Guardium - WKC UDF Installation Guides

Tables of Contents

Installing or updating the Hive UDFs

The IBM Cloud Pak for Data Data Privacy Service (along with Watson Knowledge Catalog) allows you to transform (mask) data from Hive as well as to deny access to users who meet certain criteria. Before you can transform data or deny access, you need to install the Hive user-defined functions (UDFs). This document describes how to install the Hive UDFs.

Note: If you installed the Hive UDFs prior to the Guardium 12.0 release, you need to upgrade them to use the new UDF and Magen 4.7 libraries For more information, see Updating the Hive UDFs.

Installing the Hive UDFs

1. Upload the *dist/hive-udf-jars-X.X-SNAPSHOT.zip* and *scripts/setup_jar.sh* to one of the cluster's nodes that has access to HDFS, where *X.X* is the Hive UDF version.
2. Log in to the same node with a user who has write permission to HDFS.
3. Move *hive-udf-jars-X.X-SNAPSHOT.zip* and *setup_jar.sh* to a temporary directory, for example:

```
- mkdir hive_udf_tmp
- mv hive-udf-jars-1.0-SNAPSHOT.zip setup_jar.sh hive_udfs_tmp
```

4. Run the script in the directory. After it runs, it should print out a *hive-udf-X.X.X.jar *file.

   Note: Make a note of the *hive-udf-X.X.X.jar file name* that you see here.

5. Move all of the jar files in the zip file to *hdfs:///user/hive/auxlibs*, for example:

```
- hdfs dfs -mkdir /user/hive/auxlibs/
- hdfs dfs -copyFromLocal lib/* /user/hive/auxlibs/
- hdfs dfs -chmod -R 777 /user/hive/auxlibs
- hdfs dfs -ls /user/hive/auxlibs
```

6. Log in to the **default** database through beeline. The user must have **admin** role to create functions.

```
beeline -n <user name> -p <password> -u "jdbc:hive2://<hiveserver url>:<hive
server port>/default"
```

7. Use the **`CREATE DATABASE wkc`**command to create the wkc database. Then switch to this database with the **`USE wkc`** command. You will use this database to store the UDFs.
8. Create the UDFs with the following commands:

```
- CREATE FUNCTION mask_udf as 'com.ibm.dbengines.adp.MaskUDF' USING JAR
'hdfs:///user/hive/auxlibs/hive-udf-X.X.X.jar'
- CREATE FUNCTION throw_error_udf as 'com.ibm.dbengines.adp.ThrowErrorUDF'
USING JAR 'hdfs:///user/hive/auxlibs/hive-udf-X.X.X.jar'
```

   Use the hive-udf-X.X.X.jar from step 4, for example, for mask_udf:

```
mycompany.com> CREATE FUNCTION mask_udf as 'com.ibm.dbengines.adp.MaskUDF'
using jar 'hdfs:///user/hive/auxlibs/hive-udf-1.0.6.jar';
INFO  : Compiling command(queryId=hive_20221118140607_b33e4709-e306-490c-a96a-
6a354e68958b): CREATE FUNCTION mask_udf as 'com.ibm.dbengines.adp.MaskUDF'
using jar 'hdfs:///user/hive/auxlibs/hive-udf-1.0.6.jar'
INFO  : Semantic Analysis Completed (retrial = false)
```

```
INFO  : Created Hive schema: Schema(fieldSchemas:null, properties:null)
INFO  : Completed compiling command(queryId=hive_20221118140607_b33e4709-e306-
490c-a96a-6a354e68958b); Time taken: 0.031 seconds
INFO  : Executing command(queryId=hive_20221118140607_b33e4709-e306-490c-a96a-
6a354e68958b): CREATE FUNCTION mask_udf as 'com.ibm.dbengines.adp.MaskUDF'
using jar 'hdfs:///user/hive/auxlibs/hive-udf-1.0.6.jar'
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Added [/tmp/e8b35fdb-a137-4c5d-9752-3d792ab5d86c_resources/hive-udf-
1.0.6.jar] to class path
INFO  : Added resources: [hdfs:///user/hive/auxlibs/hive-udf-1.0.6.jar]
INFO  : Completed executing command(queryId=hive_20221118140607_b33e4709-e306-
490c-a96a-6a354e68958b); Time taken: 0.048 seconds
INFO  : OK
No rows affected (0.123 seconds)
mycompany.com>
```

Verifying the installation

From Hive, use the *describe* function to ensure that *mask_udf* and *throw_error_udf* were successfully created, for example:

```
mycompany.com> describe function mask_udf;
INFO  : Compiling command(queryId=hive_20221110091927_14d05490-face-4b29-ab25-
80dcb53a2a99): describe function mask_udf
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name,
type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20221110091927_14d05490-face-4b29-
ab25-80dcb53a2a99); Time taken: 0.145 seconds
INFO  : Executing command(queryId=hive_20221110091927_14d05490-face-4b29-ab25-
80dcb53a2a99): describe function mask_udf
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20221110091927_14d05490-face-4b29-
ab25-80dcb53a2a99); Time taken: 0.012 seconds
INFO  : OK
+----------------------------------------------------+
|                      tab_name                      |
+----------------------------------------------------+
| There is no documentation for function 'mask_udf'  |
+----------------------------------------------------+
1 row selected (0.63 seconds)
```

Updating the Hive UDFs

1. Download the new version of hive-udf-jars-X.X-SNAPSHOT.zip.
2. Follow steps 1-4 in *Installing the Hive UDFs* to generate the jars.
3. Replace the jars in hdfs:///user/hive/auxlibs/ with the new ones.

```
- hdfs dfs -rm -r /user/hive/auxlibs/*
- hdfs dfs -copyFromLocal hive_udf-jars-X.X-SNAPSHOT/lib/* /user/hive/auxlibs/
- hdfs dfs -chmod -R 777 /user/hive/auxlibs/
```

4. Log in to beeline.
5. Drop the two UDFs that you created earlier:

```
- DROP FUNCTION wkc.mask_udf;
- DROP FUNCTION wkc.throw_error_udf;
```

6. Follow Step 8 in *Installing the Hive UDFs* to recreate the functions again. Make sure to use the correct hive-udf-X.X.X.jar. **Note:** The version number (X.X.X) will be different.

