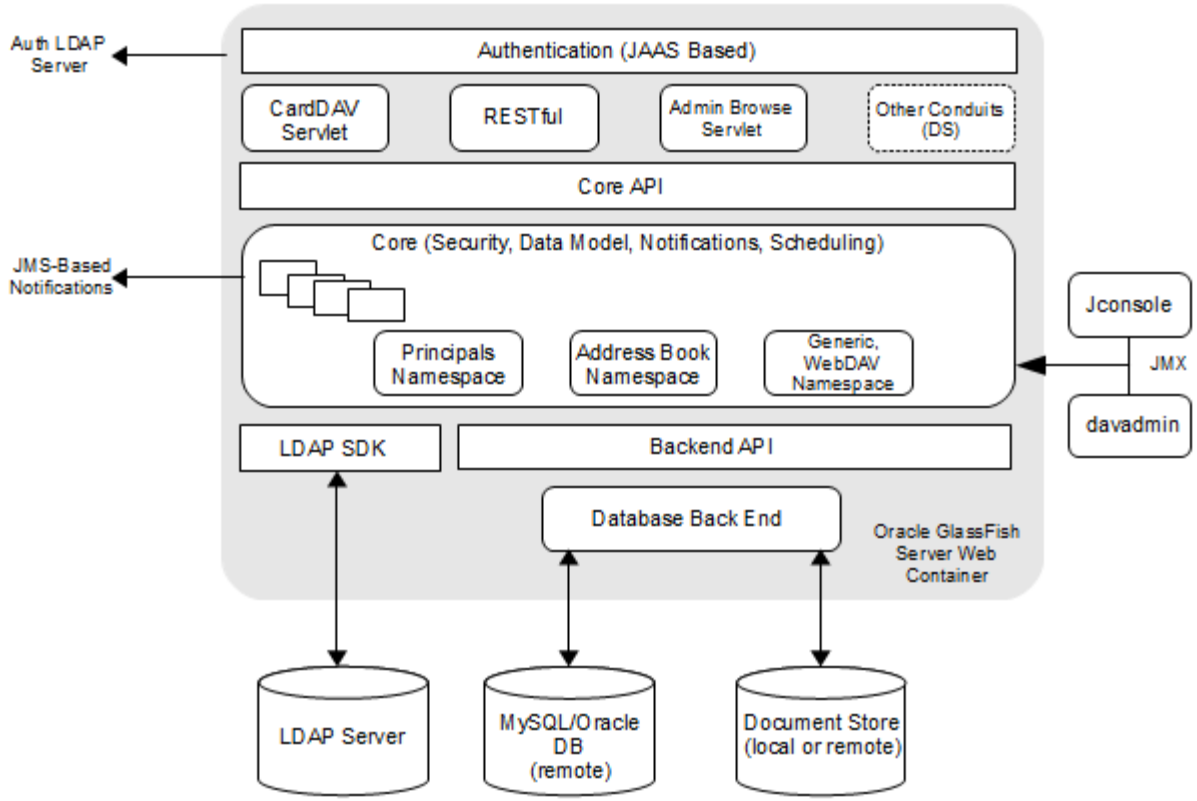
  In some cases, a fault in your security scheme is easily detected and considered nothing more than an inconvenience. In other cases, a fault might cause great damage to companies or individual clients that use Contacts Server. Understanding the security ramifications of each resource help you protect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

Overview of Contacts Server Security

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

Note: The following figure includes Oracle GlassFish Server as an example for a container. However, this figure is also applicable for Oracle WebLogic Server as a container.

Figure 1–1 Contacts Server Components



Recommended Deployment Topologies

You can deploy Contacts 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 *Contacts 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 topic on determining your Communications Suite network infrastructure needs in *Communications Suite Deployment Planning Guide*.

Operating System Security

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

Firewall Port Configuration

Contacts 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 (the default value is 3306)
- Oracle Database server port (the default value is 1521)

- Contacts Server back-end remote document store port (the default value is 8008)
- GlassFish Server administration server port (the default value is 4848)
- Contacts Server access port (the default value is 443)
- Notification mail server port (the default value is 25)
- Oracle WebLogic Server Administration Server and Managed Server ports (the default values are 7002 and 7004)

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*: `Oracle Database 19c Documentation`.

