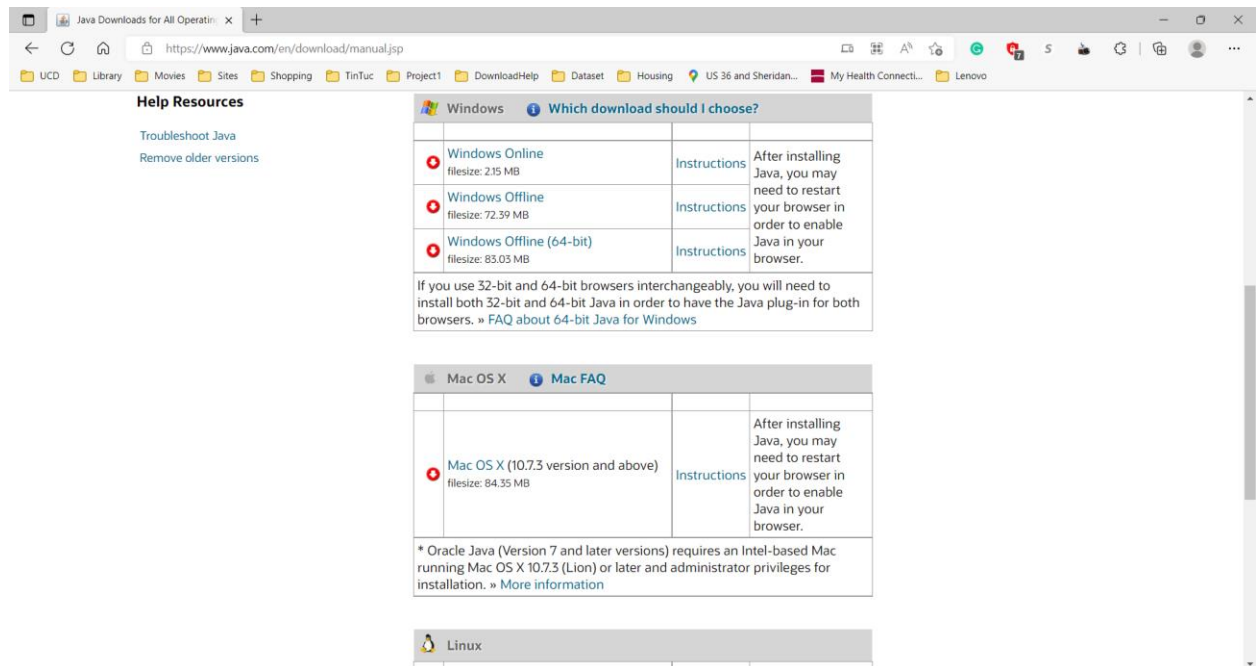


Configuring Pentaho Data Integration to use the Oracle Wallet

Introduction

Oracle Cloud databases use the Oracle wallet security. The Oracle Wallet contains cryptographic certificates and keys used to establish secure, encrypted communications between an Oracle client and the target Oracle database server. Pentaho Data Integration can be configured to use the Oracle Wallet to establish secure communications between PDI and an Oracle cloud server. This tutorial assumes you have the following components set up.

- An Oracle Cloud account with at least one Oracle database instance running. Please refer to Getting Started document on the course website for the steps to set up an Oracle Cloud account and Autonomous Database. Connectivity between a client such as Oracle SQL Developer and the autonomous Database should be tested and confirmed.
- Pentaho Data Integration installed on Windows or Mac OSX. This document contains brief installation instructions. The course website contains documents and links to more detailed instructions for installing the latest version of PDI.
- The steps in this document have been verified since January 2019 using Pentaho Data Integration 8.3 running on Java Development Kit 1.8 to the current versions of PDI and JDK. To download Java, visit [Java Downloads for All Operating Systems](#) and install the version for your operating system. For Windows users, make sure that the Java version matches the word size (64 or 32 bits) of your Windows system.



Installing the Oracle JDBC Drivers in Pentaho Data Integration

Pentaho Data Integration requires the Oracle JDBC Drivers to be installed to connect to any Oracle database. In addition to the main `ojdbc8.jar` file, several additional files that are part of the full JDBC distribution are also required. To get started visit the page for the Oracle Database JDBC driver and Companion Jars Downloads.

<https://www.oracle.com/database/technologies/appdev/jdbc-downloads.html>

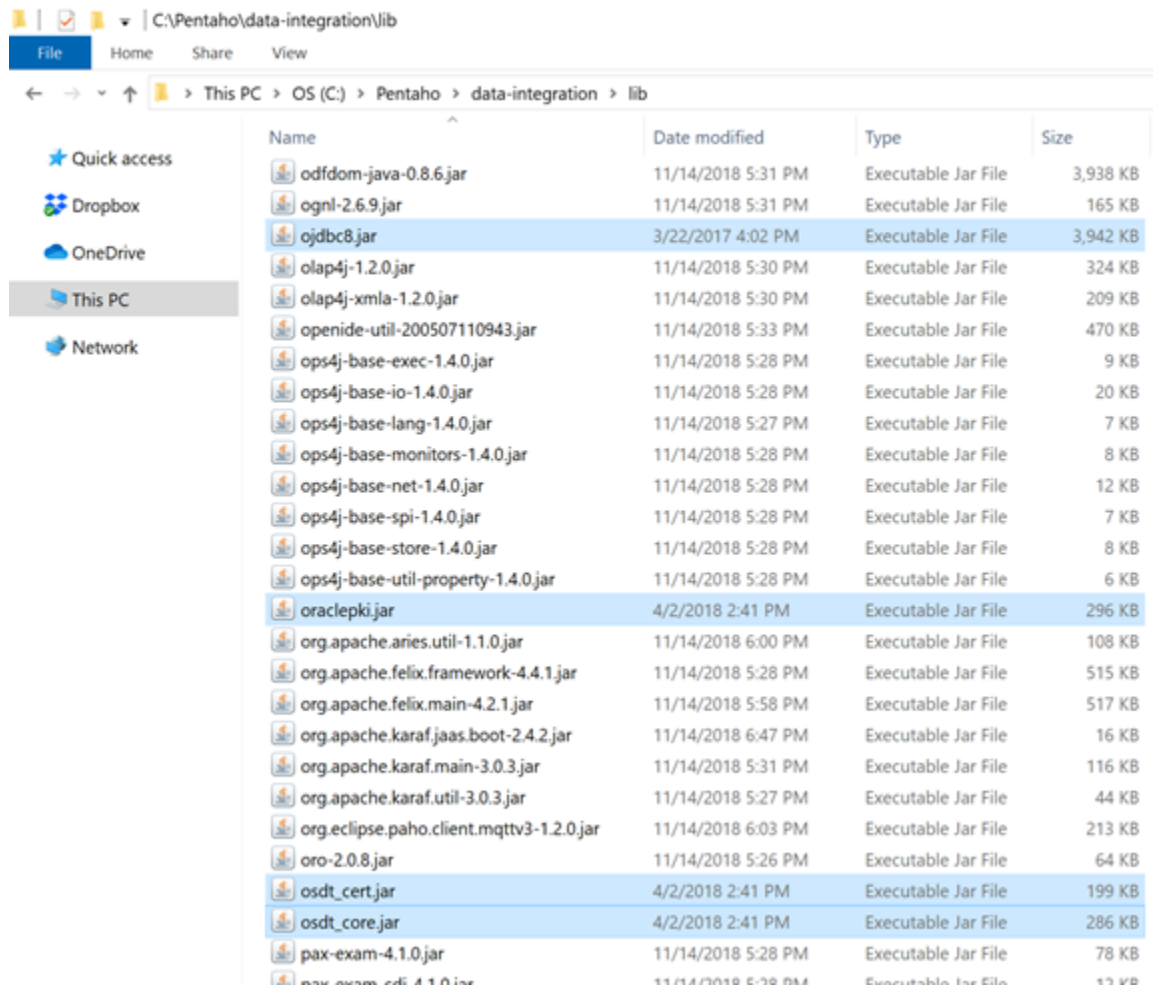
In May 2022, the list of files for Oracle Database 21c is shown in the following table.