7. Verify the upgrade as described in Verifying the installation.

Installing or updating the Microsoft SQL Server UDFs

Learn to install or update data protection user-defined functions (UDFs) for Microsoft SQL Server.

Note: If you installed SQL Server UDFs prior to the Guardium 12.0 release, you need to update them to use the new UDF and Magen 4.7 libraries. For more information, see Updating the MS SQL Server UDFs.

Prerequisites:

- Windows Server 2016, 2019 or 2022
- Microstoft SQL Server 2014 or 2016
- Microsoft SQL Server Management Studio
- The target Windows machine requires the following dll libraries:
  - kernel32.dll
  - user32.dll
  - libcrypto-3-x64.dll: This library is part of openSSL and is installed when you install openSSL. You can also download the library and add it to the Windows path (usually C:\Windows\System32).
  - msvcp140.dll, vcruntime140.dll, vcruntime140_1.dll: These libraries are usually part of the Visual Studio redistributable. If you need to restore them, you can find instructions online.

Create the UDFs

1. Clone the Watson Knowledge Catalog SQL Server UDF package to your target machine.

   Note: The following steps assume that you clone the repository to a folder called C:\UDF. Adjust the instructions as needed to reflect your actual target.

2. Copy the magen.dll from *C:\UDF\sqlserver-advanced-release\dist\magen.dll* to a location in your environment variable path on the target machine. For example, you can copy magen.dll to *C:\Windows\system32*, which is always part of the path variable.

3. Verify that the magen.dll library is visible in the Windows path and available to SQL Server:

   1. Open a Windows command-line prompt and enter the following command:

      ```
      regsvr32 magen.dll
      ```

   2. The command prompt returns the following message (or similar):

      ```
      The module magen.dll was loaded but the entry-point DllRegisterServer was
      not found. Make sure that magen.dll is a valid DLL or OCX file and then
      try again.
      ```

   Note: You can ignore the *entry-point DllRegisterServer* not found warning.

   If you receive an error that indicates that magen.dll was not found (or similar critical error) you need to address it.

   Note: When you load a Windows library with the *regsrv32* command, if either the *libname.dll* (in this case, magen.dll) or any of its dependencies are missing, a Windows error indicates that the *libname.dll* is not found. To address *libname.dll not found* errors, ensure that magen.dll and its required dependencies

are not corrupted and that they are correctly installed in a folder within the Windows path (such as C:\Windows\system32).

4. Update the target database to allow you to create and run unsafe assemblies:

    1. Connect to your SQL Server instance with SQL Server Management Studio and open a command line window. Connect as a member of the admin group on your Windows machine (that is, Administrator). In SQL Server, that should log you in as SA (system administrator).

    2. The next steps allow your databases to accept untrusted assemblies (that is, the UDFs). Run the following commands to switch to the master database and grant unsafe assembly to your system administrator:

        ```
        USE MASTER;
        GRANT UNSAFE ASSEMBLY TO [SA];
        GO
        ```

    3. If needed, switch to your target database. To deploy the UDFs to MASTER, then skip this step. To deploy the UDFs to the WKC database, run the following commands:

        ```
        USE WKC;
        GO
        ```

    4. Run the following commands to allow you to create unsafe assemblies in the WKC database:

        ```
        ALTER DATABASE WKC SET trustworthy ON;
        GO
        ```

    5. Run the following commands to allow you to run unsafe assemblies:

        ```
        EXEC sp_configure 'clr enabled', 1;
        RECONFIGURE;
        GO
        ```

Add the UDFs to SQL Server

1. Browse to the folder C:\UDF\sqlserver-advanced-release\scripts.

2. Edit the **`setup.sql`** file. Replace the command **USE TEST** to use the name of the target database where you will create the UDFs. For example: **USE MYDB**.

3. In the CREATE ASSEMBLY command, change the path to src.dll to the path where you cloned or installed the repository. For example, if you cloned the repository to the C:\UDF\sqlserver-advanced-release\dist\bin\Release\ directory, run the following command:

    ```
    C:\UDF\sqlserver-advanced-release\dist\bin\Release\src.dll
    ```

4. In a Windows command prompt, run the following command:

    ```
    sqlcmd -e -i setup.sql
    ```

    Note: Instead of running *sqlcmd* from the command line, you can copy and paste all of the TSQL commands from the *setup.sql* file to a SQL Server Management Studio window and run the commands 'manually' from there.

5. When all commands in *setup.sql* run successfully, the UDFs are created.

Set up the resources folder and path

The masking library uses a number of dictionaries, which are available in the folder C:\UDF\sqlserver-advanced-release\dist\resources.

1. Copy the *resources* folder to any location on the target machine. For example, C:\Masking\resources.

   The setup.sql script creates a stored procedure in the WKC schema called WKC.SET_RESOURCES_FOLDER_PATH.

2. Run WKC.SET_RESOURCES_FOLDER_PATH to indicate to the masking library where to find the resources folder. Set the second parameter to the path where the resources folder is located (in this example, *C:\Masking*), and then run the stored procedure in SQL Server Management Studio:

   ```
   declare @status varchar(max);
   exec WKC.SET_RESOURCES_FOLDER_PATH @status output,
   @resources_path='C:\Masking\';
   select @status;
   GO
   ```

3. The variable @status returns a message from the stored procedure. When the resources folder is correctly found and its contents loaded, the return status indicates:

   ```
   SUCCESS: The RESOURCES folder content was loaded successfully
   ```

Test the UDFs

To test the UDFS, run the scripts that start with test_* in the C:\UDF\sqlserver-advanced-release\scripts directory.

Updating the MS SQL Server UDFs

If you already have MS SQL Server UDFs installed, you need to upgrade to the new UDFs and Magen 4.7 libraries.

Upgrade steps

1. Clone the Watson Knowledge Catalog SQL Server UDF package to your target machine.

   Note: The following steps assume that you clone the repository to a folder called **`C:\UDF`**. Adjust the instructions as needed to reflect your actual target.

