HITACHI
Inspire the Next

Pentaho Data Integration (PDI) with Oracle Wallet Security

HITACHI
Inspire the Next

Change log (if you want to use it):

Date	Version	Author	Changes

Contents

This page intentionally left blank.

Overview

This document is intended to provide best practices for setting up Pentaho Data Integration (PDI) to work with Oracle's security. Oracle provides single sign-on (SSO) using wallets, and allows packet encryption using Secure Sockets Layer (SSL) with a proprietary SSO key store.

Some of the topics discussed here include creating server-side and client-side wallets, wallets for password-less logins, simplifying wallet maintenance, and configuring PDI to use wallets. The document does not cover the installation of Oracle or PDI.

The intention of this document is to speak about topics generally; however, these are the specific versions covered here:

Software	Version(s)
Pentaho	6.x, 7.x, 8.0
Oracle Java	7, 8

The Components Reference in Pentaho Documentation has a complete list of supported software and hardware.

Before You Begin

Before beginning, use the following information to prepare for the procedures described in the main section of the document.

This document assumes that you have knowledge of PDI software and that you have already installed and set up PDI on your environment.

Use Case: Oracle in SSL Mode

Janice administers a database environment that requires security to prevent data snooping. She has set up her Oracle environment in SSL mode for this reason, and now wants to use Pentaho to connect to Oracle using its SSL feature.

Creating Oracle Wallets

You will need to use Oracle tools to create server-side and client-side Oracle wallets.

You can find details on these topics in the following sections:

- Creating Server-Side Wallets
- Creating Client-Side Wallets
- X.509 Certificate Exchange Between Server and Client

Step 1: Creating Server-Side Wallets

The following code will create the Oracle wallet and the X.509 certificate for SSL connection to encrypt the data. The encryption strength is set to 2048, which will require the unlimited strength of encryption for the Java modules (Java Cryptography Extension (JCE)). See the section on Installing JCE for the JRE.

The last statement listed below exports the SSL certificate for the server.

1. Create a server-side wallet to allow local login on the server:

```
orapki wallet create  -wallet "/home/oracle/svrwallet" \
-pwd WalletPasswd123 -auto_login_local
```

2. Add a server X.509 certificate to the wallet:

```
orapki wallet add     -wallet "/home/oracle/svrwallet" \
-pwd WalletPasswd123 -dn "CN=`hostname`" -keysize 2048 -self_signed -
validity 3650
```

3. Display the contents of the wallet created if it is successful:

```
orapki wallet display -wallet "/home/oracle/svrwallet" \
-pwd WalletPasswd123
```

4. You will then see:

```
Oracle PKI Tool : Version 11.2.0.1.0 - Production
Copyright (c) 2004, 2009, Oracle and/or its affiliates. All rights
reserved.

Requested Certificates:
User Certificates:
Subject: CN=centos.cluster.com
Trusted Certificates:
Subject: OU=Class 2 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
Subject: CN=client
Subject: OU=Secure Server Certification Authority,O=RSA Data Security\,
Inc.,C=US
```

```
Subject: CN=GTE CyberTrust Global Root,OU=GTE CyberTrust Solutions\,
Inc.,O=GTE Corp, C=US
Subject: CN=centos.cluster.com
Subject: OU=Class 3 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
Subject: OU=Class 1 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
```

5. Export the X.509 server certificate from the wallet

```
orapki wallet export  -wallet "/home/oracle/svrwallet" \
-pwd WalletPasswd123 -dn "CN=`hostname`" -cert /tmp/server-certificate.crt
```

Step 2: Creating Client-Side Wallets

We recommend having an Oracle database administrator (DBA) create the client-side wallet on the server, even though this step can be done on the client-side. This prevents the passwords from being compromised.

Make sure that the wallets are created by the DBA before distribution, since the exchange of X.509 certificates is required.