Name	Download	JDK Supported	Description
Oracle JDBC driver	ojdbc11.jar	Implements JDBC 4.3 spec and certified with JDK11 and JDK17	<ol style="list-style-type: none"> Oracle JDBC driver except classes for NLS support in Oracle Object and Collection types. (5,173,200 bytes) - (SHA1: 3e588fed4618a39c7d655bfc1159ecb57fab217)
Oracle JDBC driver	ojdbc8.jar	Implements JDBC 4.2 spec and	<ol style="list-style-type: none"> Oracle JDBC driver except classes for NLS support in Oracle Object and Collection types. (5,080,094 bytes) - (SHA1:

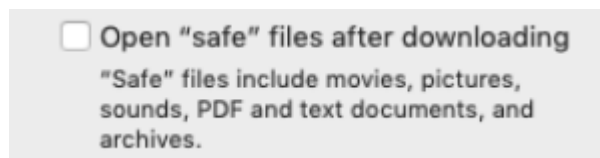
Name	Download	JDK Supported	Description
		certified with JDK8 and JDK11	eb13f0948ca8675f71d59b5e8a1129d4c6189e7a)
Universal Connection Pool - ucp11.jar	ucp11.jar	Certified with JDK11 and JDK17	<ol style="list-style-type: none"> 1. Universal Connection Pool to be used with ojdbc11.jar 2. (1,799,268 bytes) - (SHA1: 207a260b482423effd7b0bb8c04695932cf4992d)
Universal Connection Pool	ucp.jar	Certified with JDK8	<ol style="list-style-type: none"> 1. Universal Connection Pool to be used with ojdbc8.jar 2. (1,798,021 bytes) - (SHA1: b9910cb21986a1a2635a3c082f05dc1ba47b2fa3)
Zipped JDBC driver (ojdbc11.jar) and Companion Jars	ojdbc11-full.tar.gz	Certified with JDK11 and JDK17	<ol style="list-style-type: none"> 1. This archive contains ojdbc11.jar, ucp.jar, Reactive Streams Ingest (rsi.jar), companion jars¹, diagnosability jars², JDBC, UCP, RSI Javadoc, their Readmes, and Bugs-fixed-in-215.txt. Refer to README.txt in the zip for details. 2. (40,741,038 bytes) - (SHA1: ca27ad596f41a6421f9bafcf0ac161d30f06f63e)
Zipped JDBC driver (ojdbc8.jar) and Companion Jars	ojdbc8-full.tar.gz	Certified with JDK8 and JDK11	<ol style="list-style-type: none"> 1. This archive contains ojdbc8.jar, ucp.jar, Reactive Streams Ingest (rsi.jar), companion jars¹, diagnosability jars², JDBC, UCP, RSI Javadoc, their Readmes, and Bugs-fixed-in-215.txt. Refer to README.txt in the zip for details. 2. (40,251,929 bytes) - (SHA1: b9ca56002855b27f42f6c0f40245cdf731001945)

You need these files: ojdbc8.jar, oraclepki.jar, osdt_cert.jar, and osdt_core.jar. To download these files, download one of the tar files such as ojdbc8-full.tar.gz. Open the archive file to copy the files. Each of these files should be copied into the Pentaho data-integration\lib folder. As

always, you must restart Pentaho Data Integration so the new jar files are recognized.

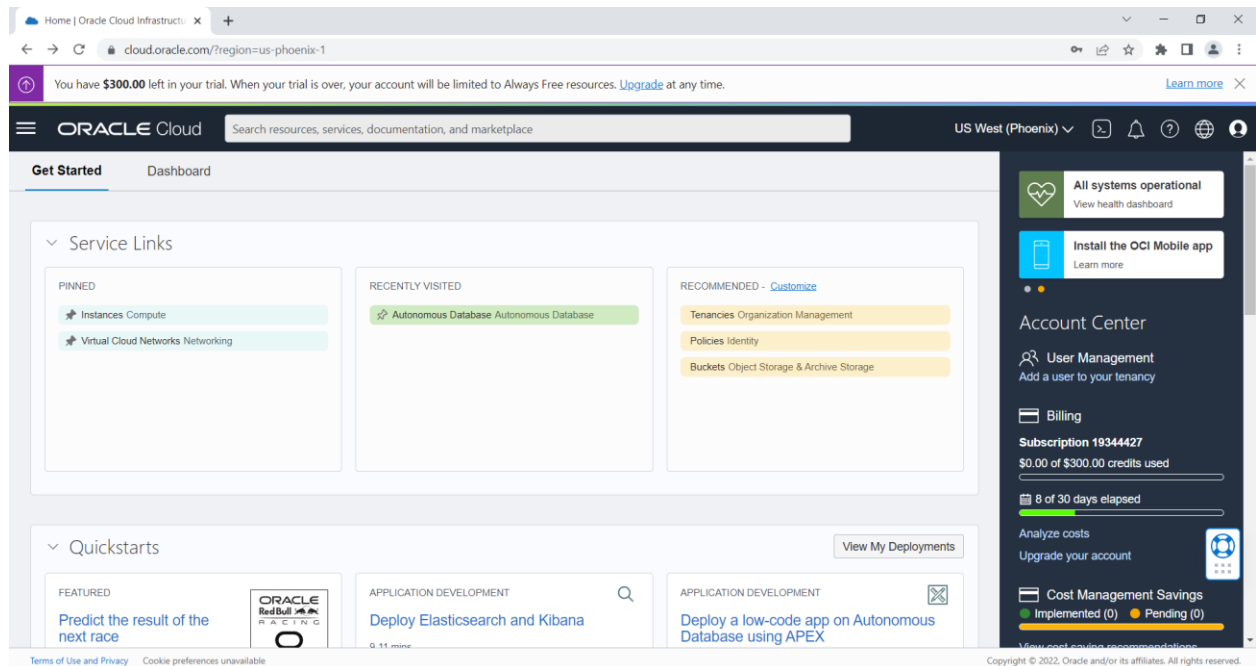


Note: If you are using Safari on Mac OSX, change the preferences in Safari to prevent “safe” files from being automatically opened after downloading. Safari > Preferences > Uncheck the Open “safe” files after downloading check box.