2. Log in to the SQL Server database and unload the previously installed Magen library

   ```
   exec WKC.UNLOAD_MAGEN_LIB
   ```

3. Stop SQL Server service

   ```
   net stop MSSQL$SQLEXPRESS
   ```

4. Copy the extracted magen.dll to previous magen.dll location (i.e. C:\Windows\System32)

   ```
   copy  "*C:\UDF\sqlserver-advanced-release\dist\magen.dll*"
   "C:\Windows\System32\magen.dll"
   ```

5. Make sure that **libcrypto-3-x64.dll**, is in the same location as the **magen.dll** location. Re-register the **magen.dll**. For example:

   ```
   regsvr32 C:\Windows\System32\magen.dll
   ```

   Note: Verify that the magen.dll loaded successfully. You can ignore the **entry-point DllRegisterServer not found** warning.

6. Start the SQL Server service:

```
net start MSSQL$SQLEXPRESS
```

7. Browse to **C:\UDF\sqlserver-advanced-release\scripts\ and then edit the** setup.sql` file. Replace the command **USE TEST** to use the name of the target database where you created the UDFs. For example: **USE MYDB**.

8. Run **setup.sql** either from the command line or from SQL Server Management Studio.

   - Command line

     ```
     sqlcmd -e -i setup.sql
     ```

   - Copy and paste all of the TSQL commands from the setup.sql file to a SQL Server Management Studio window and run the commands 'manually' from there.

9. Copy the *resources* folder to any location on the target machine. For example, C:\Masking\resources.

   The setup.sql script creates a stored procedure in the WKC schema called WKC.SET_RESOURCES_FOLDER_PATH.

10. Run WKC.SET_RESOURCES_FOLDER_PATH to indicate to the masking library where to find the resources folder. Set the second parameter to the path where the resources folder is located (in this example, *C:\Masking*), and then run the stored procedure in SQL Server Management Studio. For example:

```
declare @status varchar(max);
exec WKC.SET_RESOURCES_FOLDER_PATH @status output, @resources_path='C:\Masking\';
select @status;
GO
```

The variable @status returns a message from the stored procedure. When the resources folder is correctly found and its contents loaded, the return status indicates:

```
  SUCCESS: The RESOURCES folder content was loaded successfully
  ```

Test the UDFs
To test the UDFS, run the scripts that start with test_* in the ` C:\UDF\sqlserver-advanced-release\scripts ` directory.

Uninstalling the UDFs

To uninstall UDFs, edit, then run the drop.sql script located in C:\UDF\sqlserver-advanced-release\scripts, replacing "TEST" with the database name on which you installed the UDFs:
```

sqlcmd -e -i drop.sql
```

Installing or updating MySQL UDFs

Prerequisites

- Solaris (SPARC), RHEL 7.x, RHEL 8.x, or compatible operating systems
- MySQL database server: 5.x, 8.x, or compatible
- OpenSSL (libcrypto provided in libs)

Installing the MySQL UDFs

1. Download the Watson Knowledge Catalog MySQL UDF package and copy it to your database server.

2. Unpack the tarball on your database server, for example:

```
# pwd
/tmp
# tar -xvf mysql-advanced-release-example.0.1.tar.gz
# ls
mysql-advanced-release-example.0.1/  mysql-advanced-release-example.0.1.tar.gz
# cd mysql-advanced-release-example.0.1
# ls
dist/  libs/  README.md  scripts/
```

3. Change directories to the /dist folder that you unpacked in step 2. Unpack the version of *libmagen* and *libtxmysql* that are appropriate for your operating system.

```
# cd dist/
# tar -xvf libmagen.rhel_7.9-4.6.0.tgz
libmagen.so
libmagen.rhel_7.9-4.6.0.so
# tar -xvf libtxmysql.rhel_7.9-4.6.0.tgz
libtxmysql_rhel_7.9-4.6.0.so
libtxmysql.so
```

4. Copy the *libmagen.so* dependency library. For RHEL-based systems, copy the library to /usr/lib64. For Solaris, copy the library to /lib/64. Make sure that you preserve symlinks. Change the ownership and group to *root:root*. For example:

```
RHEL:
# cp -a libmagen.*so /usr/lib64/
# chown root.root /usr/lib64/libmagen*

Solaris:
# cp -P libmagen.*so /lib/64/
# chown root:root /lib/64/libmagen*
```

5. Locate your MySQL plugin path and copy the WKC MySQL UDF library *libtxmysql.so* to that directory. Change the ownership and group as needed. For example:

```
RHEL:
# cp -a dist/libtxmysql*.so ${MYSQL_HOME}/mysql/lib/plugin
# chown mysql8.mysql ${MYSQL_HOME}/mysql/lib/plugin/libtxmysql*.so

Solaris:
```

```
# cp -P dist/libtxmysql*.so ${MYSQL_HOME}/mysql/lib/plugin
# chown mysql8:mysql ${MYSQL_HOME}/mysql/lib/plugin/libtxmysql*.so
```

6. Copy both folders, dist/resources and scripts, to the ${MYSQL_HOME} directory. Change the ownership and group as needed. For example:

```
RHEL:
# cp -R resources ${MYSQL_HOME}
# chown -R mysql8.mysql ${MYSQL_HOME}/resources
# cp -R scripts ${MYSQL_HOME}
# chown -R mysql8.mysql ${MYSQL_HOME}/scripts

Solaris:
# cp -RP resources ${MYSQL_HOME}
# chown -R mysql8:mysql ${MYSQL_HOME}/resources
# cp -RP scripts ${MYSQL_HOME}
# chown -R mysql8:mysql ${MYSQL_HOME}/scripts
```

7. Use an administrative or privileged MySQL account to register the WKC UDFs. For example:

```
# Create a database named WKC
mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/setup.sql"
# Register UDFs
mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/register_udf.sql"
```

Verifying the installation

To verify the installation, run the following commands:

```
- mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_mask_string.sql"
- mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_mask_numerics.sql"
- mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_mask_datetime.sql"
- mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_mask_tonull.sql"
- mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_stored_procedures.sql"
```

Note: If successful, all test cases pass except for *test_stored_procedures.sql*.

The following output (or similar) is expected when you run the *test_stored_procedures.sql* script.

```
ERROR 1644 (28000) at line 1 in file:
'/home/mysql8/scripts/test_stored_procedures.sql': Error message (-99)
```

Updating the MySQL UDFs

If the MySQL UDF is already installed at your site (pre-Guardium 12.0) you need to update the UDF and the magen library to magen 4.7.x. The steps required to update the UDFs are similar to installing. The main difference is that you need to run some additional scripts.

