20
DBMS_CLOUD Family of Packages

To use the `DBMS_CLOUD` and other packages in the `DBMS_CLOUD` family of packages, you must complete certain tasks.

Starting with Oracle Database 23ai (23.7), you can install `DBMS_CLOUD` and other packages of the `DBMS_CLOUD` family with installation scripts deployed with Oracle Database. These packages are not preinstalled. You must manually install these packages, and also configure users or roles to use these packages.

- Using the DBMS_CLOUD Family of Packages
  Learn about the requirements for deploying and using the `DBMS_CLOUD` family of packages.
- Installing DBMS_CLOUD
  To use the `DBMS_CLOUD` family of packages on a customer-managed Oracle Database, you must create a new user and install `DBMS_CLOUD` packages as that user.
- Create SSL Wallet with Certificates
  To access HTTP URIs and Object Stores safely from within your database, you must create a wallet with the appropriate certificates.
- Configure Your Environment to Use the New SSL Wallet
  To have your SSL wallet take effect on your Oracle Database environment, you must point to the newly created SSL wallet.
- Configure the Database with ACEs for DBMS_CLOUD
  Create Access Control Entries (ACEs) to enable communication with Object Stores and other trusted `https` endpoints (URIs).
- Verify Configuration of DBMS_CLOUD
  After you verify that the `DBMS_CLOUD` code is correctly installed, verify the proper setup of the SSL Wallet and the Access Control Entities (ACEs).
- Configuring Users or Roles to use DBMS_CLOUD
  After successfully installing `DBMS_CLOUD`, you must configure users or roles to be able to use all of its supported functionality.

20.1 Using the DBMS_CLOUD Family of Packages

Learn about the requirements for deploying and using the `DBMS_CLOUD` family of packages.

The `DBMS_CLOUD` packages are preinstalled, configured, and maintained in Oracle Autonomous Database. However, to use the `DBMS_CLOUD` packages on a customer-managed Oracle Database, you must perform manual installation and configuration procedures. In comparison to the use of `DBMS_CLOUD` packages in Oracle Autonomous Database, these packages can offer only a subset of functionality available in Oracle Autonomous Database as a fully managed Cloud-native Oracle Database service with components beyond the core database.

Note:

For customer-managed, non-Autonomous Database uses of `DBMS_CLOUD`, see the documentation for the Oracle Database release. The installation in Oracle Database 19c, Oracle Database 21c, and earlier releases of Oracle Database 23ai is different than the processes for Oracle Database 23ai Release update 7 and later. For information about using `DBMS_CLOUD` with earlier releases, see *How To Setup And Use DBMS_CLOUD Package (Doc ID 2748362.1)*

DBMS_CLOUD Packages

The `DBMS_CLOUD` family of packages consists of the following:

- `DBMS_CLOUD`
- `DBMS_CLOUD_AI`
- `DBMS_CLOUD_NOTIFICATION`
- `DBMS_CLOUD_PIPELINE`
- `DBMS_CLOUD_REPO`

Overview of Installation and Configuration Steps

To set up `DBMS_CLOUD`, the following installation and configuration steps must be completed:

Installing and configuring `DBMS_CLOUD`:

- Create a schema owning the `DBMS_CLOUD` package, and install the `DBMS_CLOUD` code in the container database (CDB), and all pluggable databases (PDBs).
- Create a wallet to contain the certificates required to access HTTP URIs and Object Stores.
- Configure your Oracle Database environment to use the new SSL wallet.
- Configure your database with access control lists (ACLs) for `DBMS_CLOUD`.
- Verify the configuration of `DBMS_CLOUD`.

Configuring users or roles to use `DBMS_CLOUD`:

- Grant the minimal privileges to a user or role for using `DBMS_CLOUD`
- Configure ACLs for a user or role to use `DBMS_CLOUD`
- Verify the proper setup of your user or role for using `DBMS_CLOUD`

Related Topics

- DBMS_CLOUD in *Oracle Database PL/SQL Packages and Types Reference*

20.2 Installing DBMS_CLOUD

To use the `DBMS_CLOUD` family of packages on a customer-managed Oracle Database, you must create a new user and install `DBMS_CLOUD` packages as that user.

The default `DBMS_CLOUD` procedure installation is owned by a separate schema, the `C##CLOUD$SERVICE` schema. The schema is locked by default so that no connections are directly made as this user.