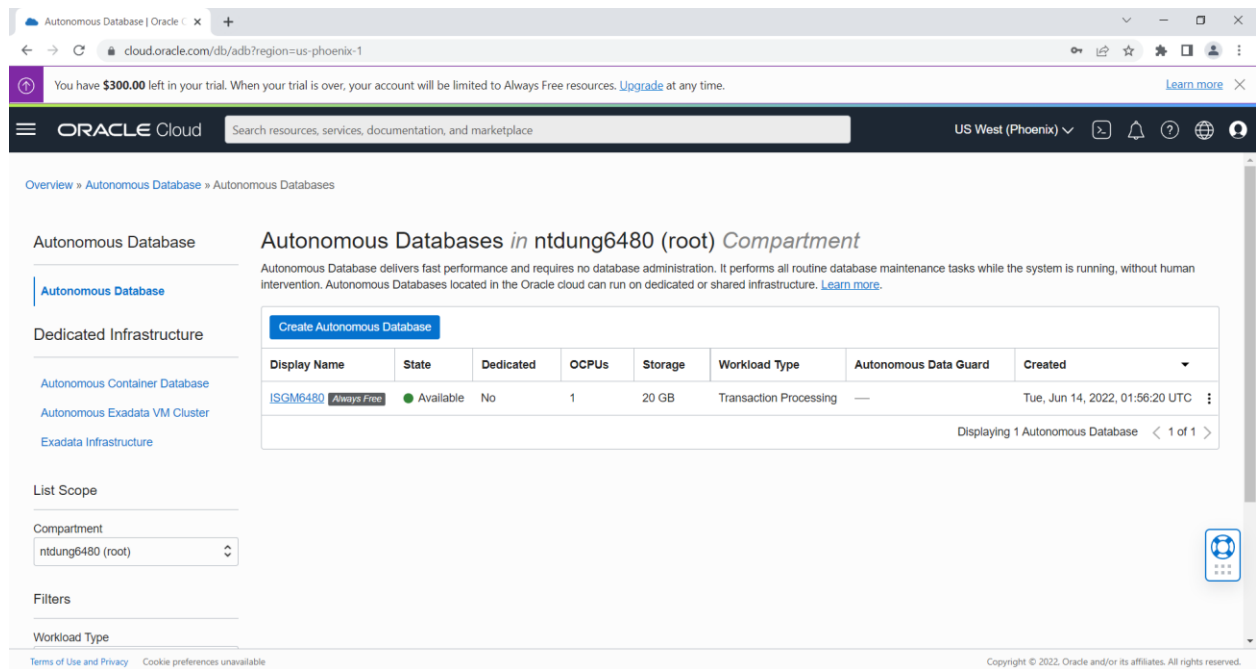


Downloading the Oracle Wallet from Oracle Cloud

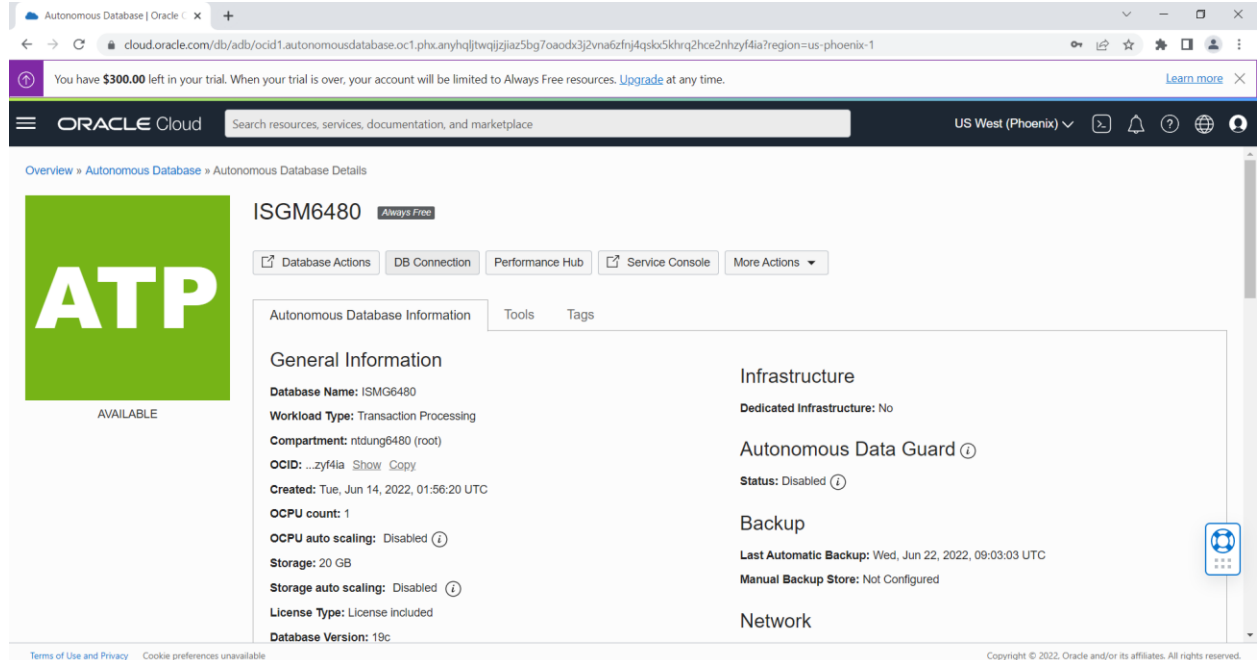
Log into your Oracle Cloud console and navigate to the Autonomous Database page. Click on the link under the **Recent Visited** to view your autonomous Database page.



Click on the database that you want to create a connection to.



When a new page is prompted, click on **DB Connection**



Autonomous Database | Oracle Cloud

You have \$300.00 left in your trial. When your trial is over, your account will be limited to Always Free resources. [Upgrade](#) at any time.

ORACLE Cloud

Overview » Autonomous Database » Autonomous Database Details

ISGM6480 Always Free

Database Actions DB Connection Performance Hub Service Console More Actions

Autonomous Database Information Tools Tags

General Information

Database Name: ISGM6480
Workload Type: Transaction Processing
Compartment: ntdung6480 (root)
OCID: ...zyf4ia [Show](#) [Copy](#)
Created: Tue, Jun 14, 2022, 01:56:20 UTC
OCPU count: 1
OCPU auto scaling: Disabled [i](#)
Storage: 20 GB
Storage auto scaling: Disabled [i](#)
License Type: License included
Database Version: 19c

Infrastructure

Dedicated Infrastructure: No

Autonomous Data Guard

Status: Disabled [i](#)

Backup

Last Automatic Backup: Wed, Jun 22, 2022, 09:03:03 UTC
Manual Backup Store: Not Configured

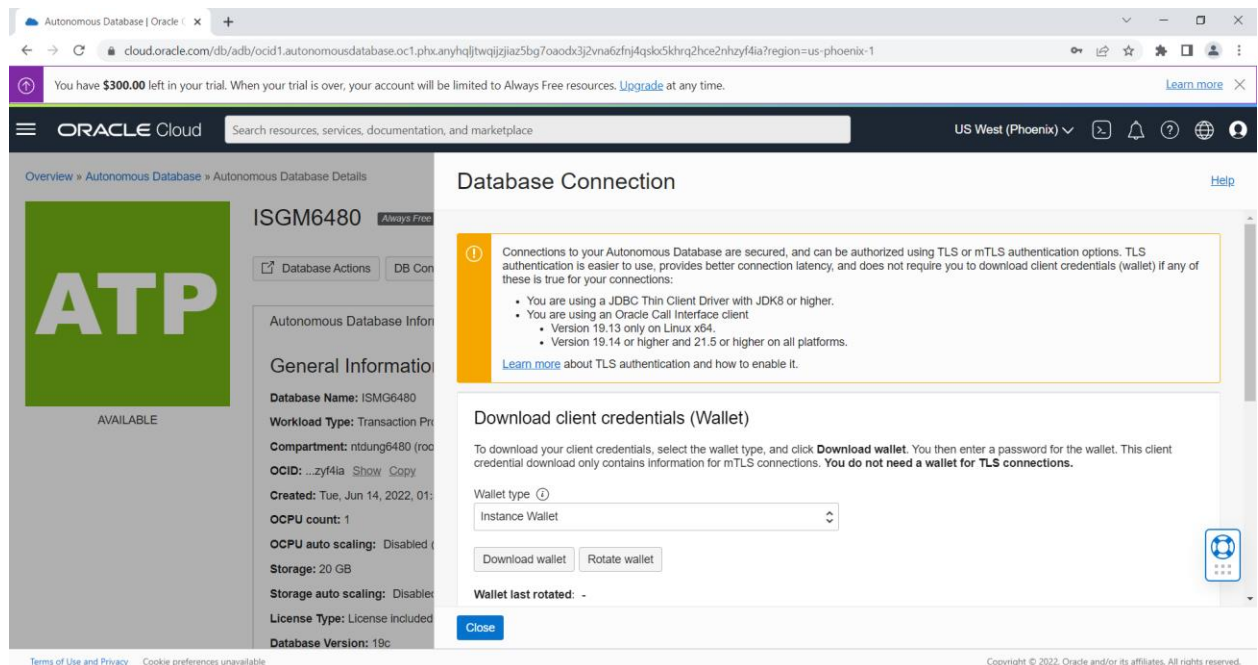
Network

ATP AVAILABLE

Terms of Use and Privacy Cookie preferences unavailable

Copyright © 2022, Oracle and/or its affiliates. All rights reserved.

When the Database Connection window appears, click on the **Download Wallet** button.



Autonomous Database | Oracle Cloud

You have \$300.00 left in your trial. When your trial is over, your account will be limited to Always Free resources. [Upgrade](#) at any time.

ORACLE Cloud

Overview » Autonomous Database » Autonomous Database Details

ISGM6480 Always Free

Database Actions DB Connection

Autonomous Database Information

General Information

Database Name: ISGM6480
Workload Type: Transaction Processing
Compartment: ntdung6480 (root)
OCID: ...zyf4ia [Show](#) [Copy](#)
Created: Tue, Jun 14, 2022, 01:56:20 UTC
OCPU count: 1
OCPU auto scaling: Disabled [i](#)
Storage: 20 GB
Storage auto scaling: Disabled [i](#)
License Type: License included
Database Version: 19c

Database Connection

Connections to your Autonomous Database are secured, and can be authorized using TLS or mTLS authentication options. TLS authentication is easier to use, provides better connection latency, and does not require you to download client credentials (wallet) if any of these is true for your connections:

- You are using a JDBC Thin Client Driver with JDK8 or higher.
- You are using an Oracle Call Interface client
 - Version 19.13 only on Linux x64.
 - Version 19.14 or higher and 21.5 or higher on all platforms.

[Learn more](#) about TLS authentication and how to enable it.