For more information about securing MySQL Server, see `Security in MySQL`.

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

SSL enables secure communication between applications connected through the Web. In a Contacts Server deployment, you can configure SSL between the following components:

- Application server and client connections
- Contacts Server and Directory Server
- Application server and the JMX port used by Contacts Server administration utility
- Contacts Server and the back-end database
- Contacts Server and the document store

See "Implementing Contacts Server Security" for more information.

LDAP Security

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

2

Performing a Secure Contacts Server Installation

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

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

Installing Infrastructure Components Securely

Contacts Server is deployed within the application server.

When installing and configuring GlassFish Server, it is recommended to:

- Use a non-root user account to install and run 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*.

When installing and configuring WebLogic Server, it is recommended to:

- Use a non-root user account to install and run WebLogic Server
- Configure SSL Keystores and HTTPS port for Administration Server and Managed Server
  - Oracle WebLogic Server provides four keystore options in its configuration. However, Contacts Server supports only **CustomIdentityandCustomTrust** and **CustomIdentityandJavaStandardTrust** options. You can use one of these options.

    Note: Ensure to configure the Administration server and Managed servers similarly. It means you should configure the same options and certificates for the Administration Server and Managed servers.

  - The keystores passwords must match with the password of the WebLogic Server Administration password.

> **Note:** Contacts Server is deployed on WebLogic Server only if the passwords of Keystores and WebLogic Server match.

For more information about configuring and administering WebLogic Server Security, see WebLogic Server documentation:
https://docs.oracle.com/middleware/12213/wls/wls-secure.htm

Contacts 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 *Contacts Server Installation and Configuration Guide*.

Credentials Needed to Install Contacts Server Components

The installation prompts for authentication credentials for the following:

- Database user
- Application server's administrator
- Directory Server manager (bind DN and password)
- Contacts Server administrator

Post-Installation Configuration

After the installation, configuring Contacts Server for a secure deployment involves the following procedure:

> **Note:** In the following steps, application server refers to the application server on which Contacts Server is deployed.

1. Ensure that HTTPS is configured correctly on the front-end application 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 application server host.
3. Ensure that JMX port for the application server uses 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 Contacts Server database
   - Set up secure communications to the remote document store
7. Add LDAP access control for Contacts Server.

See "Implementing Contacts Server Security" for more information.

3

Implementing Contacts Server Security

This chapter explains the security features of Oracle Communications Contacts 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
- Configuring HTTPS on Front-End WebLogic Server Hosts
- Disabling SSLv3 on Front-End Oracle WebLogic Server Hosts
- Enabling LDAP SSL in Contacts Server
- Enabling Secure Notification Mail Submission
- Configuring and Using Authentication
- Configuring and Using Access Control
- Adding LDAP Access Control for Contacts Server Features
- Configuring and Using SSL with the Contacts Server Database and Document Store
- Securing Contacts Server and Application Server Secure
- Detecting Possible Security Issues

About System Security in Contacts 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,