1. Create a client-side wallet to allow local login:

```
orapki wallet create  -wallet "/home/oracle/cliwallet" \
-pwd WalletPasswd123 -auto_login_local
```

2. Add an X.509 certificate to the client-side wallet:

```
orapki wallet add     -wallet "/home/oracle/cliwallet" \
-pwd WalletPasswd123 -dn "CN=client" -keysize 2048 -self_signed -validity
3650
```

3. Display wallet content to examine the certificate if successful:

```
orapki wallet display -wallet "/home/oracle/cliwallet" \
-pwd WalletPasswd123
```

4. You will then see:

```
Oracle PKI Tool : Version 11.2.0.1.0 - Production
Copyright (c) 2004, 2009, Oracle and/or its affiliates. All rights
reserved.

Requested Certificates:
User Certificates:
Subject: CN=centos.cluster.com
Trusted Certificates:
Subject: OU=Class 2 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
Subject: CN=client
```

```
Subject: OU=Secure Server Certification Authority,O=RSA Data Security\,
Inc.,C=US
Subject: CN=GTE CyberTrust Global Root,OU=GTE CyberTrust Solutions\,
Inc.,O=GTE Corp, C=US
Subject: CN=centos.cluster.com
Subject: OU=Class 3 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
Subject: OU=Class 1 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
```

5. Export the X.509 server certificate from the wallet to a temporary location:

```
orapki wallet export  -wallet "/home/oracle/cliwallet" \
-pwd WalletPasswd123 -dn "CN=client" -cert /tmp/client-certificate.crt
```

The code above will create two wallet-related files, `cwallet.sso` and `ewallet.p12`. `cwallet.sso` is the wallet manager, containing the X.509 certificates. `ewallet.p12` holds the private keys in PKCS #12 format.

Step 3: X.509 Certificate Exchange Between Server and Client

This final step is required so you will get an acknowledgement between the server and client when you connect to the Oracle database from any JDBC driver:

1. Import the X.509 client certificate into the server wallet:

```
orapki wallet add -wallet "/home/oracle/svrwallet" -trusted_cert -cert
/tmp/client-certificate.crt -pwd WalletPasswd123
```

2. Import the X.509 server certificate into the client wallet:

```
orapki wallet add -wallet "/home/oracle/cliwallet" -trusted_cert -cert
/tmp/server-certificate.crt -pwd WalletPasswd123
```

The client wallet is ready for distribution to those who need it, once the exchange is completed.

Oracle Net Listener Changes (Oracle DB Server)

This section describes how to configure Oracle Net Listener to accept client SSL connections.

We do not recommend including everything in the URL, because you may have multiple databases in the thin client.

1. The `listener.ora` file is in `<ORACLE_HOME>\network\admin`. Add the following code to the file to change the wallet location, the encryptions supported, and the additional SSL port.

```
SSL_CLIENT_AUTHENTICATION = FALSE
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE) (METHOD_DATA = (DIRECTORY = /home/oracle/svrwallet))
   )
SSL_VERSION = 1.0
SSL_CIPHER_SUITES=(SSL_RSA_WITH_AES_128_CBC_SHA,SSL_RSA_WITH_AES_256_CBC_SH
A)


LISTENER =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCPS)(HOST = 192.168.56.70)(PORT = 1523))
  )
```

2. Be sure to *remove* any non-SSL ports from the Oracle listener, such as the default port `1521` that is associated with TCP connection. The new port is `1523`, and will require an SSL TCP Port. Those are highlighted in red above.
3. You will need to restart the Oracle Listener in order to enable the newly configured port. The client wallet is now ready for use.
4. Copy the `cwallet.sso` and `ewallet.p12` files that were created onto the Oracle client computer.

Client-Side Configuration

Modify two files, `tnsnames.ora` and `sqlnet.ora`, in the directory that is defined by an environment variable such as `TNS_ADMIN`.

Install the Oracle Client Tools on any PC or server where the Oracle connection will initiate from.

