SSL Security with an Oracle Database

Best Practices Guide

ImageNow Version: 6.7.x

perceptivesoftware

Table of Contents

Secure Sockets Layer security and Oracle

SSL security uses cryptography and symmetric encryption to provide communication at the transport layer for data sent over a network for application-specific protocols. Mutual authentication, also known as two-way authentication, ensures that both servers trust each other. This document contains the procedures necessary for configuring Mutual Authentication between an Oracle database server and a trusted client server.

Note You must have Oracle Enterprise Edition with the Advanced Security option installed on your database server. If you do not have the Oracle client installed on your client server, you can create the client server wallet and certificates on the database server and copy over the client server wallet.

Oracle database server

To configure the Oracle database server, complete the following procedures.

1. Create a new wallet.
2. Create a certificate request.
3. Export the certificate request.
4. Import the certificates.
5. Add wallet permissions.
6. Configure the SQLNET.ora file.
7. Configure the LISTENER.ora file.
8. Configure the TNSNAMES.ora file.

ImageNow Server

To configure the ODBC Datasource for ImageNow Server, complete the following procedures.

Note If ImageNow Server does not have the Oracle client, create a separate client server wallet on the Oracle database server. Copy the Oracle client wallet file (ewallet.p12) to ImageNow Server and specify that file as the Key Store for the DataDirect ODBC driver.

Prerequisite The Oracle database server must be set up for SSL.

1. Create a new wallet.
2. Create a certificate request.
3. Export the certificate request.
4. Import the certificates.
5. Add wallet permissions.
6. Configure the ImageNow ODBC Datasource.

Business Intelligence

To configure SQLNET for Business Intelligence, complete the following procedures.

Prerequisite The Oracle database server must be set up for SSL.

1. [Create a new wallet](#).
2. [Create a certificate request](#).
3. [Export the certificate request](#).
4. [Import the certificates](#).
5. [Add wallet permissions](#).
6. [Configure the SQLNET.ora file](#).
7. [Configure the TNSNAMES.ora file](#).

Set up SSL encryption

Create a new wallet

You can use Oracle Wallet Manager (OWM) or the Oracle orapki utility to create PKCS#12 wallets that store credentials in a directory on your file system. This is the standard wallet type. To create a new wallet, complete the following step.

- Complete the procedure that is appropriate for your situation.

Situation	Steps
Oracle Wallet Manager (OWM)	1. Click **Wallet** > **New**. 2. Click **Yes** to create the default wallet directory if it does not exist. 3. In the **New Wallet** dialog box, provide and confirm a password for the wallet. 4. Select the **Standard** wallet type. 5. In the **OWM** dialog box, click **Yes** to confirm creation. 6. Click **Wallet** > **Save**. 7. In the **Select Directory** dialog box, choose a directory to store the wallet files. 8. Click **Wallet** > **Auto Login**. 9. Click **Wallet** > **Save**. **Note** You now have a cwallet.sso and ewallet.p12 file. You may have to adjust the permissions on these files, and possibly the directories containing these files, in order to be able to access the wallet during connection attempts.

Situation	Steps
Command line	• Enter the following command in a command prompt. `orapki wallet create -wallet /u01/app/oracle/product/11.2.0/db_1/owm/wallets/oracle -auto_login -pwd <wallet password>` **Note** The wallet password must be sufficiently complex to create a wallet.

Create a certificate request

Create a certificate request to export. It is recommended to use the fully qualified server name, or some other unique identifier. The Common Name should represent the identity of the server. To create a certificate request, complete the following step.

• Complete the procedure that is appropriate for your situation.

Situation	Steps

Oracle Wallet Manger (OWM)	1. In the **OWM**, go to **Operations** and select **Add Certificate Request**.
	
	2. In the **Create Certificate Request** dialog box, enter the appropriate information to create an identity.
	
	    • Common Name = *<unique identification string>*
	
	    • Key Size = 4096
	
	    Note It is recommended that you use a key size of 1024 or greater. Larger key sizes are more secure.
	