When you update to a release update (RU) has a new `DBMS_CLOUD` deployment, you must rerun the installation procedure on top of your existing procedure on the PDBs where you want to access the `DBMS_CLOUD` family of packages. The installation is written-idempotent, so you do not have to uninstall and reinstall the `DBMS_CLOUD` family of packages, but the user you create to administer this installation can connect to the schema.

To ensure correct installation of `DBMS_CLOUD` into any existing and future pluggable databases (PDBs), install the packages using the `catcon.pl` utility that is located in the directory *`Oracle home`*`/rdbms/admin/`. The code and installation scripts for `DBMS_CLOUD` are part of the Oracle distribution. The two main scripts are:

- `catclouduser.sql`: This script creates the schema `C##CLOUD$SERVICE` with the necessary privileges. Do not modify this script.
- `dbms_cloud_install.sql`: This script installs the `DBMS_CLOUD` packages in schema `C##CLOUD$SERVICE`. Do not modify this script.

Log in to the CDB where you want to install the `DBMS_CLOUD` packages, and use `catcon.pl` to perform the installation.

In the following example, the `DBMS_CLOUD` packages are installed, and the log files are configured to be created in the `/tmp` directory with the prefix `dbms_cloud_install`:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -u sys/your-
password -force_pdb_mode 'READ WRITE' -b dbms_cloud_install -d $ORACLE_HOME/
rdbms/admin/ -l /tmp catclouduser.sql
```

In the following example, the `DBMS_CLOUD` packages are installed in schema `C##CLOUD$SERVICE`:

```
$ORACLE_HOME/perl/bin/perl $ORACLE_HOME/rdbms/admin/catcon.pl -u sys/your-
password -force_pdb_mode 'READ WRITE' -b dbms_cloud_install -d $ORACLE_HOME/
rdbms/admin/ -l /tmp dbms_cloud_install.sql
```

After the installation is complete, check the log files for any errors. For example, you should see the package `DBMS_CLOUD` created and valid in both `ROOT` and any `PDB`.

To see the packages in `ROOT`, log in to SQL and run the following check:

```
select con_id, owner, object_name, status, sharing, oracle_maintained from
cdb_objects where object_name like 'DBMS_CLOUD%'
```

To see the packages in a PDB, log in to SQL and run the following check:

```
select owner, object_name, status, sharing, oracle_maintained from
dba_objects where object_name like 'DBMS_CLOUD%';
```

The installation will force all pluggable database to be open for the installation of `DBMS_CLOUD`, but the prior stage of a PDB will be retained after installation. Accordingly, these query checks will only show and work for open pluggable databases.

If the install logs show any error, or if you have any invalid objects owned by `C##CLOUD$SERVICE`, then you must analyze and correct these issues.

20.3 Create SSL Wallet with Certificates

To access HTTP URIs and Object Stores safely from within your database, you must create a wallet with the appropriate certificates.

You must manually install the appropriate certificates in a wallet to access the `DBMS_CLOUD` family of packages. The certificates are not part of the Oracle Database distribution. You can download the necessary certificates from the following site:

https://objectstorage.us-phoenix-1.oraclecloud.com/p/KB63IAuDCGhz_azOVQ07Qa_mxL3bGrFh1dtsltreRJPbmb-VwsH2aQ4Pur2ADBMA/n/adwcdemo/b/CERTS/o/dbc_certs.tar

The security wallet must have the following properties.

- The wallet must be created with auto-login capabilities.
- On Oracle Real Application Clusters (Oracle RAC) installations, the wallet must either be accessible for all nodes centrally, or you must create the wallet on all nodes for local wallet storage.

Oracle recommends that you store the SSL wallet in an equivalent location. In the following SSL wallet creation example, we assume that the SSL wallet is in the location `/u01/app/oracle/dcs/commonstore/wallets/ssl`, and you have unmpacked the certificates in the path `/home/oracle/dbc`:

```
cd /u01/app/oracle/dcs/commonstore/wallets/ssl
orapki wallet create -wallet . -pwd your_chosen_wallet_pw -auto_login

#! /bin/bash
for i in $(ls /home/oracle/dbc/*cer)
do
orapki wallet add -wallet . -trusted_cert -cert $i -pwd SSL Wallet password
done
```

Note:

If you are already having a wallet for SSL certificates, then you do not have to create a new wallet. Instead, you can add the required certificates to the existing wallet.

Oracle recommends that you check the certificate location. For example:

```
cd /u01/app/oracle/dcs/commonstore/wallets/ssl
orapki wallet display -wallet .
```

The following is an excerpt of what you should see in the certificate wallet. Note that this is not the complete list of all certificates:

```
[oracle@mydb ssl]$ orapki wallet display -wallet .