1. Follow steps 1 through 6 of *Installing the MySQL UDFS*.

2. Run update.sql to upgrade the UDFs in the database.

```
 # mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
dbutil/update.sql"
```

3. Verify the installation

   Run the following scripts to verify the installation.

```
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_maskDateTime.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_maskDouble.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_maskFloat.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_maskIntShortLong.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_maskNumeric.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_numericShift.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_maskChar.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_mask_string.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_mask_numerics.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_mask_datetime.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_mask_tonull.sql"
# mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/test_stored_procedures.sql"
```

Note: If successful, all test cases pass except for *test_stored_procedures.sql*.

The following output (or similar) is expected when you run the *test_stored_procedures.sql* script.

```
ERROR 1644 (28000) at line 1 in file:
'/home/mysql8/scripts/test_stored_procedures.sql': Error message (-99)
```

Removing the UDFs

Run the following script to remove the UDFs from your MySQL database:

```
mysql -uYOUR_USERNAME -pYOUR_PASSWORD YOUR_DATABASE -e "source
${MYSQL_HOME}/scripts/unregister_udf.sql"
```

Installing or updating the Oracle UDFs

Learn to install or update the Oracle UDFs. The IBM Cloud Pak for Data Data Privacy Service (along with Watson Knowledge Catalog) allows you to transform (mask) data from Oracle. Before you can transform data, you need to install the Oracle user-defined functions (UDFs).

Note: If you installed SQL Server UDFs prior to the Guardium 12.0 release, you need to upgrade them to use the new UDF and Magen 4.7 libraries For more information, see Updating Oracle UDFs.

Prerequisites

- Solaris (SPARC), RHEL 7.x, RHEL 8.x, or compatible operating systems
- OpenSSL

Installating the UDFs

1. Download Watson Knowledge Catalog Oracle UDF package and copy it to your database server
2. Unpack the tarball on your database server:

   For example,

   ```
   # pwd
   /tmp
   # tar -xvf oracle-advanced-release-example.0.1.tar.gz
   # ls
   oracle-advanced-release-example.0.1/  oracle-advanced-release-
   example.0.1.tar.gz
   # cd oracle-advanced-release-example.0.1
   # ls
   dist/  libs/  README.md  scripts/
   ```

3. Change directories to the dist folder unpacked in the previous step. Unpack the libmagen and libUDF that is appropriate for your operating system. For example:

   ```
   # cd dist/
   # tar -xvf libmagen.rhel_7.9-4.6.0.tgz
   libmagen.so
   libmagen.rhel_7.9-4.6.0.so
   # tar -xvf libUDF-rhel_7.9-4.6.0.tgz
   libUDF.so
   libUDF-rhel_7.9-4.6.0.so
   ```

4. Copy the resources folder from the dist/resources directory to $ORACLE_HOME, and copy the Magen and UDF libraries to $ORACLE_HOME/libs. If needed, override any previous versions. For example:

   ```
   RHEL:
   cp -ar dist/resources $ORACLE_HOME/
   cp -a dist/lib*.so $ORACLE_HOME/lib/

   SOLARIS:
   cp -RP dist/resources $ORACLE_HOME/
   cp -P dist/lib*.so $ORACLE_HOME/lib/
   ```

5. Check the permissions on the libraries that you copied in the previous step. Make sure that permissions are set to 755 oracle19:oradba.

```
-rwxr-xr-x  1 oracle19 oradba   1632752 Nov 18 12:32 libmagen.rhel_7.9-
4.6.0.so
-rwxr-xr-x  1 oracle19 oradba     19808 Nov 18 12:32 libUDF-rhel_7.9-4.6.0.so
lrwxrwxrwx  1 oracle19 oradba        26 Nov 28 08:48 libmagen.so ->
libmagen.rhel_7.9-4.6.0.so
lrwxrwxrwx  1 oracle19 oradba        24 Nov 30 16:48 libUDF.so -> libUDF-
rhel_7.9-4.6.0.so
```

6. Register the UDFs.

   Note: For multitenant container database configuration (default), you might need to register the UDFs on the containers that are appropriate for your environment. You might also need to change the password for the WKC user in setup1_sys.sql in accordance with your database security standards.

   Non-containerized database configuration example:

```
cd ../scripts

# Script to create wkc user.
sqlplus sysdba/syspassword @setup1_sys.sql

# Script to register udfs.
sqlplus wkc/wkc @setup2_wkc.sql

# Script to create throw procedure
sqlplus wkc/wkc @setup3_wkc.sql
```

   Containerized database configuration example:

   The following example shows that you musst set up the UDFS on the *ON9PSNIF* database.

```
$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Oct 27 09:53:03 2022
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 ON9PSNIF                       READ WRITE NO
```

   Log in to the apprORpriate database using **your** DBA account (or any account with necessary privileges) to configure the UDFs.

```
# Script to create wkc user on ON9PSNIF.
sqlplus sysdba/syspassword@ON9PSNIF @setup1_sys.sql

# Script to register udfs on ON9PSNIF
sqlplus wkc/wkc@ON9PSNIF @setup2_wkc.sql
```

```
# Script to create throw procedure on ON9PSNIF
sqlplus wkc/wkc@ON9PSNIF @setup3_wkc.sql
```

Note: If the UDFs are installed on the seed database, they may not be visible to user accounts in other databases. Install the UDFs where they can be referenced by user accounts that may be subject to WKC transformation actions

After running these scripts your output should resemble the following

```
Session altered.

Library created.

Function created.
```

Note: Depending on your database configuration and administrative privilege settings you might see an error message similar to the following message when you run the setup SQL scripts. As long as the functions are created and run properly when tested, you can ignore this error.

```
ERROR:
ORA-02097: parameter cannot be modified because specified value is invalid
ORA-01031: insufficient privileges
```

7. Grant permissions. Depending on your database configuration and individual requirements, you can decide how to accomplish this step. Any user that may be subject to transformation action must have execute permission on these functions for the feature to work properly. An example of how to create a WKC role and execute permissions on the UDFs created in the previous step follows.

```
create role wkc_role;

grant execute on wkc.maskInteger to wkc_role;
grant execute on wkc.maskDateTime to wkc_role;
grant execute on wkc.maskDate to wkc_role;
grant execute on wkc.maskShort to wkc_role;
grant execute on wkc.maskLong to wkc_role;
grant execute on wkc.maskDouble to wkc_role;
grant execute on wkc.maskFloat to wkc_role;
grant execute on wkc.maskNumber to wkc_role;
grant execute on wkc.maskString to wkc_role;
grant execute on wkc.maskToNull to wkc_role;
grant execute on wkc.maskBoolean to wkc_role;
grant execute on wkc.THROW to wkc_role;
grant execute on wkc.maskDate2 to wkc_role;
grant execute on wkc.maskDateTime2 to wkc_role;
grant execute on wkc.maskTime2 to wkc_role;

grant wkc_role to ibm_example_user;
```

After you create a role and grant permissions, your output should include the following messages:

```
Role created.
Grant succeeded.
```

Note: In the absence of role based permissions, you can also grant Execute permissions to public.