	3. Click **OK** to acknowledge creation of the certificate request.

Command line	• Enter the following command in a command prompt. `orapki wallet add -wallet /u01/app/oracle/product/11.2.0/db_1/owm/wallets/oracle -dn "CN=<identification string>" -keysize 4096 -pwd <wallet password>`

Export the certificate request

You need to export the certificate request to be signed by a Certificate Authority. To export the certificate request, complete the following step.

- Complete the procedure that is appropriate for your situation.

Situation	Steps
Oracle Wallet Manger (OWM)	1. In the **OWM**, right-click on **Requested Certificate** and select **Certificate:[Requested]** > **Export Certificate Request**. 2. In the **Export Certificate Request** dialog box, enter the path for the certificate request file. **Example** C:\oracle\product\11.2.0\db_1\owm\wallets\<*user*> 3. Click **Save** to save the file as a CSR file and complete the certificate request export. **Step Result** Your certificate request is now ready to be signed by a Certificate Authority. Contact your administrator for information on how your company obtains signed certificates.

Command line	- Enter the following command in a command prompt.
	
	`orapki wallet export -wallet`
	`/u01/app/oracle/product/11.2.0/db_1/owm/wallets/oracle`
	`-dn "CN=<identification string>" -request`
	`[path]/<certificate request name>.csr`

Import the certificates

You must import both a Certificate Authority (CA) and signed user certificates. The wallet must contain the trusted certificate representing the Certificate Authority (CA) who issued that user certificate before you import the user certificate. To import certificates, complete the following steps.

- Complete the procedure that is appropriate for your situation.

  Note Import the CA certificate prior to importing the user certificate.

Situation	Steps

Oracle Wallet Manger (OWM)	1. In **OWM**, click **Operations** and select **Import Trusted Certificate**.
	
	2. Select the **Select a file that contains the certificate** option and click **OK**.
	
	3. Navigate to the file location and click **Open**.
	
	  Step Result The new trusted certificate displays in the **Trusted Certificates** list.
	
	4. From the **Operations** menu, select **Import User Certificate**.
	
	5. Select the **Select a file that contains the certificate option** and click **OK**.
	
	6. Navigate to the file location and click **Open**.
	
	  Step Result The certificate changes from **Certificate:[Requested]** to **Certificate:[Ready]**.
	
	7. Remove all other default trusted certificates. Right-click on the trusted certificate you want to remove and select **Remove Trusted Certificate**.
	
	8. Click **Yes** to confirm you want to remove the trusted certificate.
	
	9. **Save** and **Close** your wallet.

Command line	1. Import the CA certificate by entering the following command in a command prompt.
	
	`orapki wallet add -wallet /u01/app/oracle/product/11.2.0/db_1/owm/wallets/oracle -trusted_cert -cert [path]/<ca certificate>.cer -pwd <wallet password>`
	
	2. Import the user certificate by entering the following command in a command prompt.
	
	`orapki wallet add -wallet /u01/app/oracle/product/11.2.0/db_1/owm/wallets/oracle -user_cert -cert [path]/<user certificate>.cer -pwd <wallet password>`

Add wallet permissions

You need to add privileges for the appropriate user or group for your cwallet.sso and ewallet.p12 files. To add wallet permissions, complete the following step.

- Complete the procedure that is appropriate for your situation.

Situation	Steps
Oracle Wallet Manger (OWM) (Windows)	1. Select the file, right-click, and select **Properties**. 2. Select the **Security** tab and click **Change Permissions**. 3. Click **Add** > **Locations**, and select the appropriate location. 4. In the **Select User or Group** field, type `ora_dba`. Click the **Check names** button to verify that the ora_dba group exists. 5. Click OK and the **Permission Entry** dialog box displays. 6. Select the **Allow** check box next to **Full Control** and click **OK**. 7. In the **Advanced Security** dialog box click **Apply**. 8. Click **OK** to exit the dialogs.