Oracle PKI Tool Release 21.0.0.0.0 - ProductionVersion 21.0.0.0.0 Copyright
```

```
(c) 2004, 2020, Oracle and/or its affiliates. All rights reserved.

Requested Certificates:
User Certificates:
Trusted Certificates:
Subject: CN=VeriSign Class 3 Public Primary Certification Authority -
G5,OU=(c) 2006 VeriSign\, Inc. - For authorized use only,OU=VeriSign Trust
Network,O=VeriSign\, Inc.,C=US
Subject: CN=Baltimore CyberTrust Root,OU=CyberTrust,O=Baltimore,C=IE
Subject: CN=DigiCert Global Root CA,OU=www.digicert.com,O=DigiCert Inc,C=US
```

20.4 Configure Your Environment to Use the New SSL Wallet

To have your SSL wallet take effect on your Oracle Database environment, you must point to the newly created SSL wallet.

To point to the SSL wallet, add it to your `SQLnet.ora` file on the OCI Server side. If you are on an Oracle Real Application Clusters (Oracle RAC) installation, then you must adjust the `SQLnet.ora` file on all nodes.

The location of the `SQLnet.ora` file that you update depends on your OCI deployment:

- Cloud installations without Oracle Grid Infrastructure: The default location of this file is `$ORACLE_HOME/network/admin`.
- Cloud installations with Oracle Grid infrastructure: The default location is `$GRID_HOME/network/admin`.

Note:

If you already had a wallet for SSL certificates and added the certificates to the existing wallet, then this step is not necessary.

```
WALLET_LOCATION=
(SOURCE=(METHOD=FILE)
(METHOD_DATA=(DIRECTORY=/u01/app/oracle/dcs/commonstore/wallets/ssl)))
```

You do not have to to restart the database listener.

20.5 Configure the Database with ACEs for DBMS_CLOUD

Create Access Control Entries (ACEs) to enable communication with Object Stores and other trusted `https` endpoints (URIs).

By default, Oracle Database does not permit outside communication. To provide access to Object Stores, you must enable the appropriate Access Control Entries. If your database is behind a firewall, then you must provide information about your Internet Gateway, and configure the Access Control Entries (ACEs) appropriately.

If you are using an HTTP proxy to connect to the Internet, then you must configure your database to enable secure use of your gateway. This configuration process requires you to enable your database to access external network services through the gateway, and then configure your database to use the HTTP proxy gateway for `DBMS_CLOUD` external network services.

1. Enable your database to access to the external network services through the gateway, so that the database can access the Object Store.

   To allow access to your HTTP proxy gateway for external network services for the schema-owning `DBMS_CLOUD`, to append the access control list of your database using the parameter `DBMS_NETWORK_ACL_ADMIN` package with the `APPEND_HOST_ACE` procedure, where *`your-proxy-host-DNS-name`* is the name or IP address of your HTTP proxy gateway host:

   ```
   host=your-proxy-host-DNS-name
   ```

   For example, if your HTTP proxy setting is `http://myproxyhost.mydomain:99`, then enter `'myproxyhost.mydomain'`.

   ```
   low_port=your_proxy_low_port
   high_port=your_proxy_high_port
   ```

   Those two parameters can be null or a port number. By default, there is no port restriction for TCP connections. To limit the access to a specific port your HTTP proxy is communicating on, you can use the same port as both the low and high port. In the example that follows, both of these parameters are set to port 99.

2. Configure your database to use the HTTP proxy gateway for `DBMS_CLOUD` external network services.

   `DBMS_CLOUD` internally recursively issues REST calls leveraging `UTL_HTTP`. The proxy URI information for `DBMS_CLOUD` is set with the database property `'http_proxy'`, following the proxy URI format as set with `UTL_HTTP.SET_PROXY(). proxy_uri=`*`your-proxy-URI-address`*. The proxy can include an optional TCP/IP port number on which the proxy server listens. The syntax is `http://`*`host`*`:`*`port`*. For example: `www-proxy.my-company.com:80`. If the port is not specified for the proxy, then by default port 80 is used.

   Optionally, you can specify a port number for each domain or host. If the port number is specified, then the no-proxy restriction is only applied to the request at the port of the particular domain or host. For example: `corp.my-company.com, eng.my-company.com:80`.

   When `no_proxy_domains` is `NULL` and the proxy is set, all requests go through the proxy. When the proxy is not set, `UTL_HTTP` sends requests to the target Web servers directly.

   You can define a user name and password for the proxy that you want to be specified in the proxy string. The format is `http://`*`user`*`:`*`password@host`*`:`*`port`*. For more details about configuring access control for external network services using the `DBMS_NETWORK_ACL_ADMIN` package, see the "Syntax for Configuring Access Control for External Network Services" section link at the bottom of this topic.

   To configure the database, wrap the commands into a SQL script and run the commands in your multitenant environment by connecting to the `CDB$ROOT` container as `SYS`. Create the script by using the `sqlsessstart.sql` template script, which is located in the path `$ORACLE_HOME/rdbms/admin/sqlsessend.sql`. Save a version of the script customized for your environment, and run that script.

Example 20-1 Configure database to use the HTTP and HTTP_PROXY for DBMS_CLOUD

Cut and paste the entire content in this code example into a new SQL script (for example, `configure_cloud_user.sql`), and update as required for your environment. This code example contains comments in the script itself that explain how the proxy URL and host values are set. When you configure the script with your own values, you can then run the script in your multitenant environment by connecting to the `CDB$ROOT` container as `SYS`.

Note:

Ensure that you set variables for your environment correctly. If you do not set them correctly, then `DBMS_CLOUD` will not function properly.