For example:

```
grant execute on wkc.maskInteger to public;
grant execute on wkc.maskDateTime to public;
grant execute on wkc.maskDate to public;
grant execute on wkc.maskShort to public;
```

```
grant execute on wkc.maskLong to public;
grant execute on wkc.maskDouble to public;
grant execute on wkc.maskFloat to public;
grant execute on wkc.maskNumber to public;
grant execute on wkc.maskString to public;
grant execute on wkc.maskToNull to public;
grant execute on wkc.maskBoolean to public;
grant execute on wkc.THROW to public;
grant execute on wkc.maskDate2 to wkc_role;
grant execute on wkc.maskDateTime2 to wkc_role;
grant execute on wkc.maskTime2 to wkc_role;
```

8. Log in to SQL with a normal user account and test the UDFs

```
sqlplus ibm_test/test

select wkc.maskShort(0,'5','','',4) from dual;
select wkc.maskLong(2,'5','','',2143647) from dual;
select wkc.maskInteger(0,'5','','',4) from dual;
select wkc.maskDouble(8,'5','','',4.4d) from dual;
select wkc.maskFloat(4,'5','','',4.4f) from dual;
select
wkc.maskNumber(2,'','','0123456789abcdef','-2345678991231312312324256546575423
4234231122334341.11232432343434545435534543543122456 7',0,-3) from dual;
select wkc.maskString(4,'','UsaSocialSecurityNumber','0123456789abcdef','123-
45-6789',20) from dual;
select wkc.maskDate2(0,'1990-10-24','','','1985-09-02') from dual;
select wkc.maskDate2(4,'','Date','0123456789abcdef','2020-02-29') from dual;
select wkc.maskDate2(10,'','Date','0123456789abcdef','2020-02-29') from dual;
select wkc.maskTime2(0,'09:45:56','','','02:38:12') from dual;
select wkc.maskDateTime2(0,'1990-10-24 20:00:00','','','1985-09-02') from
dual;
select wkc.maskDateTime2(4,'','DateTime','0123456789abcdef','2020-02-29
12:52:42.278') from dual;
select wkc.maskDateTime2(10,'','DateTime','0123456789abcdef','2020-02-29
12:52:42.278') from dual;
select wkc.maskDate('1985-09-02') from dual;
select wkc.maskTime('02:38:12') from dual;
select wkc.maskDateTime('2020-02-29 12:52:42.278') from dual;
select wkc.maskBoolean(0,'5','','','F') from dual;
select wkc.THROW('errortext','2') from dual;
```

Testing the stored procedure

```
EXECUTE THROW('Error Text', '-99');
```

Note: If you encounter an error similar to the following message, check your **$ORACLE_HOME/hs/admin/extproc.ora** configuration. Depending on other external UDFs installed on your database, and the configuration in this file, you might need to adjust settings such as **EXTPROC_DLLS** or **LD_LIBRARY_PATH**.

```
SQL> select wkc.maskShort(0,'5','','',4) from dual;
select wkc.maskShort(0,'5','','',4) from dual
*
ERROR at line 1:
ORA-28595: Extproc agent : Invalid DLL Path
```

The following line shows an example of the **LD_LIBRARY_PATH** when you add it to **extproc.ora**. Your exact path might differ depending on the details of your installation.

```
SET
LD_LIBRARY_PATH=/export/home/oracle18/app/oracle/product/18.0.0.0/dbhome_1/lib
;
```

Updating Oracle UDFs

If you installed the Oracle UDFs prior to the Guardium 12.0 release, you need to upgrade to the new UDF and Magen 4.7 libraries.

Prerequisites

- Solaris (SPARC), RHEL 7.x, RHEL 8.x, or compatible operating systems
- OpenSSL

Upgrade steps

1. Follow the installation steps 1 through 4. Overwrite any existing files as needed.
2. Check permissions on the libraries that you copied in the previous step. The correct permissions are **755 oracle:oinstall**: For example:

```
# chown -R oracle:oinstall libUDF.so libmagen.so
# chmod  -R 755 libUDF.so libmagen.so

-rwxr-xr-x. 1 root   root        20152 Jun  7 03:31 libUDF-oracle-rhel_7.9-
4.7.0.so
-rwxr-xr-x. 1 root   root      2393328 Jun  7 03:31 libmagen.rhel_7.9-4.7.0.so
lrwxrwxrwx. 1 oracle oinstall       26 Jun  7 03:31 libmagen.so ->
libmagen.rhel_7.9-4.7.0.so
lrwxrwxrwx. 1 oracle oinstall       24 Jun  7 03:31 libUDF.so -> libUDF-
oracle-rhel_7.9-4.7.0.so
```

3. Register the UDFs by logging in to SQL with proper credentials. For details about containerbased vs non-container based Oracle deployments, see Step 6 of Installing the UDFs.

```
cd ../scripts

# Script to register udfs.
sqlplus wkc/wkc1 @setup2_wkc.sql

# Script to create throw Function
sqlplus wkc/wkc1 @setup3_wkc.sql
```

4. Grant permissions for the newly created UDFs in the same manner you granted them originally (in step 7). For example:
   - For role based permission:

     ```
     grant execute on wkc.maskDate2 to wkc_role;
     grant execute on wkc.maskDateTime2 to wkc_role;
     grant execute on wkc.maskTime2 to wkc_role;
     ```

   - For general public permission:

     ```
     grant execute on wkc.maskDate2 to wkc_role;
     grant execute on wkc.maskDateTime2 to wkc_role;
     ```

```
grant execute on wkc.maskTime2 to wkc_role;
```

See step 7 of the Installing UDFs section for more information.