In our example below, the client PC is a Windows PC with the Oracle client tools installed.

tnsnames.ora

In our example, the Transparent Network Substrate (TNS) alias is named `ORCLS`, uses SSL connectivity, and listens on port `1523`. The important sections are highlighted in red below.

For security, change from TCP to TCPS so that there is no ability to connect through port 1521.

1. Make sure that the TNS aliases are different.
2. Add the following code to `tnsnames.ora`:

```
# tnsnames.ora Network Configuration File:
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.56.70)(PORT
= 1521)))
    (CONNECT_DATA = (SID = orcl))
  )

ORCLS =
  (DESCRIPTION =
    (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCPS)(HOST = 192.168.56.70)(PORT
= 1523)))
    (CONNECT_DATA = (SID = orcl))
  )
```

sqlnet.ora

The important sections are highlighted in red in the following code. This code will be in `sqlnet.ora` by default, and the red text shows what to add or change. For example, normally you would see `TCP` in the file, so remove that and put in `TCPS`.

1. Make sure that Cipher Suites match what is allowed on the Oracle Listener that it supports.
2. Make sure that the directory points to the location where the wallet files have been copied.

```
SQLNET.AUTHENTICATION_SERVICES= (TCPS,NTS)
NAMES.DIRECTORY_PATH= (TCPS, TNSNAMES, EZCONNECT)

WALLET_LOCATION = (
  SOURCE =
    (METHOD = FILE)
```

```
    (METHOD_DATA = (DIRECTORY =
C:\app\dgodhia\product\11.2.0\clientwallet))
  )
SQLNET.WALLET_OVERRIDE = TRUE
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_CIPHER_SUITES =
(SSL_RSA_WITH_AES_128_CBC_SHA,SSL_RSA_WITH_AES_256_CBC_SHA)
```

Testing the Connection Using Wallet

To test the connection:

1. Make sure that connectivity to the database using SQL*Plus is successful, before proceeding to configuring PDI.
2. Open a DOS command prompt in your Windows environment.
3. Navigate to the directory where the SQL*Plus binary is in your local computer where you will run the PDI Client tool:

```
C:>
C:>cd C:\app\<username>\product\11.2.0\client_1
```

4. Make sure that the environment variable for `TNS_ADMIN` is set:

```
C:>SET TNS_ADMIN=<ORACLE_HOME>\network\admin
```

5. Run SQL*Plus command using the following connection string:

```
C:>sqlplus scott/tiger@orcls
SQL*Plus: Release 11.2.0.1.0 Production on Fri Aug 5 16:46:49
2016Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit
Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options
SQL>
```

Connection Troubleshooting Tips

When testing the command `sqlplus scott/tiger@orcls`:

- If you get this error, reset the wallet password:

```
## ORA-29106: Cannot import PKCS #12 wallet.
      orapki wallet change_pwd -wallet
"C:\app\dgodhia\product\11.2.0\clientwallet" -oldpwd WalletPasswd123 -
newpwd WalletPasswd234
```

- If you get this error, correct your SSL authentication setting:

```
## ORA-28860: Fatal SSL error
```

Also, check on the client side and make sure that the `sqlnet.ora` is set to the following:

```
SSL_CLIENT_AUTHENTICATION = TRUE
```

Preparing the Setup for PDI

PDI uses Karaf as a container for the PDI engine to run in. You can find details on these topics in the following sections:

- Installing JCE for the Java Runtime Environment (JRE)
- Client-Side Configuration and Modifications
- Installing the PKI-Related Files - Oracle Security Provider
- Configuring Karaf for Oracle Security Provider
- Starting Up Spoon
- Connecting to Oracle

Step 1: Installing JCE for the Java Runtime Environment (JRE)

For this document and examples, we are using JRE 1.7. Oracle uses unlimited strength encryption for the wallet, so we need to install the JCE package in the JRE.

1. Download Java Cryptography Extension (JCE) Unlimited Strength for 1.7 from Oracle.
2. Unzip and install `local_policy.jar` and `US_export_policy.jar` in the `<JRE_HOME>/lib/security` directory.