```
@$ORACLE_HOME/rdbms/admin/sqlsessstart.sql

-- you must not change the owner of the functionality to avoid future issues
define clouduser=C##CLOUD$SERVICE

-- CUSTOMER SPECIFIC SETUP, NEEDS TO BE PROVIDED BY THE CUSTOMER-- - SSL 
Wallet directory
define sslwalletdir=<Set SSL Wallet Directory>

---- UNCOMMENT AND SET THE PROXY SETTINGS VARIABLES IF YOUR ENVIRONMENT NEEDS 
PROXYS--

-- define proxy_uri=<your proxy URI address>
-- define proxy_host=<your proxy DNS name>
-- define proxy_low_port=<your_proxy_low_port>
-- define proxy_high_port=<your_proxy_high_port>

-- Create New ACL / ACE s
begin
-- Allow all hosts for HTTP/HTTP_PROXY
    dbms_network_acl_admin.append_host_ace(
        host =>'*',
        lower_port => 443,
        upper_port => 443,
        ace => xs$ace_type(
            privilege_list => xs$name_list('http', 'http_proxy'),
            principal_name => upper('&clouduser'),
            principal_type => xs_acl.ptype_db
            )
        );
--
-- UNCOMMENT THE PROXY SETTINGS SECTION IF YOUR ENVIRONMENT NEEDS PROXYS
--
-- Allow Proxy for HTTP/HTTP_PROXY
-- dbms_network_acl_admin.append_host_ace(
-- host =>'&proxy_host',
-- lower_port => &proxy_low_port,
-- upper_port => &proxy_high_port,
-- ace => xs$ace_type(
-- privilege_list => xs$name_list('http', 'http_proxy'),
-- principal_name => upper('&clouduser'),
-- principal_type => xs_acl.ptype_db));
--
-- END PROXY SECTION
```

```
--

-- Allow wallet access
    dbms_network_acl_admin.append_wallet_ace(
        wallet_path => 'file:&sslwalletdir',
        ace => xs$ace_type(
            privilege_list =>xs$name_list('use_client_certificates',
'use_passwords'),
            principal_name => upper('&clouduser'),
            principal_type => xs_acl.ptype_db));
end;
/

-- Setting SSL_WALLET database property
begin
    if sys_context('userenv', 'con_name') = 'CDB$ROOT' then
        execute immediate 'alter database property set
ssl_wallet=''&sslwalletdir''';
--
-- UNCOMMENT THE FOLLOWING COMMAND IF YOU ARE USING A PROXY
--
--        execute immediate 'alter database property set
http_proxy=''&proxy_uri''';
    end if;
end;
/