Testing the upgrade

To test your upgraded UDFs, run the following scripts. TIf all tests pass, the upgrade is successful.

```
cd test

sqlplus wkc/wkc1 @test_maskChar.sql
sqlplus wkc/wkc1 @test_maskDateTime.sql
sqlplus wkc/wkc1 @test_maskDouble.sql
sqlplus wkc/wkc1 @test_maskFloat.sql
sqlplus wkc/wkc1 @test_maskIntShortLong.sql
sqlplus wkc/wkc1 @test_maskNumeric.sql
sqlplus wkc/wkc1 @test_numericShift.sql
sqlplus wkc/wkc1 @test_maskUpgradeTest.sql
```

Installing or updating PostgreSQL UDFs

Learn to install or update data protection user-defined functions (UDFs) for PostgreSQL.

Note: The steps for updating and installing are identical. When prompted, overwrite the previously installed version. If you installed PostgreSQL UDFs prior to the Guardium 12.0 release, you need to upgrade them to use the new UDF and Magen 4.7 libraries.

Prerequisites

- RHEL8
- OpenSSL
- PostgreSQL 11 - 14

Installing (or upgrading) the UDF package

1. Download the Watson Knowledge Catalog PostgreSQL UDF package and copy it to your database server.
2. Unpack the .tar file on your database server.

```
$0:~> tar -xf postgresql-advanced-release-example-0.1.tar.gz
$0:~> tar -xf postgresql-advanced-release-example-0.1.tar.gz
```

3. Change directories to your newly unpacked folder. Unpack the UDF and magen libraries that match your platform and database version.

```
$0:~> cd postgresql-advanced-release-example-0.1/dist

$0:postgresql-advanced-release-example-0.1/dist> ls -1
libmagen.rhel.8.x-4.6.0.tgz
postgresql_udf_pg11_rhel_8.x-4.6.0.tgz
postgresql_udf_pg12_rhel_8.x-4.6.0.tgz
postgresql_udf_pg13_rhel_8.x-4.6.0.tgz
postgresql_udf_pg14_rhel_8.x-4.6.0.tgz
resources/
```

Be sure to unpack the UDF that matches your version of PostgreSQL. For example, for PostgreSQL 12.x:

```
$0:postgresql-advanced-release-example-0.1/dist> tar -xf
postgresql_udf_pg12_rhel_8.x-4.6.0.tgz
$0:postgresql-advanced-release-example-0.1/dist> txr -xf libmagen.rhel.8.x-
4.6.0.tgz

$0:postgresql-advanced-release-example-0.1/dist> ls -1

dist/
libmagen.rhel.8.x-4.6.0.so*
libmagen.rhel.8.x-4.6.0.tgz
libmagen.so@
postgresql_udf_pg11_rhel_8.x-4.6.0.tgz
postgresql_udf_pg12_rhel_8.x-4.6.0.tgz
postgresql_udf_pg13_rhel_8.x-4.6.0.tgz
postgresql_udf_pg14_rhel_8.x-4.6.0.tgz
resources/
```

```
scripts/
```

4. Copy **libmagen.so** and **dist/libtxpostgresql_pg${version_number}.so** to the $libdir directory. Preserve all symbolic links. *Note* Make sure that the **dynamic_library_path** parameter includes $libdir. PostgreSQL recommends that you set this parameter in the postgresql.conf configuration file.

```
$0:postgresql-advanced-release-example-0.1/dist> cp -a libmagen.*so
/usr/pgsql-12/lib/
$0:postgresql-advanced-release-example-0.1/dist> cp -a
dist/libtxpostgresql_pg12*.so /usr/pgsql-12/lib/
```

5. Recursively copy resources and the script directory to the postgres user ${HOME} directory ${POSTGRES_HOME}. Change the ownership and group as needed to match your postgres account:

```
$0:postgresql-advanced-release-example-0.1/dist> cp -R resources
/var/lib/pgsql
$0:postgresql-advanced-release-example-0.1/dist> cp -R scripts /var/lib/pgsql
```

6. Log in to a superuser account, and then register the UDFs:

```
psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/setup.sql
psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/register_udf_pg12.sql
```

Testing the UDF

To test the UDF installation, run the following procedures:

```
To run the following tests please first run test_setup.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_setup.sql

- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_mask_char.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_mask_string.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_maskDouble.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_mask_numerics.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_maskNumeric_testcases.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_mask_datetime.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_MaskDateTime_testcases.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_mask_tonull.sql
- psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/test_stored_procedures.sql
```

The following output (or similar) is expected when you run the **test_stored_procedures.sql** script:

```
ERROR:  Error message (-99)
CONTEXT:  PL/pgSQL function wkc.throw(text,text) line 3 at RAISE
```

Uninstalling the UDFs

To uninstall UDFs, run the following script:

```
psql postgresql://YOUR_USERNAME:YOUR_PASSWORD@localhost/YOUR_DATABASE -a  -f
${POSTGRES_HOME}/scripts/unregister_udf.sql
```

Installing or updating Teradata UDFs

The IBM Cloud Pak for Data Data Privacy Service allows you to transform (mask) data from Teradata. Before you can transform data, you need to install the Teradata user-defined functions (UDFs). If the Teradata UDFs are already installed at your site (pre-Guardium 12.0), you need to update the UDF and the magen library to magen 4.7. For more information, see Updating Teradata UDFs.

Prerequisites

- Teradata 17 on SUSE 12 SP3
- OpenSSL
- GCC compiler libstdc++6 version 11.3.0 installed

Note In a multi-node Teradata deployment, or clique, the resource files and libraries, and dependencies must be distributed to each node that can execute queries involving WKC enforcement.