Step 2: Client-Side Configuration and Modifications

For this section, complete the following steps:

1. Make sure you have completed the client-side configuration.
2. Add four parameters that need to be added to `spoon.bat` or `spoon.sh` for Java to be able to recognize the Security Provider:
    a. Open the `set-pentaho-env.bat` or `set-pentaho-env.sh` file with any text editor depending on your working operating system and add the following lines:

```
set OPT="-Djavax.net.ssl.trustStore=C:\<wallet_dir_location>\cwallet.sso"
%OPT%
set OPT="-Djavax.net.ssl.trustStoreType=SSO" %OPT%
set OPT="-Djavax.net.ssl.keyStore=C:\<wallet_dir_location>\cwallet.sso"
%OPT%
set OPT="-Djavax.net.ssl.keyStoreType=SSO" %OPT%
set OPT="-Doracle.net.tns_admin=C:\<tns_admin_directory>" %OPT%
```

3. Make sure that Oracle's JDBC JAR file, such as `ojdbc6.jar`, is copied into the `<Pentaho home>/data-integration/lib` directory.

Step 3: Installing the PKI-Related Files – Oracle Security Provider

Install or copy these three files to the `<Pentaho home>/data-integration/lib` folder. The following files are required:

- `oraclepki.jar`
- `osdt_cert.jar`
- `osdt_core.jar`

Step 4: Configuring Karaf for Oracle Security Provider

Karaf maintains its own security realm for its container. For the self-managed container to recognize the Oracle Security Provider:

1. Open the `<pentaho_dir>\data-integration\system\karaf\etc\custom.properties` file with a text editor.
2. Add the following line at the end of the file, if it is not already there:

```
org.apache.karaf.security.providers=oracle.security.pki.OraclePKIProvider
```

Karaf Security in the Apache documentation has more detailed information.

Step 5: Starting Up Spoon

Spoon can be started once the startup script has been modified to use the Java System Environment (JSE) variables. This was accomplished in Step 2 previously.

If you start `spoon.bat` in console mode, you will be able to verify that the `keystore` is automatically recognized once Karaf is brought up. A sample console log will show the following:

```
*** Karaf Instance Number: 1 at /C:/Users/dgodhia/Desktop/Software/pentaho6
***
***   /data-integration/./system/karaf//data1
***
*** Karaf Port:8801
***
*** OSGI Service Port:9050
***
********************************************************************************
**
Jun 21, 2016 8:26:19 PM org.apache.karaf.main.Main$KarafLockCallback
lockAquired
INFO: Lock acquired. Setting startlevel to 100
trustStore is: C:\app\dgodhia\product\11.2.0\svrwallet\cwallet.sso
trustStore type is : SSO
trustStore provider is :
init truststore
adding as trusted cert:
.....
.....
```

```
adding as trusted cert:
  Subject: CN=GTE CyberTrust Global Root, OU="GTE CyberTrust Solutions,
Inc.", O=GTE Corporation, C=US
  Issuer:  CN=GTE CyberTrust Global Root, OU="GTE CyberTrust Solutions,
Inc.", O=GTE Corporation, C=US
  Algorithm: RSA; Serial number: 0x1a5
  Valid from Wed Aug 12 17:29:00 PDT 1998 until Mon Aug 13 16:59:00 PDT
2018
```

Step 6: Connecting to Oracle

Currently, we recommend using the Generic database connection type, due to a bug (PDI-15578).

Once this bug has been resolved, we recommend switching over to the Oracle connection type so you can make use of the more advanced capabilities and features definable in the JDBC Drivers Advanced Options.