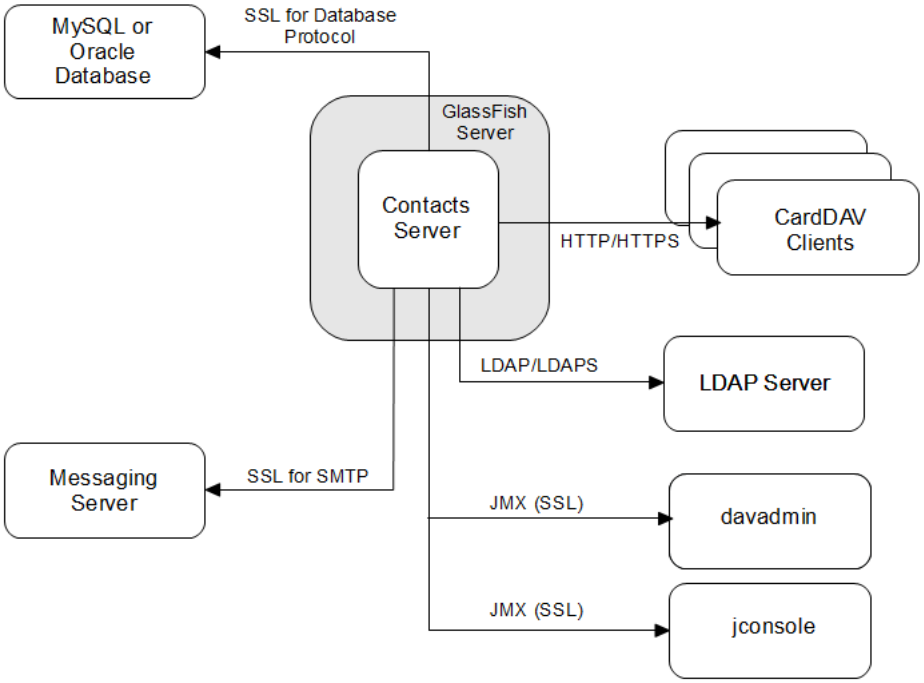
a digital certificate, or something else. Usually, servers or applications require clients to authenticate themselves. Additionally, clients might require servers to authenticate themselves. When authentication is bidirectional, it is called *mutual authentication*.

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.

Contacts 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 Contacts Server that can be secured. The HTTPS, LDAPS, JMX, and SMTPS protocols must be secured by using SSL. SQL/JDBC connections between the application server and the database are also secured by using SSL.

Figure 3–1 Contacts Server Protocol Flows



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 about configuration parameters in *Contacts Server System Administrator's Guide* for information on the **notification.dav.smtp.*** parameters. See "Enabling LDAP SSL in Contacts Server" for information on LDAPS. See "Configuring SSL for the MySQL Database" 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 topic on designing for security in *Unified Communications Suite Deployment Planning Guide*.

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

The default GlassFish Server installation comes with a self-signed certificate, which is incompatible with production usage. To install an official certificate, see the topic on configuring GlassFish Server to use a Certificate Authority in *Calendar Server System Administrator's Guide*.

