Hive-ODCI- Install Guide

Hive-ODCI is an Oracle Data Cartridge Interface for dynamically accessing Hadoop/Hive data-stores through an Oracle 12c database. In other words Hive-ODCI makes Hadoop/Hive tables accessible as first-class, native, objects directly using PL/SQL, SQL, VIEWS, DML, DDL, etc.... in an Oracle 12c database.

Author

Metasystem Technologies Inc. (MTI) www.mtihq.com

Nicholas Van Wyen nvanwyen@mtihq.com

License

Pre-installation

It may be necessary to perform some pre-installtion steps before getting started.

Oracle JVM Version

Check that the Oracle JVM Version installed is Java 7 (1.7.x) before attempt to install Hive-ODCI. Using SQL*Plus, run the following query in the database and make sure you have a similar output ...

```
SQL> select dbms_java.get_jdk_version from dual;

GET_JDK_VERSION
------------------
1.7.0_51
```

If your current Java is still at 1.6.x then you will need to update the JVM. You can use the distribution provided bash shell script `jdbc/update_javavm.sh` in the following manner. On the Oracle Database Server, run the script like ...

```
jdbc/update_javavm.sh 7
```

This will shutdown the database, run the Oracle provided `$ORACLE_HOME/javavm/install/update_javavm_binaries.pl` script and (re)link the Oracle binaries in the `$ORACLE_HOME/rdbms/lib` directory using the command `make -f ins_rdbms.mk ioracle`.

> Note: the `/etc/oratab` file must be set for the `$ORACLE_SID` to be auto start/stop through `dbstart` and `dbshut`. If this is not the case, the update will fail.

Important: If this is a RAC installation, then you will need to run `$ORACLE_HOME/javavm/install/update_javavm_binaries.pl` and `make -f ins_rdbms.mk ioracle` in `$ORACLE_HOME/rdbms/lib` on each Node in the cluster

Java Policy Files

Depending on your intended use, you may need to use the (JCE) Unlimited Strength Jurisdiction Policy Files. This is certainly the case if you intended to use Kerberos Authentication with a high encryption algorithm, such as AES-256.

You can update the Policy file from the `./policy` directory using the `set_policies.sh` bash script.

> Note: the Java 7 JCE Unlimited Policy JAR files are already provided with the distribution. *US_export_policy.jar* local_policy.jar

The script will make sure the JCE policy file in the `./policy` directory have not already been installed. It will backup the existing policy files the `$ORACLE_HOME/javavm/jdk/jdk7/lib/security/` directory and then copy the Unlimited JCE files.

```
policy/set_policies.sh 7
```

> Note: There is no need to bounce the database after setting these files.

Important: If this is a RAC installation, then you will need to update the files on each Node in the cluster.

Installation

The scripts are intended for Linux distributions only, and tested on RHEL 6. If your target installation is Windows, then you will need to perform additional steps prior to beginning. Please see Windows section.

After downloading the distribution and check sum file, make the integrity of the download is verified first. For example,

```
$ ls -l hive-odci-v0.1.5.19.sha hive-odci-v0.1.5.19.tgz
-rw-r--r-- 1 user mti      66 2016 hive-odci-v0.1.5.19.sha
-rw-r--r-- 1 user mti 1818992 2016 hive-odci-v0.1.5.19.tgz

$ sha1sum -c hive-odci-v0.1.5.19.sha
hive-odci-v0.1.5.19.tgz: OK
```

Extract the contents of the distribution file.

```
$ tar -xvf hive-odci-v0.1.5.19.tgz
LICENSE
README
VERSION
source/
...
```

Go the `./source` directory of the install home and using the SYSDBA account run the `install_hive.sql` script using SQL*Plus.

```
$ sqlplus "/ as sysdba" @install_hive.sql
```

A log file will be created in the current directory in the format `<sid>_remove_hive_odci.<date_time>.log`. Please check this log file for errors before proceeding. The end of the log should look similar to the following.

```
... show post installation object errors

no rows selected

Installation successful

PL/SQL procedure successfully completed.
```

> Note: If the installation was unsuccessful, remove the installation using the `remove_hive.sql` as shown in the Removal section. Fix the errors and try again