Command line (Windows/UNIX)	- Enter the following commands at the command prompt.
	 `chmod 775 ewallet.p12`
	 `chmod 775 cwallet.sso`

Configure database server ORA files for SSL

To configure Oracle SQLNET files to use SSL, complete the following procedures.

1. Configure the SQLNET.ora file.
2. Configure the LISTENER.ora file.
3. Configure the TNSNAMES.ora file.

Configure the SQLNET.ora file

To configure the SQLNET.ora file, complete the following steps.

Note You may have additional parameters in your configuration.

- Update the server SQLNET.ora file to include the following parameters.

Note UNIX paths are shown in the example. A Windows path example is C:\app\oracle\product\11.2.0\dbhome_1\owm\wallets\oracle.

```
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)

SQLNET.AUTHENTICATION_SERVICES= (BEQ, NTS)

SSL_VERSION = 3.0
SSL_SERVER_DN_MATCH = Yes
SSL_CLIENT_AUTHENTICATION = True
SSL_CIPHER_SUITES= (SSL_RSA_WITH_RC4_128_SHA)
```

```
WALLET_LOCATION =
 (SOURCE =
   (METHOD = FILE)
   (METHOD_DATA =
     (DIRECTORY = /u01/app/oracle/product/11.2.0/db_1/owm/wallets/oracle)
   )
 )
```

Configure the LISTENER.ora file

The LISTENER.ora file is only configured for the database server. To configure the LISTENER.ora file, complete the following steps.

1. In the LISTENER.ora file for the database server, change the protocol from TCP to TCPS, change the port number from 1521 to 2484, and add the WALLET_LOCATION.

Note UNIX paths are shown in the example. A Windows path example is C:\app\oracle\product\11.2.0\dbhome_1\owm\wallets\oracle.

```
SSL_VERSION = 3.0
SSL_CLIENT_AUTHENTICATION = True

LISTENER =
 (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCPS)(HOST = OracleDBServer)(PORT = 2484))
 )

SID_LIST_LISTENER =
 (SID_LIST =
   (SID_DESC =
       (GLOBAL_DBNAME = INOW6)
       (ORACLE_HOME = /u01/app/oracle/product/11.2.0/db_1)
       (SID_NAME = INOW6)
   )
 )

WALLET_LOCATION =
 (SOURCE =
   (METHOD = FILE)
   (METHOD_DATA =
     (DIRECTORY = /u01/app/oracle/product/11.2.0/db_1/owm/wallets/oracle)
   )
 )
```

2. Restart the Oracle listener.

Configure the TNSNAMES.ora file

To configure the TNSNAMES.ora file, complete the following steps.

1. In the TNSNAMES.ora file, change the protocol from TCP to TCPS and the port number from 1521 to 2484. Also, add the SECURITY parameter SSL_SERVER_CERT_DN and ensure that it matches the entire DN of the database server certificate. The DN can be found in the user certificate of the Oracle database server wallet.

```
INOW6 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCPS)(HOST = OracleDBServer)(PORT = 2484))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = INOW6)
    )
    (SECURITY=
      (SSL_SERVER_CERT_DN = "CN=<identifier>"))
  )
```

2. In a command prompt, enter the following commands to test the SQLNET configuration and make sure you can connect to the INOW6 database.

```
tnsping INOW6
sqlplus inuser/imagenow@INOW6
```

Configure the ImageNow ODBC Datasource

To configure the ImageNow ODBC Datasource, complete the following steps.