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 Contacts 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 Contacts 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 modify** command to set the **davcore.uriinfo.fulluriprefix** parameter to the desired host name and port. For example:

```
davadmin config modify -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 Contacts 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 Contacts 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.

Configuring HTTPS on Front-End WebLogic Server Hosts

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

- Setting up Certificates for Production Environment
- Configuring Keystore and SSL in WebLogic Server
- Ensuring the URL Prefix when Reconfiguring for SSL

Setting up Certificates for Production Environment

Generate and set up certificates that WebLogic Server supports. For more information, see the discussion about obtaining and storing certificates for production environments on Oracle Fusion Middleware Administering Security for Oracle WebLogic Server.

To ensure generated certificates are in the appropriate format that WebLogic Server requires and the usage of host name verifiers are appropriate, refer to the following topics in the Fusion Middleware Administering Security for Oracle WebLogic Server guide:

- Using the Default WebLogic Server Host Name Verifier
- How the Wildcarded Host Name Verifier Works
- Configure a custom host name verifier

Configuring Keystore and SSL in WebLogic Server

Configure Oracle WebLogic Server in a secure mode.

To enable SSL and configure keystores in Oracle WebLogic Server, see Configuring Keystores and Set up SSL on the Oracle WebLogic Server documentation.

Oracle WebLogic Server provides four keystore options in its configuration. However, Contacts Server supports only **CustomIdentityandCustomTrust** and **CustomIdentityandJavaStandardTrust** keystore options. You can use one of these options.

Note: You must always set the keystore type to **JKS**.

The keystore configuration must be the same for an Administration Server and Managed Servers. It means you should configure the same options and certificates on these servers for hosting Contacts Server.

You must set keystore passwords identical to the Oracle WebLogic Server Administration Sever password.

Ensuring the URL Prefix when Reconfiguring for SSL

When you configure Contacts Server, if you specify the HTTP port, the **davcore.uriinfo.fulluriprefix** parameter is set to **http://***host:http_port*. As a result, Contacts Server constructs all URLs that have **http://**. For example, it constructs the URLs pointing to attachments to include the **http://** *URL*. When you reconfigure Contacts Server to use SSL, the host name part of this prefix should match the host name associated with the certificate.

To change **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 Oracle WebLogic Server Hosts

Identify the http-listener for the publicly accessible port that has SSL/TLS enabled on which requests for Contacts Server are received.

To disable SSLv3 on front-end Oracle WebLogic Server hosts:

1. Log in to the Oracle WebLogic Server Administration Console.
2. In the **Domain Structure** section, expand the **Environment** node.
3. Click **Servers**. The Summary of Servers page appears.
4. Click the name of the Managed Server in which Contacts Server is deployed. The settings page for the selected server appears.
5. Click **Lock & Edit**.
6. Click the **Configuration** tab, and then the **Server Start** tab.
7. Add the **weblogic.security.SSL.protocolVersion** and **weblogic.security.SSL.minimumProtocolVersion** parameters to the **Arguments** field.

   For example,

   -Dweblogic.security.SSL.protocolVersion=All: This is the default behavior. If **ALL** is selected, the default value depends on the JSSE provider and JDK version. For the supported protocol version table for Sun JSEE, see Java SE 8 Documentation.

   -Dweblogic.security.SSL.protocolVersion=TLS1 -Dhttps.protocols=TLSv1: This property value enables any protocol starting with TLS for messages that are sent and accepted, for example, TLS V1.0, TLS V1.1, and TLS V1.2.

   For information on specifying the SSL protocol version, see *Oracle Fusion Middleware Administering Security for Oracle WebLogic Server*.
8. Click **Save**.

9. Click **Activate Changes**.
10. Restart Oracle WebLogic Server.

Enabling LDAP SSL in Contacts Server

If you specified to use an LDAPS URL during the Contacts 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 *ContactsServer_home***/lib/** directory.

To configure Contacts 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 modify** command to make the configuration change. For example:

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

   Warning messages about restarting the application server are displayed.

3. Restart the application server.

Enabling Secure Notification Mail Submission

The Contacts 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 Contacts 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 application 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.nab.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 Contacts Server and LDAP authentication, see the topic on provisioning users in *Calendar Server System Administrator's Guide*.

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 Contacts Server Features

This section provides sample Access Control Instructions (ACIs) that show the attributes that Contacts Server needs for granting end users permission to search the LDAP directory for other users, groups, and resources. These samples use **inetorgperson** as the object class for users. Tailor these samples to your individual site's security needs. For example, you can use **nabuser** if that is how you have provisioned users. 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, groups, 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:///($dn),o=isp")
 (targetattr!="userPassword")
(targetfilter=(|(objectClass=inetorgperson)(objectclass=inetresource)(objectcla
ss=groupofuniquenames)(objectclass=groupofurls)(objectclass=inetmailgroup)))
 (version 3.0; acl "Contacts Server User search access to all users, groups,
and resources in own domain - product=nabserver,class=install,num=3,version=1";
 allow (read,search)
 userdn="ldap:///[$dn],o=isp??sub?(objectclass=inetorgperson)";)
```

- 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=inetorgperson)(objectclass=inetresource)(objectcla
ss=groupofuniquenames)(objectclass=groupofurls)(objectclass=inetmailgroup)))
 (version 3.0; acl "Contacts Server search access to all users, groups, and
resources in domain A - product=nabserver,class=install,num=3,version=1";
 allow (read,search)
 userdn="ldap:///domainA.orgdn??sub?(objectclass=inetorgperson)";)
```

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=inetorgperson)(objectclass=inetresource)(objectcla
ss=groupofuniquenames)(objectclass=groupofurls)(objectclass=inetmailgroup)))
 (version 3.0; acl "example.com users, groups, and resources access in
varrius.com - product=nabserver,class=install,num=3,version=1";
 allow (read,search)
 userdn="ldap:///o=example.com,o=isp??sub?(objectclass=inetorgperson)";)
```

- 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 Security Guide*. Though that guide is written for Contacts Server, the same concepts apply to Contacts Server.