Download client credentials (Wallet)

To download your client credentials, select the wallet type, and click **Download wallet**. You then enter a password for the wallet. This client credential download only contains information for mTLS connections. **You do not need a wallet for TLS connections.**

Wallet type [i](#)

Instance Wallet

Download wallet Rotate wallet

Wallet last rotated: -

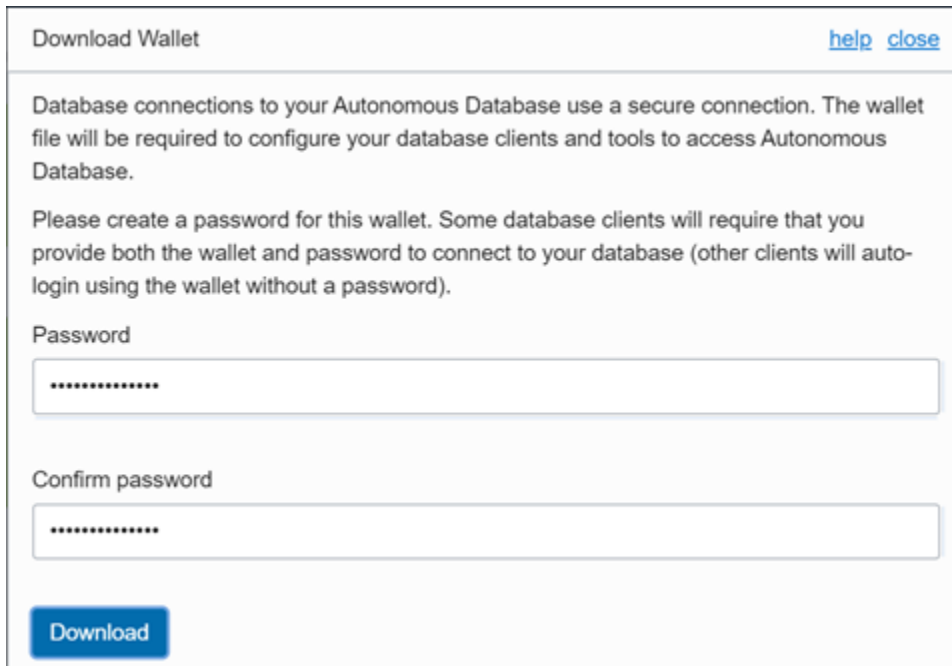
Close

ATP AVAILABLE

Terms of Use and Privacy Cookie preferences unavailable

Copyright © 2022, Oracle and/or its affiliates. All rights reserved.

Provide a matching password for the Wallet and then click the blue **Download** button.



Download Wallet [help](#) [close](#)

Database connections to your Autonomous Database use a secure connection. The wallet file will be required to configure your database clients and tools to access Autonomous Database.

Please create a password for this wallet. Some database clients will require that you provide both the wallet and password to connect to your database (other clients will auto-login using the wallet without a password).

Password

Confirm password

Download

Note that the wallet file name consists of the word “Wallet” with the name of the autonomous databases added to it. For this example, the file is: Wallet_ismg6480.zip.

Be sure to save your Wallet file in a safe, secure place.

Go back to the Database Details screen. Click on the **DB Connection** button again. Scroll down until you see the section titled **Connection Strings**. Note that a connection string corresponds to a specific connection type saved in the tnsnames.ora file that is included in the wallet. Make a note of these names such as the one with _HIGH at the end of it. These names should consist of the name of your database (without spaces) followed by either _HIGH, _MEDIUM or _LOW. In this example, the name of the database is “ISMG6480” so the first connection string is named: Ismg6480_HIGH. Write down or copy and save the entry under **TNS Name** for your

database.

The screenshot shows the Oracle Cloud console interface. On the left, the 'Autonomous Database Details' page for instance 'ISMG6480' is visible, showing a green 'ATP' status and 'AVAILABLE' state. The 'General Information' tab is active, displaying details like Database Name, Workload Type, and OCPU count. Overlaid on this is the 'Database Connection' dialog box. This dialog shows 'Connection Strings' for 'Mutual TLS' authentication. It contains a table with two columns: 'TNS Name' and 'Connection String'. Five entries are listed, all pointing to the same endpoint with different TNS names. A 'Close' button is at the bottom left of the dialog.

TNS Name	Connection String
ismg6480_high	...dwood City, ST=California, C=US*)]] Show Copy
ismg6480_low	...dwood City, ST=California, C=US*)]] Show Copy
ismg6480_medium	...dwood City, ST=California, C=US*)]] Show Copy
ismg6480_tp	...dwood City, ST=California, C=US*)]] Show Copy
ismg6480_tpurgent	...dwood City, ST=California, C=US*)]] Show Copy

Installing Pentaho Data Integration

To install Pentaho Data Integration, follow the steps in this subsection. It is highly recommended that you use the community edition from SourceForge as other course documents provide instructions that follow the community edition interface.

- The PDI version used in this document is 9.3.0. You should be able to use this version or a newer version to complete course exercises and assignments.
- You can find the community edition download for version 9.3.0 and other versions at <https://sourceforge.net/projects/pentaho/files/>.
- Unzip the downloaded zip file to C:\Pentaho\data-integration on Windows or /Users/<username>/Pentaho/data-integration on Mac OSX.
- Windows users should copy the folder data-integration to the folder C:\Program Files\Pentaho. Mac and Linux users may move the file to any folder.

- For Mac OS X users, the course website provides an additional document about installing PDI on the Mac OSX. The instructions in this document were gleaned from other websites as installation on the Mac OSX has not been verified by the instructor.

To ensure that the installation worked, you should launch Pentaho Data Integration.

- Run the file Spoon.bat by double clicking on it. You may want to create a shortcut to the spoon.bat file so starting data integration is easier. If you get a permission error or cannot execute the bat file, you should right click and select "Run as Administrator". For Mac and Linux users, run the Spoon.sh from terminal (./spoon.sh).
- If Pentaho Data Integration does not launch, the Module 5 contains documents with more detailed installation instructions for Windows and Mac OSX. Another document in Module 5 contains resolutions of common installation and execution problems with Pentaho Data Integration.
- If your installation works correctly, you will see the Welcome window.

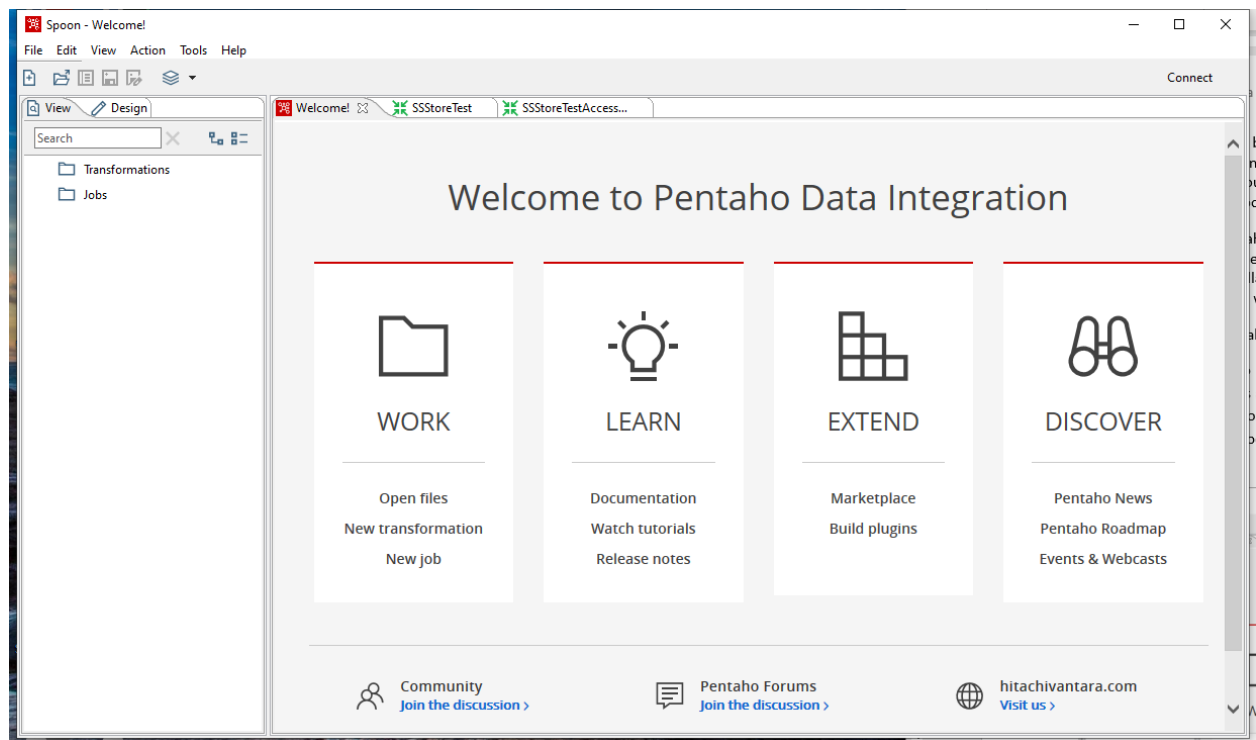


Figure 2: Spoon Opening Window

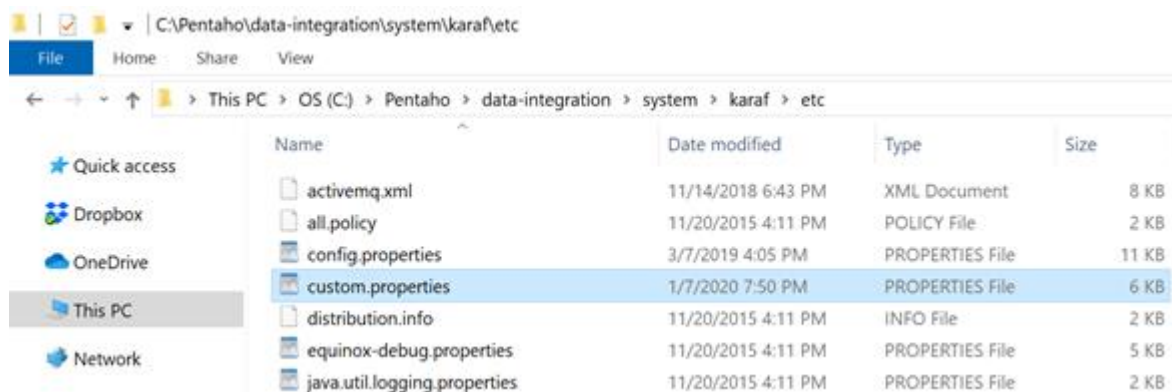
Integrating Oracle Cloud Wallet into Pentaho Data Integration

Several steps must be followed to direct Pentaho Data Integration to use the Oracle Wallet.