If you are installing on a Linux server, then the installation should automatically continue to the next steps of building and loading the Hive-ODCI jar file.

```
!( ./java/compile.sh )
Using Java compiler Version: 1.7.0_51
Compiling log.java ... OK
Compiling dbms_types.java ... OK
Compiling hive_exception.java ... OK
Compiling hive_parameter.java ... OK
Compiling hive_properties.java ... OK
Compiling callback_handler.java ... OK
Compiling hive_session.java ... OK
Compiling hive_bind.java ... OK
Compiling hive_bindings.java ... OK
Compiling hive_connection.java ... OK
Compiling hive_context.java ... OK
Compiling hive_manager.java ... OK
Compiling hive_attribute.java ... OK
Compiling hive.java ... OK
Building JAR hive-odci.jar ... done
```

> Note: A patch may be required if you are getting `O/S Message: No child processes`, see MOS: Doc ID 2021977.1, apply `Patch 19033356. SQLPLUS WHENEVER OSERROR FAILS REGARDLESS OF OS COMMAND RESULT` to resolve this error. If you do not wish to apply the patch simply, run the ./java/compile.sh script manually to build the `jdbc/hive-odci.jar` file yourself.

Finally the installtion will attempt to load the driver JAR files found in the `./jdbc/` directory, including the one built above. So if you have already copied the JDBC driver of your choice to that directory or you are using the one provided, the installation will load those classes.

Load the Driver

If the PL/SQL object installation was successful (as show above), then it's time to install the JDBC Driver of your choice in the database.

Copy all the JAR files needed for your Driver into the `./jdbc` directory. You may want to remove the existing `jdbc/hive.jar` file which came with the distribution.

Then run the load script `jdbc/load-jdbc.sh`. It will prompt you for a User/Password if one was not provided on the command line. if you are using the SYSDBA account SYS locally on your Linux machine, then the password will be ignored by the utility, so fell free to use anything.

```
jdbc/load-jdbc.sh "sys" "xxx"
```

At the end of the load operation, the log output should have `0 errors` reported, for example ...

```
...
Classes skipped: 0
Synonyms Created: 0
Errors: 0
```

> Note: While the Hive-ODCI should be able to work with any JDBC driver, it has been tested with the following: * CDH 6 Apache Hive JDBC Standalone * MTI CDH 5, JDBC Thin Driver * Progress Data Direct JDBC Hive Driver

Windows

If you are installing the distribution on or from Windows platform you must do the following pre-steps before running the `install_hive.sql` script. Also, please forgive me as I'm not a "Windows guy", so the commands and instructions are from a lifetime ago and may not be exactly correct.

Rename and modify the `hive.par.sql.in` file

The `hive.par.sql.in` file is an input file modified with the current version information by the ver bash shell script.

Copy or move the file, to get rid of the `.in` extension, for example ...

```
copy /b hive.par.sql.in hive.par.sql
```

The `install_hive.sql` script looks for the file name `hive.par.sql`, during the installation and if it's missing will fail.

Then open the file `hive.par.sql` in a text editor (capable of reading UNIX formatted files), and find the version parameter, so it can be set manually,

```
param_( 'version', '%version%' );
```

Modify this line to reflect the version as defined in the `../VERSION` file. For example.

```
param_( 'version', 'v0.1.5.19' );
```

> **Important** -- Make sure to use the correct version data from the file, as this may impact updates and patches at a later time if not

correctly specified.

Build the JAR File

Because Windows does not support bash scripting out of the box, you may be forced to build the `hive-odci.jar` file yourself before proceeding.

On Windows change directory into the `./source/java/` directory. Then create a sub-directory `oracle/mti/odci` for the class files to be copied later. Then using `%JAVA_HOME%/javac` build each class file in the order defined in the `class.order` file ensuring the `%ORACLE_HOME%/jdbc/lib/ojdbc7.jar` file is in the class path.

```
javac -Xlint:unchecked ^
      -Xlint:-deprecation ^
      -XDignore.symbol.file ^
      -cp %CLASSPATH%;%ORACLE_HOME%/jdbc/lib/ojdbc7.jar ^
      log.java
```

Copy the `*.class` file output from the command to sub-directory created ealier (`oracle/mti/odci`). Do this for each file identified in `class.order` and in the order specified.

Once complete, create a JAR file to be loaded in the Oracle 12c database.

```
cd oracle/mti/odci
jar cvf ../../../../jdbc/hive-odci.jar *.class
```

When complete,. you will be ready to start loading the JAR and Drivers into the database.

Load the JDBC Driver

You will not be able to use the `jdbc/load-jdbc.sh` bash script in Windows (unless you have Cygwin or something similar installed). Therefore, you will have load each JAR file manually using the Oracle provided loadjava utility..