@$ORACLE_HOME/rdbms/admin/sqlsessend.sql
```

Assuming you save a modified version of the script with your environment values named `dbc_aces.sql` in a working directory called `dbc` under the home directory `/home/oracle`, you then run the following command to configure your database:

```
# Connect to CDB$ROOT
connect sys/your-password as sysdba
@@/home/oracle/dbc/dbc_aces.sql
```

After running the script, confirm that the setup is correct for your environment:

- You should not see any entry for `HTTP_PROXY` if your environment does not need one.
- The property `SSL_WALLET` should show the directory where your wallet is located.

Related Topics

- Syntax for Configuring Access Control for External Network Services
- APPEND_HOST_ACE Procedure
- SET_PROXY Procedure

20.6 Verify Configuration of DBMS_CLOUD

After you verify that the `DBMS_CLOUD` code is correctly installed, verify the proper setup of the SSL Wallet and the Access Control Entities (ACEs).

Wrap into a SQL script the commands shown in the example that follows, and run the script as the user `SYS` either within the CDB or in any PDB.

Note:

Ensure that you have set the variables for your environment appropriately. If you do not set them correctly then this example procedure will not work, independent of whether or not you have set up `DBMS_CLOUD` correctly.

```
define clouduser=C##CLOUD$SERVICE

-- CUSTOMER SPECIFIC SETUP, NEEDS TO BE PROVIDED BY THE CUSTOMER
-- - SSL Wallet directory and password
define sslwalletdir=<Set SSL Wallet Directory>
define sslwalletpwd=<Set SSL Wallet password>

-- In environments w/ a proxy, you need to set the proxy in the verification
code
-- define proxy_uri=<your proxy URI address>

-- create and run this procedure as owner of the ACLs, which is the future
owner
-- of DBMS_CLOUD

CREATE OR REPLACE PROCEDURE &clouduser..GET_PAGE(url IN VARCHAR2) AS
    request_context UTL_HTTP.REQUEST_CONTEXT_KEY;
    req UTL_HTTP.REQ;
    resp UTL_HTTP.RESP;
    data VARCHAR2(32767) default null;
    err_num NUMBER default 0;
    err_msg VARCHAR2(4000) default null;

BEGIN

-- Create a request context with its wallet and cookie table
    request_context := UTL_HTTP.CREATE_REQUEST_CONTEXT(
        wallet_path => 'file:&sslwalletdir',
        wallet_password => '&sslwalletpwd');

-- Make a HTTP request using the private wallet and cookie
-- table in the request context

-- uncomment if proxy is required
--    UTL_HTTP.SET_PROXY('&proxy_uri', NULL);

    req := UTL_HTTP.BEGIN_REQUEST(url => url,request_context =>
request_context);
```

```
    resp := UTL_HTTP.GET_RESPONSE(req);

DBMS_OUTPUT.PUT_LINE('valid response');

EXCEPTION
    WHEN OTHERS THEN
        err_num := SQLCODE;
        err_msg := SUBSTR(SQLERRM, 1, 3800);
        DBMS_OUTPUT.PUT_LINE('possibly raised PLSQL/SQL error: ' ||err_num||'
- '||err_msg);

        UTL_HTTP.END_RESPONSE(resp);
        data := UTL_HTTP.GET_DETAILED_SQLERRM ;
        IF data IS NOT NULL THEN
            DBMS_OUTPUT.PUT_LINE('possibly raised HTML error: ' ||data);
        END IF;
END;
/

set serveroutput on
BEGIN
    &clouduser..GET_PAGE('https://objectstorage.eu-
frankfurt-1.oraclecloud.com');
END;
/