Ensure that Pentaho Data Integration is installed with the necessary Oracle JDBC Drivers. PDI should not be running at this point.

Edit the custom.properties file

Locate the custom.properties file found in the data-integration\system\karaf\etc\ folder.



Use Notepad (windows) or TextEdit (MacOSX) to edit the custom.properties file. Add the following line at the bottom of the file (if it is not already there).

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

Save the custom.properties file and exit the text editor.

Edit the Spoon.bat or spoon.sh files

For Windows installations edit the Spoon.bat file (Use Notepad or other Text editor). Scroll down to about line 112 just *before* the section with REM ** Run... **

Add the following lines (highlighted in yellow).

```
REM *****
REM ** Set java runtime options                **
REM ** Change 2048m to higher values in case you run out of memory **
REM ** or set the PENTAHO_DI_JAVA_OPTIONS environment variable    **
REM *****
```

```
if "%PENTAHO_DI_JAVA_OPTIONS%"==" " set PENTAHO_DI_JAVA_OPTIONS="-
Xms1024m" "-Xmx2048m" "-XX:MaxPermSize=256m"
```

```
set OPT=%OPT% %PENTAHO_DI_JAVA_OPTIONS% "-
```

```
@REM Java options to support Oracle Wallet secure connections
```

```
@REM Assumes that the cwallet.sso file is in data-integration\Wallet\ folder
```

```
set OPT="-Djavax.net.ssl.trustStore=%KETTLE_DIR%\Wallet\cwallet.sso" %OPT%
```

```
set OPT="-Djavax.net.ssl.trustStoreType=SSO" %OPT%
```

```
set OPT="-Djavax.net.ssl.keyStore=%KETTLE_DIR%\Wallet\cwallet.sso" %OPT%
```

```
set OPT="-Djavax.net.ssl.keyStoreType=SSO" %OPT%
```

```
set OPT="-Doracle.net.tns_admin=%KETTLE_DIR%\Wallet" %OPT%
```

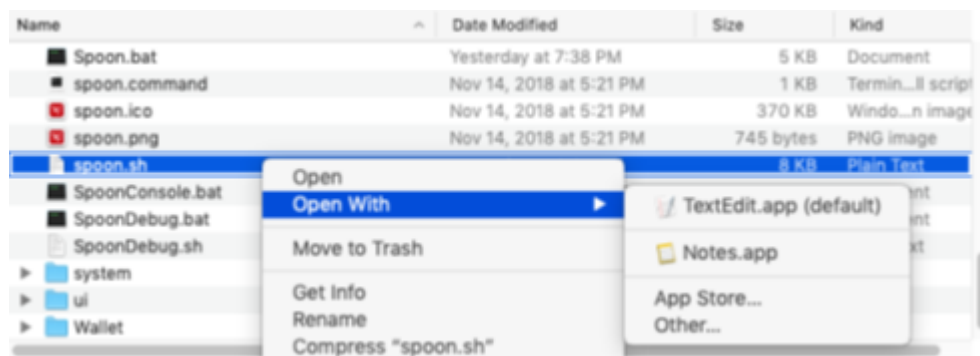
```
REM *****
```

```
REM ** Run... **
```

```
REM *****
```

Save the spoon.bat file and exit the text editor.

On Mac OSX, right click on the spoon.sh file, highlight **Open with** and then select **TextEdit app**.



Scroll down in the file until you locate the section with “# optional line for attaching a debugger”. The code to be added will go in the space just *before* this line.

Add the following lines (highlighted in blue in the picture below):

Java options to support Oracle Wallet secure connections

Assumes that the cwallet.sso file is in data-integration/Wallet/ folder

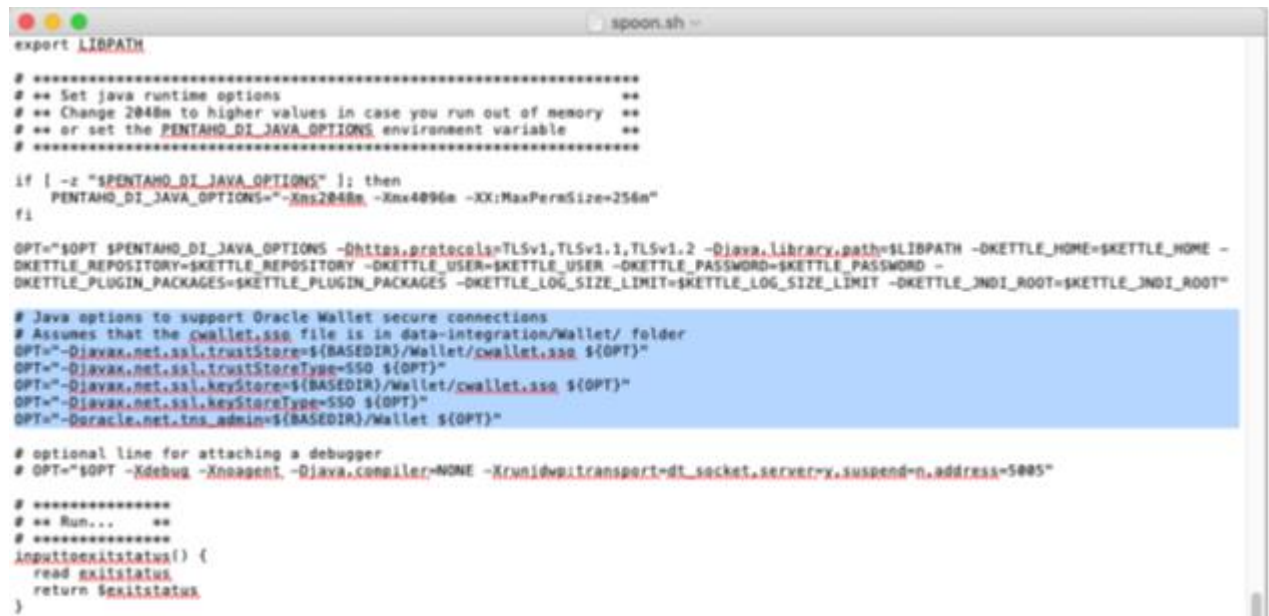
OPT="-Djavax.net.ssl.trustStore=\${BASEDIR}/Wallet/cwallet.sso \${OPT}"

OPT="-Djavax.net.ssl.trustStoreType=SSO \${OPT}"

OPT="-Djavax.net.ssl.keyStore=\${BASEDIR}/Wallet/cwallet.sso \${OPT}"

OPT="-Djavax.net.ssl.keyStoreType=SSO \${OPT}"