Installing Teradata UDFs

1. Log on as or change to a privileged user who can work in the Teradata directories (such as root):

   ```
   sudo su - root
   ```

2. Copy the **resources** folder from the dist folder to /usr/tdbms/:

   ```
   # MULTI NODE
   pcl -send dist/resources /usr/tdbms/

   # SINGLE NODE
   cp -R dist/resources /usr/tdbms/
   ```

3. Copy libmagen tar and adpMask tar files from the dist folder to /usr/tdbms/lib/:

   ```
   # MULTI NODE
   pcl -send dist/*.tgz /usr/tdbms/lib/

   # SINGLE NODE
   cp -a dist/*.tgz /usr/tdbms/lib/
   ```

4. Unpack libmagen tar and adpMask tar files to /usr/tdbms/lib:

   ```
   # MULTI NODE
   psh tar -xf /usr/tdbms/lib/adpMask*.tgz -C /usr/tdbms/lib/
   psh tar -xf /usr/tdbms/lib/libmagen*.tgz -C /usr/tdbms/lib/

   # SINGLE NODE
   tar -xf /usr/tdbms/lib/adpMask*.tgz -C /usr/tdbms/lib/
   tar -xf /usr/tdbms/lib/libmagen*.tgz -C /usr/tdbms/lib/
   ```

   Optionally remove the tar files

   ```
   # MULTI NODE
   psh rm /usr/tdbms/lib/adpMask*.tgz /usr/tdbms/lib/libmagen*.tgz

   # SINGLE NODE
   rm /usr/tdbms/lib/adpMask*.tgz /usr/tdbms/lib/libmagen*.tgz
   ```

5. Change the owner and group of copied folders and files appropriately (default is root:tdrusted):

```
# MULTI NODE
psh chown -R root:tdtrusted /usr/tdbms/resources
psh chmod -R 555 /usr/tdbms/resources

psh chown root:tdtrusted /usr/tdbms/lib/libmagen*.so
psh chmod 555 /usr/tdbms/lib/libmagen*

psh chown root:tdtrusted  /usr/tdbms/lib/adpMask*.o
psh chmod 555 /usr/tdbms/lib/adpMask*.o


# SINGLE NODE
chown -R root:tdtrusted /usr/tdbms/resources
chmod -R 555 /usr/tdbms/resources

chown root:tdtrusted /usr/tdbms/lib/libmagen*.so
chmod 555 /usr/tdbms/lib/libmagen*

chown root:tdtrusted  /usr/tdbms/lib/adpMask*.o
chmod 555 /usr/tdbms/lib/adpMask*.o
```

6. Set a password for the WKC user in the *create_adp_db.sql* script that conforms to your database security standards. **Note**: Run the following steps on a single Teradata node only.

```
vi scripts/create_adp_db.sql

CREATE USER WKC
AS
PERM = 5368709120,
PASSWORD = "", -- CHANGE
SPOOL= 5368709120,
TEMPORARY = 2147483648;
```

Note: In subsequent steps, you can either run the provided scripts or execute the commands manually.

7. Log in, using an administrative account that can create users and databases and grant execute permissions.

```
cd scripts/

bteq
.LOGON 127.0.0.1/DBC

.RUN FILE=create_adp_db.sql

.LOGOFF
```

After you either manually create the user or run *create_adp_db.sql*, you should see messages similar to:

```
*** User has been created.

*** Grant accepted.
```

8. Log in to the database as the newly created WKC user and run the *setup_udf.sql* and *setup_sp.sql* scripts. The scripts create the functions and procedure, and links them to the appropriate libraries.

```
bteq
.LOGON 127.0.0.1/WKC

.RUN FILE=setup_udf.sql
```

```
.COMPILE FILE=setup_sp.sql

.LOGOFF
```

After you run the setup scripts, you should see messages that include the following statement:

```
*** Function has been created.
*** Procedure has been created.
```

9. Grant permissions. Depending on your database configuration and individual requirements, you can decide how to accomplish this step. Any user that may be subject to transformation action must have execute permission on these functions for the feature to work properly. An example of how to grant public execute permissions on the UDFs created in the previous step follows.

```
bteq

.LOGON 127.0.0.1/DBC

GRANT EXECUTE function on WKC to PUBLIC;

GRANT EXECUTE procedure on WKC to PUBLIC;

.LOGOFF
```

After granting permissions, you should see the following messages:

```
 *** Grant accepted.
```

10. Depending on your configuration, you may need to restart Teradata database after you install and configure the UDFs

```
/etc/init.d/tpa stop
/etc/init.d/tpa start
```

11. Verify functions and procedures are registered by using the following SQL:

```
SELECT TableName FROM DBC.TablesV T WHERE databasename = 'WKC' and TableKind
in ('P', 'E', 'F') ORDER BY TableName;
```

Expected output

```
ADPMASK_BIGINT
ADPMASK_BOOLEAN
ADPMASK_DATE
ADPMASK_DOUBLE
ADPMASK_FLOAT
ADPMASK_INT
ADPMASK_NUMBER
ADPMASK_SMALLINT
ADPMASK_STRING
ADPMASK_TIME
ADPMASK_TIMESTAMP
ADPMASK_TONULL
THROW
```

12. SQL queries to verify functions and procedures

```
sel WKC.MASKSTRING(1,'maskparams', 'formatname', 'seedval','actualdata',1);
sel WKC.MASKINT(1,'maskparams', 'formatname', 'seedval',12) ;
sel WKC.MASKDOUBLE(0,'maskparams', 'formatname', 'seedval', cast(1234.55 as
DOUBLE PRECISION)) ;
sel WKC.MASKBIGINT(0,'maskparams', 'formatname', 'seedval', 1231212) ;
sel WKC.MASKSHORT(0,'maskparams', 'formatname', 'seedval', 2) ;
sel WKC.MASKFLOAT(0,'maskparams', 'formatnames', 'seedval', 12.12) ;
```

```
sel WKC.MASKDATE('2020-01-01' );
sel WKC.MASKTIME('01:01:01.000001') ;
sel WKC.MASKTIMESTAMP('0001-01-01 01:01:01.000001');
sel WKC.MASKTONULL();
sel WKC.MASKBOOLEAN(0,'maskparams', 'formatname', 'seedval', '1') ;
sel WKC.MASKNUMBER(0,'', '', '', '123456789') ;
sel WKC.MASKNUMBER( 2, '', '', '0123456789abcdef',
'-234567899123131231232425654657542342342311223343412312432343434545435534543543122456
7', 0, 0 );

call WKC.THROW('ASD','121');
```

Updating Teradata UDFs

If the Teradata UDFs are already installed at your site (pre-Guardium 12.0), you need to update the UDF and the magen library to magen 4.7.x.