set serveroutput off
drop procedure &clouduser..GET_PAGE;
```

If you have properly configured the SSL wallet and set up your database environment, then the script will return "valid response" and you can successfully connect to the Oracle Object Store.

If you receive an error, then your installation was not done properly. Correct any possible errors before continuing. If you cannot successfully access the example page, then you will not be able to access any Object Storage either.

20.7 Configuring Users or Roles to use DBMS_CLOUD

After successfully installing `DBMS_CLOUD`, you must configure users or roles to be able to use all of its supported functionality.

- Grant Minimal Privileges to a User or Role for DBMS_CLOUD
  For a user or role to use DBMS_CLOUD functionality, you have to grant at least minimal access privileges.
- Configure ACEs for a User or Role to Use DBMS_CLOUD
  To provide all the functionality of `DBMS_CLOUD` to a user or role, you must enable the appropriate Access Control Entries (ACEs).
- Verify Setup of Users and Roles to Use DBMS_CLOUD
  When user and roles are set up correctly, you can create credentials and access data in the Object Store.

20.7.1 Grant Minimal Privileges to a User or Role for DBMS_CLOUD

For a user or role to use DBMS_CLOUD functionality, you have to grant at least minimal access privileges.

The privileges shown in the examples that follows are required for a user or role to use `DBMS_CLOUD` functionality. To make the management of the necessary privileges easier for multiple users, Oracle recommends that you grant the necessary privileges through a role.

Example 20-2 Granting Privileges Using a Local Role

This example script uses a local role, `CLOUD_USER_ROLE`, and grants privileges to a local user, `SCOTT`. You can modify this script as needed for your PDB environment, and run the script within your pluggable database as a privileged administrator (for example, `SYS` or `SYSTEM`).

```
set verify off

-- target example role
define userrole='CLOUD_USER_ROLL'

-- target sample user
define username='SCOTT'

create role &userrole;
grant &userrole to &username;

REM the following are minimal privileges to use DBMS_CLOUD
REM - this script assumes core privileges
REM - CREATE SESSION
REM - Tablespace quota on the default tablespace for a user

REM for creation of external tables, e.g. DBMS_CLOUD.CREATE_EXTERNAL_TABLE()
grant CREATE TABLE to &userrole;

REM for using COPY_DATA()
REM - Any log and bad file information is written into this directory
grant read, write on directory DATA_PUMP_DIR to &userrole;

REM grant as you see fit
grant EXECUTE on dbms_cloud to &userrole;
grant EXECUTE on dbms_cloud_pipeline to &userrole;
grant EXECUTE on dbms_cloud_repo to &userrole;
grant EXECUTE on dbms_cloud_notification to &userrole;
```

Example 20-3 Granting Privileges to an Individual User

You can choose to grant DBMS_CLOUD privileges to an individual user. In this example script, privileges are granted to local user `SCOTT`. You can modify this script as needed for your PDB environment.

```
set verify off

-- target sample user
define username='SCOTT'
```

```
REM the following are minimal privileges to use DBMS_CLOUD
REM - this script assumes core privileges
REM - CREATE SESSIONREM - Tablespace quota on the default tablespace for a
user

REM for creation of external tables, e.g. DBMS_CLOUD.CREATE_EXTERNAL_TABLE()
grant CREATE TABLE to &username;

REM for using COPY_DATA()
REM - Any log and bad file information is written into this directory
grant read, write on directory DATA_PUMP_DIR to &username;

REM grant as you see fit
grant EXECUTE on dbms_cloud to &username;
grant EXECUTE on dbms_cloud_pipeline to &username;
grant EXECUTE on dbms_cloud_repo to &username;
grant EXECUTE on dbms_cloud_notification to &username;
```

20.7.2 Configure ACEs for a User or Role to Use DBMS_CLOUD

To provide all the functionality of `DBMS_CLOUD` to a user or role, you must enable the appropriate Access Control Entries (ACEs).

The `DBMS_CLOUD` family of packages have the `INVOKER` right privilege. For that reason, it is necessary to enable the appropriate access control entries (ACEs) to enable a user or role to obtain all the functionality of the `DBMS_CLOUD` family of packages. These ACEs are similar to the ones for `DBMS_CLOUD`.

To facilitate the management of these privileges for multiple users, Oracle recommends that you grant the necessary privileges through a role.

Example 20-4 Granting Access Privileges Using a Role

This example script shows the commands necessary to enable DBMS_CLOUD functionality. Wrap these commands into a SQL script and run the script either in the CDB or the PDB as `SYS` where you want to provide DBMS_CLOUD functionality to your user or role.

The example script uses a local role, `CLOUD_USER`, and grants privileges to a local user, `SCOTT`. You can modify this script as needed for your PDB environment. Run the script as a privileged administrator within your PDB (for example, `SYS` or `SYSTEM`)