OPT="-Doracle.net.tns_admin=\${BASEDIR}/Wallet \${OPT}"



```
export LBPATH

# *****
# ** Set java runtime options **
# ** Change 2048m to higher values in case you run out of memory **
# ** or set the PENTAH0_D1_JAVA_OPTIONS environment variable **
# *****

if [ -z "$PENTAH0_D1_JAVA_OPTIONS" ]; then
    PENTAH0_D1_JAVA_OPTIONS="-Xms2048m -Xmx4096m -XX:MaxPermSize=256m"
fi

OPT="$OPT $PENTAH0_D1_JAVA_OPTIONS -Dhttps.protocols=TLSv1,TLSv1.1,TLSv1.2 -Djava.library.path=$LBPATH -DKETTLE_HOME=$KETTLE_HOME -DKETTLE_REPOSITORY=$KETTLE_REPOSITORY -DKETTLE_USER=$KETTLE_USER -DKETTLE_PASSWORD=$KETTLE_PASSWORD -DKETTLE_PLUGIN_PACKAGES=$KETTLE_PLUGIN_PACKAGES -DKETTLE_LOG_SIZE_LIMIT=$KETTLE_LOG_SIZE_LIMIT -DKETTLE_INDI_ROOT=$KETTLE_INDI_ROOT"

# Java options to support Oracle Wallet secure connections
# Assumes that the cwallet.sso file is in data-integration/Wallet/ folder
OPT="-Djavax.net.ssl.trustStore=${BASEDIR}/Wallet/cwallet.sso ${OPT}"
OPT="-Djavax.net.ssl.trustStoreType=SSO ${OPT}"
OPT="-Djavax.net.ssl.keyStore=${BASEDIR}/Wallet/cwallet.sso ${OPT}"
OPT="-Djavax.net.ssl.keyStoreType=SSO ${OPT}"
OPT="-Doracle.net.tns_admin=${BASEDIR}/Wallet ${OPT}"

# optional line for attaching a debugger
# OPT="$OPT -Xdebug -Xnoagent -Djava.compiler=NONE -Xrunidwpitransport=dt_socket,server=y,suspend=n,address=5005"

# *****
# ** Run... **
# *****
inputtoexitstatus() {
    read exitstatus
    return $exitstatus
}
```

Save the spoon.sh file and exit the TextEdit app.

Copy the Wallet Files to PDI

Make sure you have the Oracle wallet downloaded from the cloud (see prior page). For this example, the database is named “Ismg6480” and the wallet file name is: Wallet_Ismg6480.zip

Unzip the Wallet file and locate the following four files within:

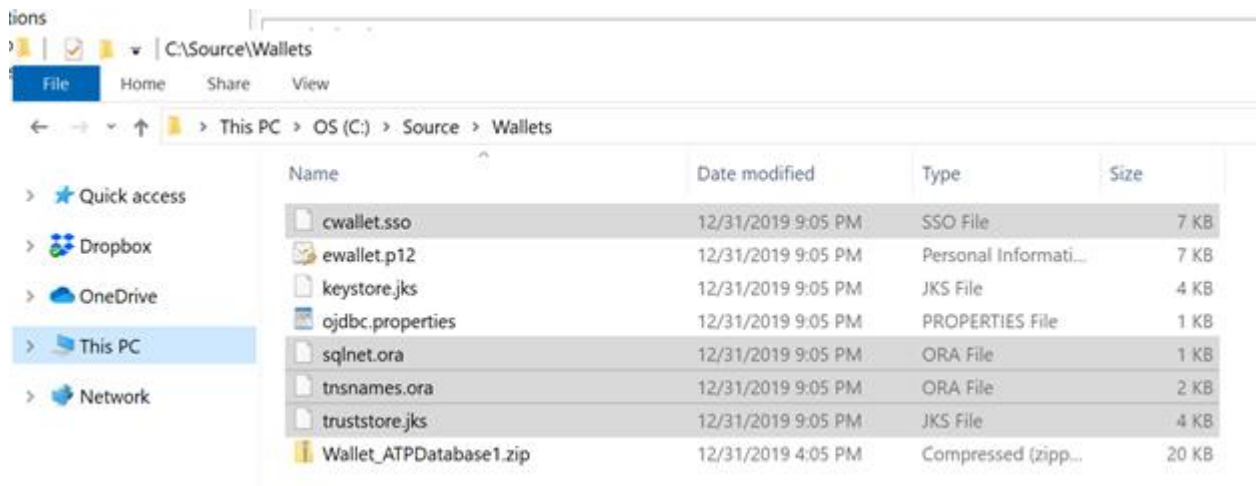
cwallet.sso

sqlnet.ora

tnsnames.ora

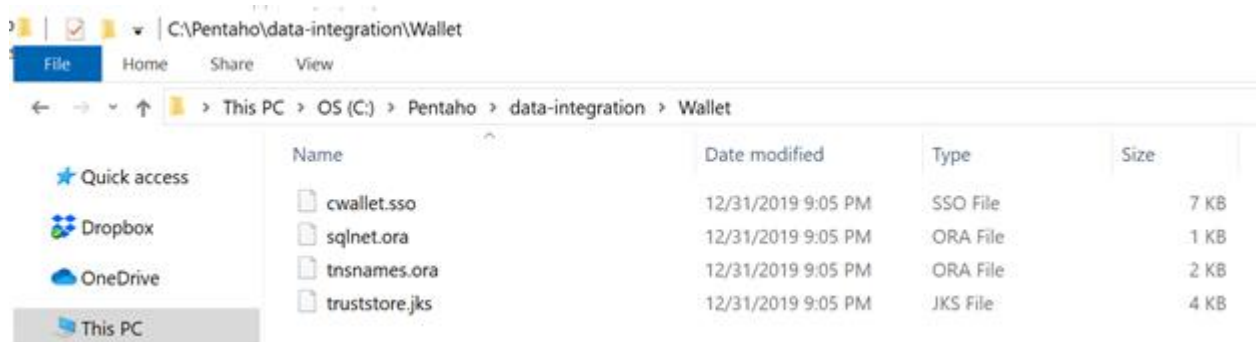
truststore.jks

These files are shown also in the figure below:

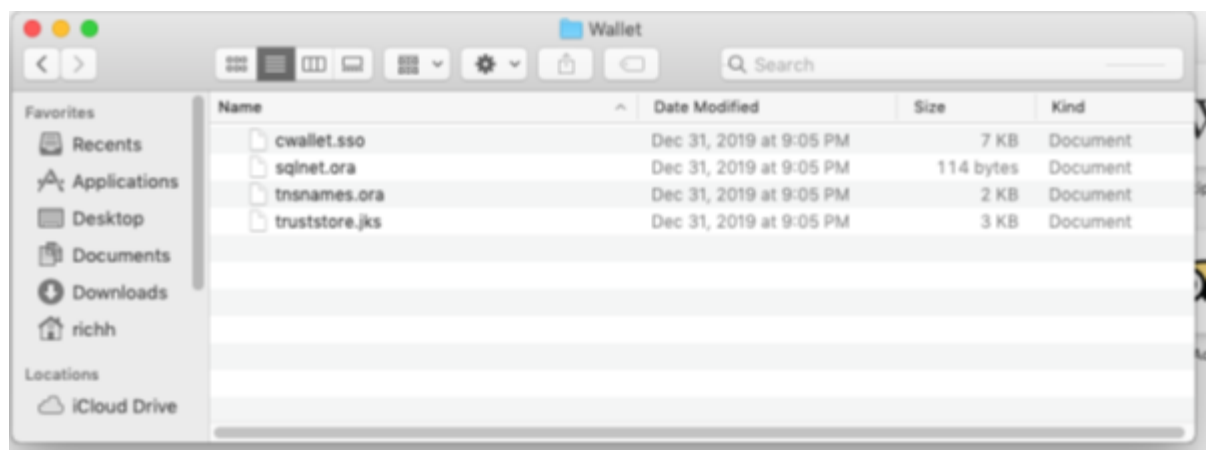


Highlight these four files and copy them.

Navigate to the Pentaho Data Integration folder. Create a sub-folder named “Wallet”. Paste in the four Oracle Wallet files into this Wallet folder as shown below (Windows):



On Mac OSX copy the four wallet files and paste them into /Users/<username>/Pentaho/data-integration/Wallet folder.