Configuring and Using SSL with the Contacts Server Database and Document Store

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

Configuring SSL for the MySQL Database

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

These instructions use a self-signed certificate for the MySQL server and the MySQL Connector/J client within GlassFish Server. These instructions also assume that the **/etc/mysql** directory already exists.

To configure SSL for the MySQL back-end database:

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.

```
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 it is 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:
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:
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 the application server 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. If you use GlassFish Server, stop GlassFish Server.

7. Add the following parameters in the **domain.xml** file to the existing Pool and 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. Start GlassFish Server.
9. If you use WebLogic Server, add the parameters listed in step 16 using WebLogic Server Administration Console for the JDBC data source used by Contacts Server.

   Note: By default, **defaultbackend** JDBC Data source is created during the Contacts Server configuration.

10. Log in to WebLogic Server Administration Console.
11. Click **Lock & Edit**.
12. Click your domain directory.
13. Navigate to **Services** and then **Data Sources**.

    The **defaultbackend** is listed in the **Configuration** tab.
14. Click the backend.
15. Click the **Connection Pool** tab.
16. In the text box under **Properties**, add the following properties after the existing properties:

    Note: By default, these properties are listed in the Properties text box: **user**, **characterEncoding**, and **databaseName**.

```
<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">
```

17. Click **Save**.
18. Click **Activate Changes**.
19. Restart WebLogic Server.
20. (Optional) For debugging SSL communication, use the following log settings:
    - If you use GlassFish Server:

      To log the SSL communication with GlassFish Server, before starting GlassFish Server, add the **-Djavax.net.debug=ssl** value to the JVM options in the **domain.xml** file:

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

      After adding the **-Djavax.net.debug=ssl** value, the GlassFish Server log file is modified as shown in the following sample file:

```
INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=39;_
ThreadName=Timer-8;| Timer-8, WRITE: TLSv1 Handshake, length = 163|#]
|INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=39;_
ThreadName=Timer-8;| Timer-8, READ: TLSv1 Handshake, length = 74|#]
|INFO|sun-appserver2.1|javax.enterprise.system.stream.out|_ThreadID=39;_
ThreadName=Timer-8;| *** ServerHello, TLSv1|#]
```

- If you use WebLogic Server:

  See the information about SSL Debugging and specifying JAVA_OPTIONS in the following WebLogic Server 12.2.1.3 documentation:

  Oracle Fusion Middleware Administering Security for Oracle WebLogic Server

  Oracle Fusion Middleware Administering Server Startup and Shutdown for Oracle WebLogic Server

  For example, in the **StartWeblogic.sh** script that starts the Administration server, you can add the following parameter:

  ```
  JAVA_OPTIONS=${JAVA_OPTIONS} -Djavax.net.debug=ssl
  ```

  After adding the above parameter in the Administration server, the WebLogic Administration Server log file is modified as shown in the following sample file:

  ```
  [ACTIVE] ExecuteThread: '13' for queue: 'weblogic.kernel.Default
  (self-tuning)', WRITE: TLSv1 Application Data, length = 48
  [ACTIVE] ExecuteThread: '13' for queue: 'weblogic.kernel.Default
  (self-tuning)', READ: TLSv1 Application Data, length = 80
  ```

Configuring SSL for the Oracle Database

You can enhance security by configuring SSL communication between the Contacts Server front-end and back-end hosts and Oracle databases. To do so, first enable the Oracle back-end database server for SSL by setting up the required Oracle wallet and Oracle Net Listener. Then configure the Contacts Server front-end hosts to connect over SSL by setting the properties on the JDBC connection pool setting.

You can also configure SSL communication between the Contacts 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 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 and specific commands, see Managing Oracle Wallets with the orapki Utility in Oracle Database Security Guide, 19c.

   For specific commands, see orapki Usage Examples in Oracle Database Security Guide, 19c.

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.
3. If the certificate authority is not known to the application server instance hosting the Contacts Server front end, import the certificate authority's certificate into the application server JVM.
   - 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 the application 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 application server machine hosting the Contacts Server front end, import the root certificate into the application server JVM as described in step 3 of 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 Network Configuration File: /u01/app/oracle/product/11.2.0/dbhome_
1/network/admin/sqlnet.ora
# 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 Network Configuration File:
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
# 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 Network Configuration File:
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/listener.ora
# 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 previous examples when entering values in the GUI.

To use Oracle Net Manager to enable the Oracle Net listener for SSL communication:

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** pulldown 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

To complete the installation and modify JDBC connection pool settings:

1. Install and configure Contacts Server to communicate with the Oracle database using the non-SSL port.
2. Modify the Pool and 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 Contacts Server and the remote document store. Configuring SSL between Contacts 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 Contacts Server host, if necessary.
5. Make Contacts Server configuration changes to support SSL for the document store.

Creating an SSL Certificate

To create a certificate, refer to the SSL certificate documentation in *Calendar Server Security Guide*.

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 the SSL certificate documentation in *Calendar Server Security Guide*.

Changing SSL Configuration 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 Contacts Server Host

If you are using a CA-signed certificate, you do not need to install the certificate on the Contacts Server host, as long as the instance of the application 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 the application server on the Contacts Server host. See

the topic on importing a certificate in the SSL certificate documentation in *Calendar Server Security Guide*.

Configuring Contacts Server to Use SSL for the Document Store

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

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

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

2. Restart the application server.

Securing Contacts Server and Application Server Secure

Note: In this section, application server refers to the application server on which Contacts Server is deployed.

To secure Contacts Server and the application server, perform the following tasks:

- Preventing Denial of Service Attacks on Application Server
- Configuring JMX Port for GlassFish Server to Use SSL
- Configuring JMX Port for WebLogic Server to Use SSL

Preventing Denial of Service Attacks on Application Server

Using the application 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 list of host names and/or IP addresses to be blocked

For more information, see the topic on DoS prevention in *Contacts 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 Contacts 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 Contacts 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/nabserver/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
   ```

   Alternatively, 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**.

   e. On the **SSL** tab, set the Certificate Nickname to the same name you used for your certificate's alias in step 1.

Configuring JMX Port for WebLogic Server to Use SSL

If Contacts Server is deployed on WebLogic Server, the JMX Port becomes the port of the Managed Server. It is recommended that for production environments, you use the secure configuration and secure port of managed server to access Contacts Server. For more information, see "Configuring HTTPS on Front-End WebLogic Server Hosts".

Detecting Possible Security Issues

You can use the Contacts 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> NabServlet [REQ] REPORT
/nabserver/principals/jsmith IP_address server:port
```

A

Secure Deployment Checklist

The following security checklist provides guidelines to help you secure Oracle Communications Contacts Server and its components.

Secure Deployment Checklist

- Install only the components you require.
- Lock and expire default user accounts.
- Use a strong LDAP password policy for user authentication.
- Enable data dictionary protection on the Oracle Database for Contacts Server.
- Restrict, control, and revisit user privileges:
  - Grant only the necessary privileges to each user.
  - Revoke unnecessary privileges from the PUBLIC user group.
  - Restrict permissions on run-time facilities.
- Enforce the use of access controls by using the Authorization Policies.
- Require clients to authenticate.
- Restrict network access by doing the following:
  - Use firewalls.
  - Never leave an unnecessary hole in a firewall.
  - Password-protect the Oracle listener against remote access.
  - Monitor listener activity.
  - Monitor who accesses your systems.
  - Restrict system access by IP addresses.
  - Encrypt network traffic.
- Apply all security patches and workarounds.
- Encrypt sensitive information.
- Contact Oracle Security Products if you discover a vulnerability in any Oracle product.