Situation	Steps
ODBC Data Source Administrator (Windows)	1. Place the client server wallet in your key store directory for ImageNow Server. Create a key store directory on your client server if one does not exist. **Example** C:\inserver6\bin64\dbconnector\keystore\ewallet.p12 2. On the **General** tab of the ODBC System DSN for ImageNow (ImageNow 6 Oracle Wire Protocol Driver), edit the following settings: • Host: Hostname or IP address of the Oracle DB Server • Port Number: The SSL Port number that was configured on the database server. The default is 2484. • SID: Oracle Service Name 3. On the **Security** tab, edit the following settings: • Authentication Method: 1 – Encrypt Password • Encryption Method: 3-SSL3 • Select Validate Server Certificate. • Trust Store: C:\inserver6\bin64\dbconnector\keystore\ewallet.p12 • Trust Store Password: (password to access the wallet) • Key Store: C:\inserver6\bin64\dbconnector\keystore\ewallet.p12 • Key Store Password: (password to access the wallet in the Key Store directory) • Host Name In Certificate: The Common Name entered when creating the database server user certificate. **Note** The server name must match the CN of the database server user certificate. 4. Click **Test Connect**. In the **Logon to Oracle Wire Protocol** dialog, enter your User Name and Password and click **OK**. **Connection Established!** appears in the **Test Connect** dialog box.

Situation	Steps
Command line (UNIX)	1. Place the client server wallet in your key store directory for the ImageNow Server. Create a key store directory on your client server if one does not exist. **Example** /opt/inserver/odbc/keystore/ewallet.p12 2. Open the /opt/inserver/etc/odbc.ini file and add the following security parameters to the **Oracle Wire Protocol** section. • loginTimeout: 15 • authenticationmethod: 1 • encryptionmethod: 3 • keystore: /opt/inserver/odbc/keystore/ewallet.p12 • keystorepassword: *<password>* • truststore: /opt/inserver/odbc/keystore/ewallet.p12 • truststorepassword: *<password>* • ValidateServerCertificate: 1 • HostNameInCertificate: The Common Name entered when creating the database server user certificate. **Note** The server name must match the CN of the database server certificate. 3. Run the following INTool command to test the connection. `intool --cmd get-db-version`

Appendix A: Entropy starvation considerations for Linux

Environmental randomness

The Progress DataDirect ODBC drivers, used for handling connections from Perceptive Content servers to Perceptive Content databases, utilize `/dev/random` for key generation during the SSL handshake for each connection to the database. For any Perceptive Content server, or any server or service that is running on a Linux operating system and is establishing SSL connections to the database, you may encounter slow or progressively slower server start-up times after a server reboot. This may also occur when restarting services after a period of time with low environmental randomness. Environmental randomness is collected from many different sources, such as hardware devices and network traffic. Under normal circumstances, these sources produce the randomness necessary to keep the entropy pool at sufficiently high levels to allow for quick connections to the database. When the entropy pool is empty, reads from `/dev/random` will block all referencing processes until additional environmental randomness is collected. If entropy levels are low and new entropy is not being introduced fast enough, delays may occur during start-up.

/dev/urandom

`/dev/urandom` is a non-blocking counterpart to `/dev/random`. `/dev/random` outputs true random noise, while `/dev/urandom` outputs true random noise until the system's entropy has been depleted. Upon depletion, `/dev/urandom` outputs pseudorandom noise based on previous system entropy.

Avoiding entropy starvation

If necessary, you can utilize various hardware random number generators to create entropy, or you can use a less secure method of running rng-tools to create pseudorandom entropy that is piped to `/dev/random`. For example, you can use `rngd -r /dev/urandom` to run the RNG service on `/dev/urandom`. The process immediately daemonizes and begins running in the background. Sufficient levels of entropy should then be generated to resolve the entropy starvation, causing any processes that are blocked due to low entropy in `/dev/random` to immediately start running again.

For instructions on configuring the carry over of seed data in the entropy pool across shut-downs and start-ups, refer to the urandom man pages on your server.

Checking entropy levels

For Linux, the current amount of entropy and the size of the Linux kernel entropy pool is available in `/proc/sys/kernel/random/` and can be displayed by using the `cat` command against the read-only file `/proc/sys/kernel/random/entropy_avail`.

Index