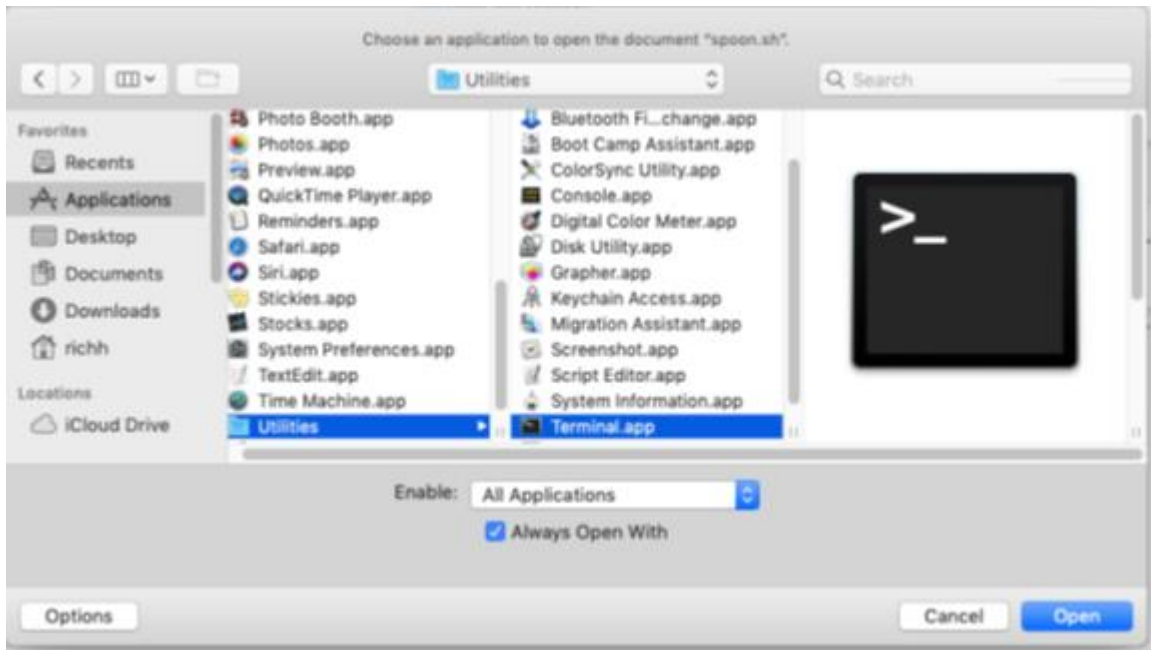
Running PDI

Once those files are in place, you can start Pentaho Data Integration normally by running the Spoon.bat file (or spoon.sh for Mac OSX users).

To run Pentaho Data Integration on Windows, double-click on the spoon.bat program. It might take several minutes for Pentaho to fully load.

To run Pentaho Data Integration on Mac OSX, right-click on the spoon.sh program, select Open with > Other... Change the filter to “All applications” and then within **Utilities** folder

choose **Terminal**. Then click the **Open** button.



Creating a Database Connection in Pentaho Data Integration

To use the Oracle Wallet in Pentaho Data Integration, a “Generic Database” type connection must be configured. The following steps outline this process.

When setting up a Table Input, Table Output or Dimension Lookup/Update step, Create a new database connection.

- Give the new connection a name of your choice.
- Change the **Connection Type** to Generic database.
- Change the **Dialect** to Oracle.
- Fill in the **Custom Connection URL** as follows:

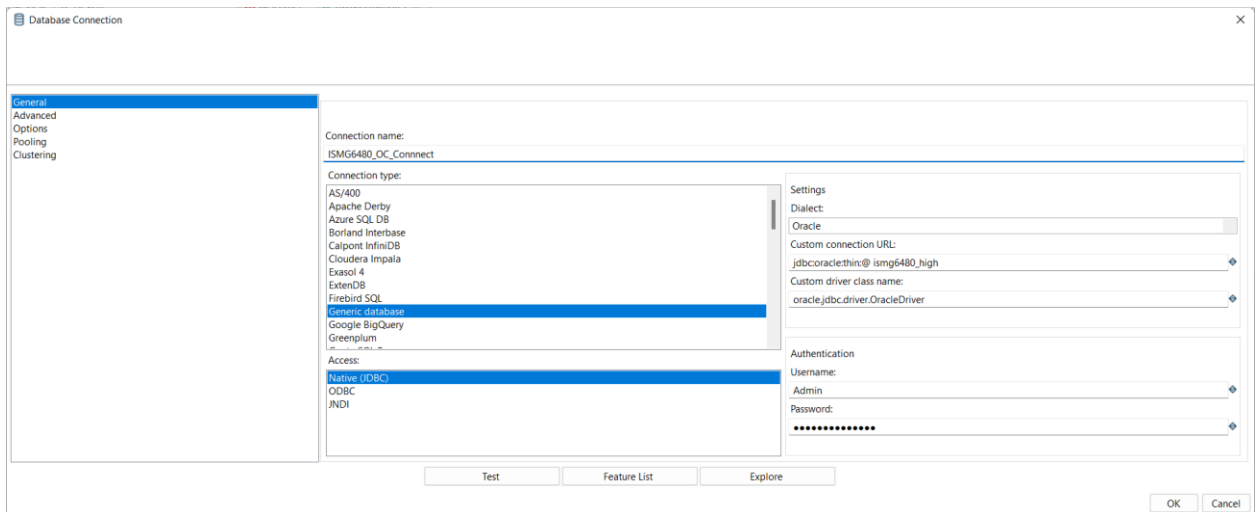
```
jdbc:oracle:thin:@ismg6480_high
```

Where ismg6480_high is the name of one of your connections to your Oracle Cloud database that you previously provided. The connection URL consists of your database name with _high attached to it. For example, if your database name is “ismg6480” then a

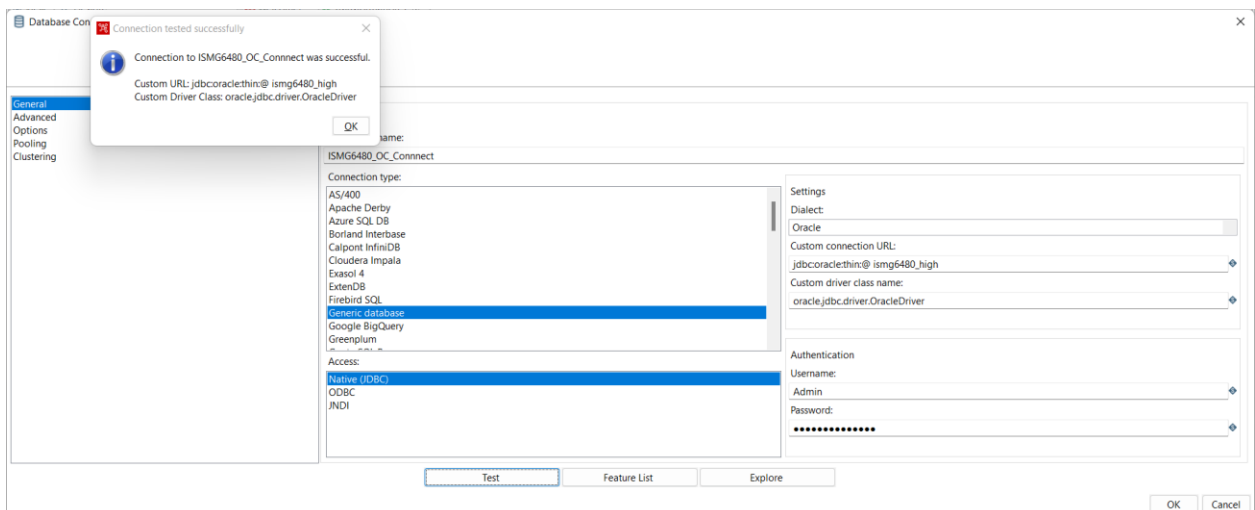
possible Connection URL is:

`jdbc:oracle:thin:@ ismg6480_high`

- Fill in the **Custom driver class name** as follows:
`oracle.jdbc.driver.OracleDriver`
- Fill in your username and password for your Oracle Cloud database.
- Click the **Test** button to test the connection.



The test result should appear successful as shown below.



Click the **OK** button to close the Connection test. Click the **OK** button to save this database connection.

Common Oracle Wallet Connection Errors and Issues

There are several common errors and issues that you may experience when using PDI and Oracle Cloud. The errors and solutions / resolutions are presented in the remainder of this section.

IO Error: could not resolve the connect identifier

If you receive an error such as “IO Error: could not resolve the connect identifier” then either your tnsnames.ora file cannot be found or the TNS Name provided does not match one of the connections. For example:

Error connecting to database [OracleCloud]

:org.pentaho.di.core.exception.KettleDatabaseException:

Error occurred while trying to connect to the database

Error connecting to database: (using class oracle.jdbc.driver.OracleDriver)

IO Error: could not resolve the connect identifier "ismg6480_hig"

Double check your tnsnames.ora file and make sure you are using a connection name that appears in the tnsnames.ora file. On Windows operating system use the NotePad program to open the tnsnames.ora file. On MacOSX, use the TextEdit program to open the tnsnames.ora file.

ORA-01017: invalid username/password; logon denied

If you receive an error such as “ORA-01017: invalid username/password; logon denied” it means the Oracle username or password supplied is invalid. For example:

Error connecting to database [OracleCloud]

:org.pentaho.di.core.exception.KettleDatabaseException:

Error occurred while trying to connect to the database

Error connecting to database: (using class oracle.jdbc.driver.OracleDriver)

ORA-01017: invalid username/password; logon denied

Check the username and password on the Oracle database instance to make sure it is correct.

Note that the Oracle database username and password will be different from your Oracle Cloud web site account. Typically for testing we use the ADMIN user and this password was set when the Autonomous Database or Autonomous Data Warehouse was created. For example on Page 4 of [this tutorial](#).