For each JAR file you have to load use the following as a template for executing.

```
loadjava -force ^
         -genmissing ^
         -order ^
         -verbose ^
         -resolve ^
         -recursivejars ^
         -resolver "((* hive) (* sys) (* public))" ^
         -user "$sys" ^
         -schema hive ^
         hive-odci.jar
```

Setup

Once Hive-ODCI has been successfully installed and a compliant JDBC Driver loaded you must setup the parameters to use that Driver. The following is **not** an exhaustive overview of the parameters used by Hive-ODCI, just those commonly needed to get started.

*** `hive_jdbc_driver`**

This parameter is the class name of the driver Hive-ODCI will load using `Class.forName()`. This must match the documentation of the driver you choose to use.. Common driver classes are

- CDH 6 Apache Hive JDBC Standalone: `com.cloudera.hive.jdbc.HS1Driver`
- MTI CDH 5, JDBC Thin Driver: `org.mti.hive.jdbc.thin.HiveDriver`
- Progress Data Direct JDBC Hive Driver: `com.ddtek.jdbc.hive.HiveDriver`

*** `hive_jdbc_url`**

This parameter is the URL of the driver, which may or may not contain additional URL parameters. Make sure the driver protocol, host, port, etc... are correct for your environment. Examples, would be ...

- CDH 6 Apache Hive JDBC Standalone: `jdbc:hive2://<host>:<port>/<db>`
- MTI CDH 5, JDBC Thin Driver: `jdbc:hive2://<host>:<port>/<db>`
- Progress Data Direct Hive Driver: `jdbc:datadirect:hive://<host>:<port>`

*** `hive_jdbc_url.x`**

Use these parameters, specified as `hive_jdbc_url.1` ... `hive_jdbc_url.x` in consecutive order (a gap in the numbering sequence will cause Hive-ODCI to stop reading the parameters assuming that it has reached the end of the list.) These parameters are specific to the driver you are using and may change from type to type. Some common examples, would be ...

```
param_( 'hive_jdbc_url.1', 'ssl=0' );
param_( 'hive_jdbc_url.2', 'UID=oracle' );
param_( 'hive_jdbc_url.3', 'PWD=welcome1' );
```

As they are read Hive-ODCI will append them in order to the URL, each delineated by a semi-colon. For example
`jdbc:hive2://hive.mtihq.com:1000;ssl=0;UID=oracle;PWD=welcome1`

*** `hive_user` and `hive_pass`**

The `Driver,.getConnection()` call in Java is overloaded to accept a URL (see above) only or optionally with a User/Password. These parameters are used for those Drivers which expect the User/Password to be provided through the `Driver.getConnection()` call. If these parameters are NULL, they are simply ignored (e.g. not provided to the call)

```
param_( 'hive_user', 'oracle' );
param_( 'hive_pass', 'welcome1' );
```

*** `java_property.x`**

Like the URL parameters, the `java_property.x` parameters are a consecutive list starting at `java_property.1` through `java_property.x` (and like the URL parameters a gap in the numbering sequence will cause Hive-ODCI to stop reading the parameters assuming that it has reached the end of the list). These parameters are used in the `System.setProperty( name, value )` call with the Java Stored Procedure, to setup the Java Environment for the Driver. For example, you may need to setup your Kerberos Authentication parameters, like so ....

```
param_( 'java_property.1', 'java.security.krb5.realm=MTI.COM' );
```

```
param_( 'java_property.2', 'java.security.krb5.realm=kdc.mti.com' );
param_( 'java_property.3', 'java.security.krb5.conf=/etc/krb5.conf' );
param_( 'java_property.4', 'java.security.auth.login.index=Client' );
param_( 'java_property.5', 'java.security.auth.login.config=/etc/jdbc.conf' );
param_( 'java_property.6', 'sun.security.krb5.debug=true' );
```

As they are read Hive-ODCI will set them in the order in which they are encountered. While most Java Properties are not inter-dependent on on another, some may be, so make sure the order you specify is correct.

> Note: Using these parameters may require additional grants not initially anticipated and provided in the installation. You may need to provide grants for any operations not already provided using

```
dbms_java.grant_permission( 'HIVE', 'SYS:<property>', '...', '...' );
```

Removal

Removal of the Hive-ODCI functionality is similar to the Installation procedures. Go the `./source` directory of the install home and using the SYSDBA account run the `remove_hive.sql` script using SQL*Plus. This will drop the schema, including all associated objects, the public synonyms and the tablespaces used when installing Hive-ODCI.

> Note: The procedure for removal remains the same on the Windows platform. There are no additional steps, before or after, which need to be done for this operating system.