```
@$ORACLE_HOME/rdbms/admin/sqlsessstart.sql

-- target sample roledefine cloudrole=CLOUD_USER

-- CUSTOMER SPECIFIC SETUP, NEEDS TO BE PROVIDED BY THE CUSTOMER
-- - SSL Wallet directory
define sslwalletdir=<Set SSL Wallet Directory>

---- UNCOMMENT AND SET THE PROXY SETTINGS VARIABLES IF YOUR ENVIRONMENT NEEDS
PROXYS
--
-- define proxy_uri=<your proxy URI address>
-- define proxy_host=<your proxy DNS name>
-- define proxy_low_port=<your_proxy_low_port>
-- define proxy_high_port=<your_proxy_high_port>
-- Create New ACL / ACEs
begin
-- Allow all hosts for HTTP/HTTP_PROXY
    dbms_network_acl_admin.append_host_ace(
        host =>'*',
        lower_port => 443,
        upper_port => 443,
        ace => xs$ace_type(
            privilege_list => xs$name_list('http', 'http_proxy'),
            principal_name => upper('&cloudrole'),
            principal_type => xs_acl.ptype_db));

--
-- UNCOMMENT THE PROXY SETTINGS SECTION IF YOUR ENVIRONMENT NEEDS PROXYS
--
-- Allow Proxy for HTTP/HTTP_PROXY
-- dbms_network_acl_admin.append_host_ace(
-- host =>'&proxy_host',
-- lower_port => &proxy_low_port,
-- upper_port => &proxy_high_port,
-- ace => xs$ace_type(
-- privilege_list => xs$name_list('http', 'http_proxy'),
-- principal_name => upper('&cloudrole'),
-- principal_type => xs_acl.ptype_db));
--
-- END PROXY SECTION
--

-- Allow wallet access
    dbms_network_acl_admin.append_wallet_ace(
        wallet_path => 'file:&sslwalletdir',
        ace => xs$ace_type(
            privilege_list =>xs$name_list('use_client_certificates',
'use_passwords'),
            principal_name => upper('&cloudrole'),
            principal_type => xs_acl.ptype_db));
end;
/

@$ORACLE_HOME/rdbms/admin/sqlsessend.sql
```

Example 20-5   Granting Access Privileges to an Individual User

In this example script, we assume local user `SCOTT` has been created with `DBMS_CLOUD` privileges, as shown previously, and you are now granting access privileges to that user. You can modify this script as needed for your PDB environment.

```
@$ORACLE_HOME/rdbms/admin/sqlsessstart.sql

-- target sample user
define clouduser=SCOTT

-- CUSTOMER SPECIFIC SETUP, NEEDS TO BE PROVIDED BY THE CUSTOMER
-- - SSL Wallet directory
define sslwalletdir=<Set SSL Wallet Directory>
```

```
-- Proxy definition
-- define proxy_uri=<your proxy URI address>
-- define proxy_host=<your proxy DNS name>
-- define proxy_low_port=<your_proxy_low_port>
-- define proxy_high_port=<your_proxy_high_port>

-- Create New ACL / ACEs
begin
-- Allow all hosts for HTTP/HTTP_PROXY
    dbms_network_acl_admin.append_host_ace(
        host =>'*',
        lower_port => 443,
        upper_port => 443,
        ace => xs$ace_type(
            privilege_list => xs$name_list('http', 'http_proxy'),
            principal_name => upper('&clouduser'),
            principal_type => xs_acl.ptype_db));

--
-- UNCOMMENT THE PROXY SETTINGS SECTION IF YOUR ENVIRONMENT NEEDS PROXYS
--
-- Allow Proxy for HTTP/HTTP_PROXY
-- dbms_network_acl_admin.append_host_ace(
-- host =>'&proxy_host',
-- lower_port => &proxy_low_port,
-- upper_port => &proxy_high_port,
-- ace => xs$ace_type(
-- privilege_list => xs$name_list('http', 'http_proxy'),
-- principal_name => upper('&clouduser'),
-- principal_type => xs_acl.ptype_db));
--
-- END PROXY SECTION
--