Make sure the password does not contain any of the following characters: @ / \ ?

It may be easiest for testing to use the ADMIN user with a relatively “plain” password such as Pw123Pw123Pw123.

Be certain to change the password to something more secure after testing is complete.

Driver class 'oracle.jdbc.driver.OracleDriver' could not be found

If you receive an error such as: “Driver class ‘oracle.jdbc.driver.OracleDriver’ could not be found”, make sure the JDBC Drivers have been copied into the data-integration\lib folder and PDI has been re-started. Also make sure to not place any spaces before the ‘o’ in oracle.jdbc.driver... For example:

Error connecting to database [OracleCloud]

:org.pentaho.di.core.exception.KettleDatabaseException:

Error occurred while trying to connect to the database

Driver class 'oracle.jdbc.driver.OracleDriver' could not be found, make sure the 'Generic database' driver (jar file) is installed. oracle.jdbc.driver.OracleDriver

ORA-12529 TNS:Connect request rejected based on current filtering rules

Of you receive an error such as: “ORA-12529 TNS:Connect request rejected based on current filtering rules” when connecting to the database, it may be the case that your Network security

settings for the autonomous database are preventing connections. The complete error message may appear as follows:

Error connecting to database [OracleCloud]

:org.pentaho.di.core.exception.KettleDatabaseException:

Error occurred while trying to connect to the database

Error connecting to database: (using class oracle.jdbc.driver.OracleDriver)

Listener refused the connection with the following error:

ORA-12529, TNS:connect request rejected based on current filtering rules

Sign in to the Oracle Cloud management interface and review the Autonomous Database

Information screen for your database. Check the **Access Type** and **Access Control List** to ensure remote login sessions are allowed from your location. You may need to restart the database

IO Error: Received fatal alert: handshake_failure, connect lapse 188 ms., Authentication lapse 0 ms.

This error can occur when there is a mis-match between the version of Java Development Kit and the Oracle server. It may be related to the code libraries used to secure the communications channel between client and server (e.g., TLS). Make sure you have the latest release of the

Oracle Java Development Kit 8. For example:

Error connecting to database [oracle_cloud_atp]

:org.pentaho.di.core.exception.KettleDatabaseException:

Error occurred while trying to connect to the database

Error connecting to database: (using class oracle.jdbc.driver.OracleDriver)

IO Error: Received fatal alert: handshake_failure, connect lapse 188 ms., Authentication lapse 0 ms.

org.pentaho.di.core.exception.KettleDatabaseException:

Error occurred while trying to connect to the database

Error connecting to database: (using class oracle.jdbc.driver.OracleDriver)

IO Error: Received fatal alert: handshake_failure, connect lapse 188 ms., Authentication lapse 0 ms.

IO Error: Inbound closed before receiving peer's close_notify

If you receive an error such as: “IO Error: Inbound closed before receiving peer’s close_notify” when running a transformation, it could be the case that you have too many open database connections to the Oracle server. This may be the case if you are using the “Always Free” services on Oracle Cloud. these services have a limited number of simultaneous connections. Close up any extra transformation tabs you have open. Exit Spoon and restart it. Only open up the transformation you have been working on and try running it again. For example in the PDI log,

2020/01/19 15:44:43 - Promotion Dim Lookup.0 - ERROR (version 8.2.0.0-342, build 8.2.0.0-342 from 2018-11-14 10.30.55 by buildguy) : An error occurred, processing will be stopped:

2020/01/19 15:44:43 - Promotion Dim Lookup.0 - Error occurred while trying to connect to the database

2020/01/19 15:44:43 - Promotion Dim Lookup.0 - IO Error: Inbound closed before receiving peer's close_notify: possible truncation attack?, Authentication lapse 0 ms.

In some rare cases, this could also be caused by a mismatch in the revision of the JDK being used. There were some incompatibilities introduced with the PKI support in the Java Development Kit version 1.8. Installing an updated 1.8 JDK seems to address these issues.

If the problem persists (and you are connecting to a non-production server), try restarting the Oracle autonomous database as this may free up any lingering connections.

ORA-00018: maximum number of sessions exceeded

The ORA-00018: maximum number of sessions exceeded error can appear if your database runs out of available connections. This can be an issue when working with the “Always Free” tier of the Oracle Autonomous database. The error may appear as follows:

org.pentaho.di.trans.steps.dimensionlookup.DimensionLookupMeta@75df4b1d -

ERROR (version 8.2.0.0-342, build 8.2.0.0-342 from 2018-11-14 10.30.55 by buildguy) :

A database error occurred:

org.pentaho.di.trans.steps.dimensionlookup.DimensionLookupMeta@75df4b1d -

Error occurred while trying to connect to the database

org.pentaho.di.trans.steps.dimensionlookup.DimensionLookupMeta@75df4b1d -

org.pentaho.di.trans.steps.dimensionlookup.DimensionLookupMeta@75df4b1d -

Error connecting to database: (using class oracle.jdbc.driver.OracleDriver)

org.pentaho.di.trans.steps.dimensionlookup.DimensionLookupMeta@75df4b1d -

ORA-00018: maximum number of sessions exceeded

Close up any extra transformation tabs you have open. Exit Spoon and restart it. Only open the transformation you have been working on and try running it again.

If the problem persists (and you are connecting to a non-production server), try restarting the Oracle autonomous database as this may free up any lingering connections.

Incorrect Data Types when creating tables

PDI should use the appropriate Oracle data types (INTEGER, NUMBER, DATE, VARCHAR) when creating tables through Dimension Lookup/Update or Table Output steps. If you see the use of datatypes such as BIGSERIAL or other non-Oracle data types, try the following:

- Go back to your source step such as CSV reading step and un-check the box for “Lazy Conversion”
- Return to your Dimension Lookup/Update or Table Output step and Edit the Database Connection. Make certain the Dialect is set to Oracle.

Click the TEST button to make sure the credentials are still working. Close up the Database connection dialog box.

- Click on the SQL button and check the proposed SQL CREATE TABLE statement to make sure the data types are now correct.
- If the data types are still not correct, try restarting PDI and follow the above steps again.

Slow Response when using the PDI Database Explorer with Oracle Cloud

You may notice Pentaho Data Integration will take a long time (and may crash) when using the Tools > Database > Explore feature to explore an Oracle Cloud database. The main reason is that most Oracle Cloud instances have many built-in database schemas that the PDI Database Explorer attempts to read.

Instead of using PDI's Database Explorer, use Oracle's SQL Developer tool to explore your schema, drop old tables and query data to see how transformations are working. Oracle Cloud has a built-in SQL Developer web application that is easy to use for this purpose.

Invalid Cache with Dimension Lookup/Update Step caused by Error serializing row to byte array

While updating data using a Dimension Lookup/Update step, you may encounter an error relating to the technical key (surrogate key) and Error serializing row to byte array. One example of this error would be:

Dimension lookup/update.0 - ERROR (version 8.2.0.0-342, build 8.2.0.0-342 from 2018-11-14 10.30.55 by buildguy) : Unexpected error

Dimension lookup/update.0 - ERROR (version 8.2.0.0-342, build 8.2.0.0-342 from 2018-11-14 10.30.55 by buildguy) : java.lang.RuntimeException:

Error serializing row to byte array

Dimension lookup/update.0 - at

org.pentaho.di.core.row.RowMeta.extractData(RowMeta.java:1134)

Dimension lookup/update.0 - at

org.pentaho.di.trans.steps.dimensionlookup.DimensionLookup.addToCache(DimensionLookup.java:1522)

Dimension lookup/update.0 - at

org.pentaho.di.trans.steps.dimensionlookup.DimensionLookup.lookupValues(DimensionLookup.java:754)

Dimension lookup/update.0 - at

org.pentaho.di.trans.steps.dimensionlookup.DimensionLookup.processRow(DimensionLookup.java:232)

Dimension lookup/update.0 - at org.pentaho.di.trans.step.RunThread.run(RunThread.java:62)

Dimension lookup/update.0 - at java.lang.Thread.run(Thread.java:748)

Dimension lookup/update.0 - Caused by: java.lang.RuntimeException: CUSTOMER_DIM_ID
BigNumber(38) :

There was a data type error: the data type of java.lang.Long object
[320] does not correspond to value meta [BigNumber(38)]

This error may be caused by an incorrect technical key value in the cache or the inability of the Dimension Lookup/Update step to access the cache of unique technical key values.

To address this issue, re-try the transformation with the **Enable the cache** option turned off in the Dimension Lookup/Update step. You may also wish to clear the cache by clicking on the **SQL** button and then click the **Clear Cache** button.