For now, two workarounds have been established: using a generic database connection type or using a JNDI connection. For the generic database connection type:

1. In the **View** tab in Spoon, right-click **Database connections** and choose **New**.
2. Set the **Connection Type** to `Generic database`, which give you the flexibility to enter parameters for connectivity:
    a. **Custom Connection URL:** `jdbc:oracle:thin:@orcls`
    b. **Driver Name:** `oracle.jdbc.driver.OracleDriver`
    c. **User Name** and **Password:** set to the ones you have set up

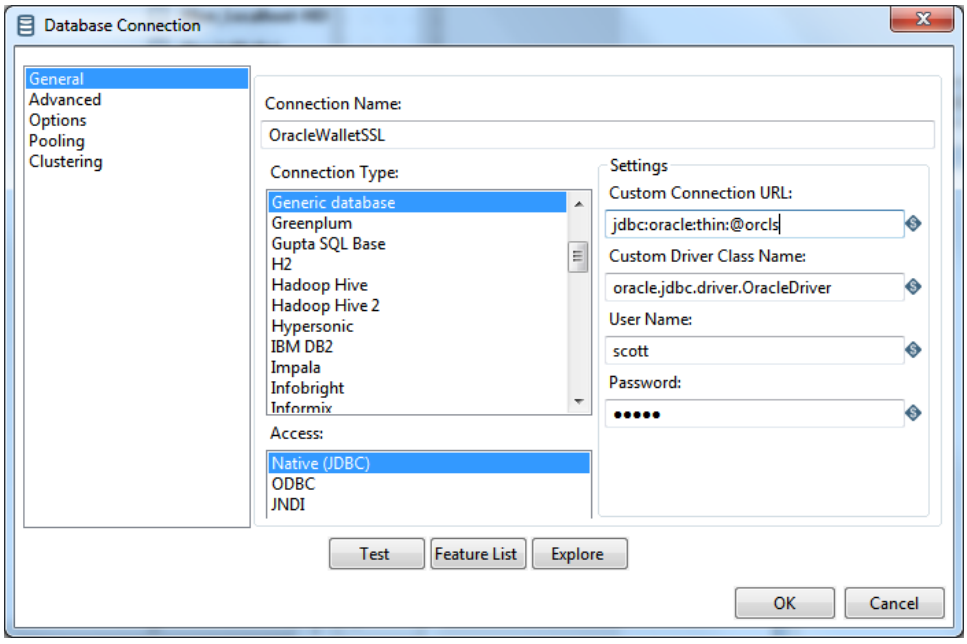

Figure 1: Database Connection Window

3. Click the **Test** button at the bottom of the **Database Connection** window and verify your connection.

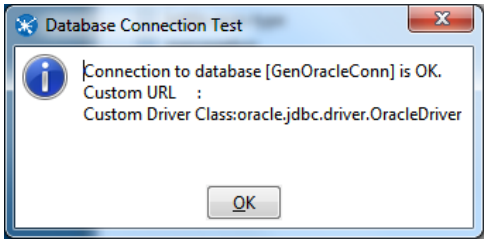

Figure 2: Database Connection Test

Passwordless Login into Oracle Database

One option you may want to make use of is passwordless login into Oracle. The wallet login will be created different from the Oracle wallet login. The passwordless (non-SSL) login wallet is created by Oracle DBAs and is used as a service account with limited or selected privileged access to Oracle objects.

To set up passwordless login into your Oracle database, complete the following:

- Creating the Wallet
- Creating a TNS Alias Entry
- Modifying `sqlnet.ora` to Use Wallet
- Modifying `spoon.bat` or `spoon.sh` for PDI
- Connecting to Oracle Using PDI
- Connecting to Oracle Using Java Naming and Directory Interface (JNDI)

Step 1: Creating the Wallet

For simplicity, the password that is used in this example wallet is `SecuredPassword123`. The following steps will create a wallet to be used on the client-side with a specific `tns` alias. (In this example, a separate `tns_admin` has been created specific to the user called `scott`.)