Prerequisites

- Teradata 17 on SUSE 12 SP3
- OpenSSL
- GCC compiler libstdc++6 version 11.3.0 installed
- Previous version of magen and UDFs installed

Note In a multi-node Teradata deployment, or clique, you must distribue the resource files and libraries, along with their dependencies, to each node that can execute queries that involve Watson Knowledge Catalog enforcement.

Updating the UDFs

1. Download the Watson Knowledge Catalog Teradata UDF package and copy it to your database server.
2. Log on as or change to a privileged user who can work in the Teradata directories (such as root).

   ```
   sudo su - root
   ```

3. Unpack the tarball on your database server. For example,

   ```
   bash
   # pwd
   /tmp
   # tar -xvf teradata-advanced-release-example.0.1.tar.gz
   # ls
   teradata-advanced-release-example.0.1/  teradata-advanced-release-
   example.0.1.tar.gz
   # cd teradata-advanced-release-example.0.1
   # ls
   dist/  libs/  README.md  scripts/
   ```

4. Unregister the previously installed UDFs. Log in to your database with proper credentials and run the `cleanup.sql` script.

   ```
   cd scripts

   # Login to bteq using the following command.
   ```

```
bteq
.LOGON 127.0.0.1/dbUser
password: userPassword

# Script to unregister udfs.
.RUN FILE=cleanup.sql
```

5. Check that all functions are removed by using the following command:

```
SELECT TableName FROM DBC.TablesV T WHERE databasename = 'WKC' and TableKind
in ('P', 'E', 'F') ORDER BY TableName;
```

Note: If a UDF displays in the result, try to remove it manually before proceeding.

6. Depending on your configuration, you may need to restart the Teradata database before you install and configure the new UDFs

```
/etc/init.d/tpa stop
/etc/init.d/tpa start
```

Note: If you recieve the following error message when you install the new UDFS, you must restart your database:

```
Errors/Warnings reported during compilation
------------------------------------------------------------------------------
/usr/bin/gcc -D_REENTRANT -D_LIBC_REENTRANT    -I/usr/tdbms/etc   -L/usr/td
bms/lib -L/usr/tdbms/lib  -fpic -c Teradata_new_delete.cpp
/usr/bin/gcc -D_REENTRANT -D_LIBC_REENTRANT    -I/usr/tdbms/etc   -L/usr/td
bms/lib -L/usr/tdbms/lib   -fpic -c  pre_maskTime.c
/usr/bin/gcc -shared -fpic -Xlinker -rpath -Xlinker /usr/tdbms/lib  -Xlinke
r -rpath -Xlinker /usr/tdbms/lib -Wl,--version-script=/var/opt/teradata/tdt
emp/UDFTemp/0460.30719.0833a/UserUdf_versions.scr -D_REENTRANT -D_LIBC_REEN
TRANT    -I/usr/tdbms/etc   -L/usr/tdbms/lib -L/usr/tdbms/lib  -o @FileList
   /usr/tdbms/lib/libmagen.so  -ludf  -lm -ljil -lstdc++
/usr/tdbms/lib/adpMaskTime.o: undefined symbol: _Z8maskTime8MaskTypePKcS1_S
1_S1_iPcPKi


Specifically:
undefined symbol: _Z8maskTime8MaskTypePKcS1_S1_S1_iPcPKi
```

7. Copy the "resources" folder from the dist folder to /usr/tdbms/

```
# MULTI NODE
pcl -send dist/resources /usr/tdbms/

# SINGLE NODE
cp -R dist/resources /usr/tdbms/
```

8. Copy the libmagen tar and adpMask tar files from the dist folder to /usr/tdbms/lib/:

```
# MULTI NODE
pcl -send dist/*.tgz /usr/tdbms/lib/

# SINGLE NODE
cp -a dist/*.tgz /usr/tdbms/lib/
```

9. Unpack libmagen tar and adpMask tar files to /usr/tdbms/lib:

```
# MULTI NODE
psh tar -xf /usr/tdbms/lib/adpMask*.tgz -C /usr/tdbms/lib/
psh tar -xf /usr/tdbms/lib/libmagen*.tgz -C /usr/tdbms/lib/

# SINGLE NODE
```

```
tar -xf /usr/tdbms/lib/adpMask*.tgz -C /usr/tdbms/lib/
tar -xf /usr/tdbms/lib/libmagen*.tgz -C /usr/tdbms/lib/
```

Optionally remove the tar files:

```
# MULTI NODE
psh rm /usr/tdbms/lib/adpMask*.tgz /usr/tdbms/lib/libmagen*.tgz

# SINGLE NODE
rm /usr/tdbms/lib/adpMask*.tgz /usr/tdbms/lib/libmagen*.tgz
```

10. Change the owner and group of copied folders and files appropriately (default is root:tdrusted). For example:

```
# MULTI NODE
psh chown -R root:tdtrusted /usr/tdbms/resources
psh chmod -R 555 /usr/tdbms/resources

psh chown root:tdtrusted /usr/tdbms/lib/libmagen*.so
psh chmod 555 /usr/tdbms/lib/libmagen*

psh chown root:tdtrusted  /usr/tdbms/lib/adpMask*.o
psh chmod 555 /usr/tdbms/lib/adpMask*.o


# SINGLE NODE
chown -R root:tdtrusted /usr/tdbms/resources
chmod -R 555 /usr/tdbms/resources

chown root:tdtrusted /usr/tdbms/lib/libmagen*.so
chmod 555 /usr/tdbms/lib/libmagen*

chown root:tdtrusted  /usr/tdbms/lib/adpMask*.o
chmod 555 /usr/tdbms/lib/adpMask*.o
```

11. Register the new UDFs after you log in to SQL with proper credentials:

```
cd scripts/

# Login to bteq using the following command.
bteq
.LOGON 127.0.0.1/WKC

# Script to register udfs.
 .RUN FILE=setup_udf.sql

# Script to register THROW stored procedure
 .RUN FILE=setup_sp.sql
```

12. Run all of the test scripts. If all tests pass, the UDF update is successful.

```
.RUN FILE=test/test_maskDouble.sql
.RUN FILE=test/test_maskChar.sql
.RUN FILE=test/test_maskIntShortLong.sql
.RUN FILE=test/test_numericShift.sql
.RUN FILE=test/test_maskNumeric.sql
.RUN FILE=test/test_maskFloat.sql
.RUN FILE=test/test_maskDateTime.sql
```