1. Create the wallet:

```
mkstore -wrl /home/scott/wallet -create
      Oracle Secret Store Tool : Version 11.2.0.1.0 - Production
Copyright (c) 2004, 2009, Oracle and/or its affiliates. All rights reserved.
Enter password: SecuredPassword123
Enter password again: SecuredPassword123
```

2. Create the Oracle security client connection credential:

```
mkstore -wrl /home/scott/wallet -createCredential remote_db scott tiger
Oracle Secret Store Tool : Version 11.2.0.1.0 - Production
Copyright (c) 2004, 2009, Oracle and/or its affiliates. All rights reserved.
Enter wallet password:
Create credential oracle.security.client.connect_string1
```

3. The following commands will allow you to list the contents of the wallet:

```
mkstore -wrl /home/scott/wallet -listCredential
Oracle Secret Store Tool : Version 11.2.0.1.0 - Production
Copyright (c) 2004, 2009, Oracle and/or its affiliates. All rights reserved.
Enter wallet password:
List credential (index: connect_string username)
1: remote_db scott
```

4. Test the connection: log in as `scott` on the server running Oracle. Log into Oracle using `sqlplus`:

```
[scott@centos ~]$ sqlplus /@remote_db
SQL*Plus: Release 11.2.0.1.0 Production on Fri Jul 15 18:12:18 2016
Copyright (c) 1982, 2009, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit
Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options

SQL> show user
scott
```

Step 2: Creating a TNS Alias Entry

The TNS alias created for the user called `scott/tiger` was `remote_db`. The file called `tnsnames.ora` will require an entry for this alias:

Edit the file named `tnsnames.ora` and add the new `tns alias` as follows:

```
REMOTE_DB =(DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.56.70)(PORT = 1521))
    )
    (CONNECT_DATA = (SID = orcl))
)
```

Step 3: Modifying *sqlnet.ora* to Use Wallet

1. Copy `cwallet.sso` and `cwallet.p12` from the server and place them where the service account can access them.
2. Make changes to the `sqlnet.ora` by adding the location of the wallet:

```
WALLET_LOCATION = (SOURCE =
    (METHOD = FILE)
    (METHOD_DATA = (DIRECTORY = C:\app\scott\product\11.2.0\pwdlesswallet))
  )
SQLNET.WALLET_OVERRIDE = TRUE
SSL_CLIENT_AUTHENTICATION = FALSE
```

Step 4: Modifying *spoon.bat* or *spoon.sh* for PDI

There are two parameters that need to be added to `spoon.bat` or `spoon.sh` for Java to be able to recognize the Security Provider. The parameter used is the wallet location in these examples.

Add the following lines to the `set-pentaho-env`:

```
set OPT="-Doracle.net.tns_admin=C:\<tns_admin_directory>" %OPT%
set OPT="-
Doracle.net.wallet_location=C:\app\dgodhia\product\11.2.0\pwdlesswallet"
%OPT%
```

Step 5: Connecting to Oracle Using PDI

Set up a new database connection in PDI.

Currently, we recommend using the Generic database connection type, due to a bug (PDI-15578).

Once this bug has been resolved, we recommend switching over to the Oracle connection type so you can make use of the more advanced capabilities and features definable in the JDBC Drivers Advanced Options.

For now, two workarounds have been established: using a generic database connection type or using a JNDI connection. For the generic database connection type, enter these parameters for connectivity. Note that the URL that uses `/@remote_db` for connectivity:

1. **Custom Connection URL:** `jdbc:oracle:thin:/@remote_db`
2. **Driver Name:** `oracle.jdbc.driver.OracleDriver`
3. **User Name:** `<left blank>`
4. **Password:** `<left blank>`

Alternate Step: Connecting to Oracle Using Java Naming and Directory Interface (JNDI)

An alternate method to connect to Oracle is to setup a JNDI connection.

This approach requires that you set certain properties in the `jdbc.properties` file found in the `<pentaho_directory>/data-integration/simple-jndi` directory:

```
File name:   jdbc.properties
Directory:   <pentaho_dir>\data-integration\simple-jndi

# Add the following properties at the end of the file

MyConn/type=javax.sql.DataSource
MyConn/driver=oracle.jdbc.driver.OracleDriver
MyConn/url=jdbc:oracle:thin:@orcls
MyConn/user=scott
MyConn/password=tiger
```

The JNDI name that has been created is called `MyConn`, in the example above.

You will be able to set up an Oracle connection using the Oracle connection type, once you have restarted PDI. The Oracle connection will use the JNDI access as shown:

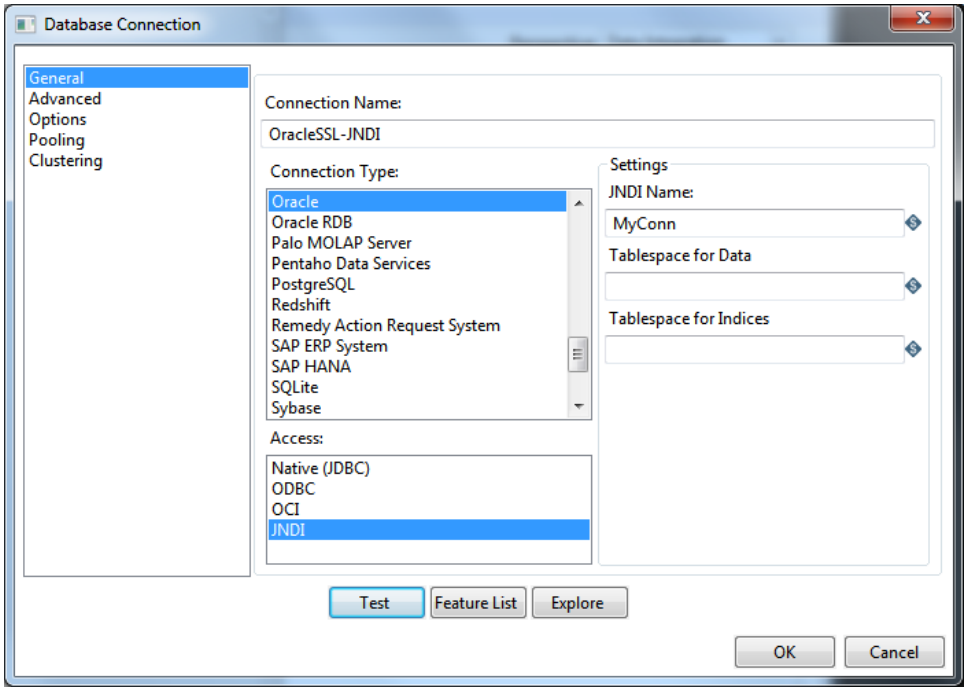

Figure 3: Setting Up Oracle Connection with JNDI

Related Information

Here are some links to information that you may find helpful while using this best practices document:

- Apache Karaf
- Oracle Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7 Download
- Pentaho
  - Bug: PDI-15578: Allow the creation of a JDBC URL per Oracle documentation
  - Components Reference

Finalization Checklist

This checklist is designed for you to use while you are thinking about how to implement Oracle Wallets for PDI. The Pentaho Enterprise Architecture Group is here to help you with any questions that arise during your implementation.

Name of the Project:_____

Date of the Review:_____

Name of the Reviewer:_____

Considerations	Response	Comments
Did you create server-side wallets?	YES_____ NO_____	
Did you create client-side wallets?	YES_____ NO_____	
Did you set up the X.509 certificate exchange?	YES_____ NO_____	
Did you install JCE for JRE?	YES_____ NO_____	
Did you install the PKI-related files?	YES_____ NO_____	
Did you modify `spoon.bat/sh` for PDI?	YES_____ NO_____	
Did you configure Karaf?	YES_____ NO_____	
Use PDI to connect to Oracle repo or with JNDI?	YES_____ NO_____	
Did you create a TNS alias entry?	YES_____ NO_____	
Did you modify `sqlnet.ora` to use Wallet?	YES_____ NO_____	