-- Allow wallet access
    dbms_network_acl_admin.append_wallet_ace(
        wallet_path => 'file:&sslwalletdir',
        ace => xs$ace_type(
            privilege_list =>xs$name_list('use_client_certificates',
'use_passwords'),
            principal_name => upper('&clouduser'),
            principal_type => xs_acl.ptype_db));
end;
/

@$ORACLE_HOME/rdbms/admin/sqlsessend.sql
```

After you run the access privileges scripts, your user or role previously granted minimal `DBMS_CLOUD` privileges is now properly configured and enabled to use the `DBMS_CLOUD` family packages.

20.7.3 Verify Setup of Users and Roles to Use DBMS_CLOUD

When user and roles are set up correctly, you can create credentials and access data in the Object Store.

To access data in the Object Store that is not public, you need to authenticate with an OCI user in your tenancy who has appropriate privileges to the object storage bucket in the region in question. You need to create either an OCI API signing key or an auth token for a user in your tenancy. For details about access to the Oracle Cloud Infrastructure (OCI) Object store, see:

https://docs.oracle.com/en-us/iaas/Content/Identity/Tasks/managingcredentials.htm

Example 20-6 Create Credential Object and Access the Object Store

Assuming you have created an authorization token (`auth`), you must create a credential object in your database schema for authentication. For example:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'your credential name',
        username => 'OCI within your tenancy',
        password => 'auth token generated for OCI user');
END;
/
```

After the creation of your credential object, you should now be able to access the Object Store bucket in your tenancy for which the OCI user in your tenancy has privileges. Replace the credential name, region, object storage name space. and bucket name with the correct values for your tenancy:

```
select * from dbms_cloud.list_objects('CredentialName','https://
objectstorage.region.oraclecloud.com/n/ObjectStorageNameSpace/b/BucketName/o/');
```

Example 20-7 Validate User Configuration and Privilege (accessibility of wallet, privilege to use wallet, database-wide setting of wallet)

If you encounter problems with `DBMS_CLOUD` with the user or role you have configured, you can test the proper configuration of your environment without `DBMS_CLOUD` by using the same example code used for the DBMS_CLOUD setup with the user or role that you configured.

Assuming you set up a user named `SCOTT`, wrap the following commands into a SQL script and execute it as SYS in the pluggable database you were configuring. Be aware of the following requirements to use the script example:

- Set the variables for your environment appropriately. If you do not set them correctly, then the example procedure will not work, independent of whether or not you have set up your user or role correctly.
- To use the example code you rquire additional privileges for your user or role. Specifically you require name EXECUTE on UTL_HTTP. If your user or role does not have this privilege, then you must grant it temporarily to run this code successfully. If you have granted the ACLs through a role, then you must grant those privileges explicitly to user `SCOTT` for this example to work

```
-- user to troubleshoot
define clouduser=SCOTT
```

```
-- CUSTOMER SPECIFIC SETUP, NEEDS TO BE PROVIDED BY THE CUSTOMER
-- - SSL Wallet directory and password
define sslwalletdir=<Set SSL Wallet Directory>
define sslwalletpwd=<Set SSL Wallet password>

-- In environments w/ a proxy, you need to set the proxy in the verification
code
-- define proxy_uri=<your proxy URI address>

-- create and run this procedure as owner of the ACLs, which is the future
owner
-- of DBMS_CLOUD
CREATE OR REPLACE PROCEDURE &clouduser..GET_PAGE(url IN VARCHAR2)
AS
    request_context UTL_HTTP.REQUEST_CONTEXT_KEY;
    req UTL_HTTP.REQ;
    resp UTL_HTTP.RESP;
    data VARCHAR2(32767) default null;
    err_num NUMBER default 0;
    err_msg VARCHAR2(4000) default null;

BEGIN

-- Create a request context with its wallet and cookie table
request_context := UTL_HTTP.CREATE_REQUEST_CONTEXT(wallet_path =>
'file:&sslwalletdir',wallet_password => '&sslwalletpwd');

-- Make a HTTP request using the private wallet and cookie
-- table in the request context

-- uncomment if proxy is required
--    UTL_HTTP.SET_PROXY('&proxy_uri', NULL);

req := UTL_HTTP.BEGIN_REQUEST(url => url,request_context => request_context);
resp := UTL_HTTP.GET_RESPONSE(req);

DBMS_OUTPUT.PUT_LINE('valid response');

EXCEPTION
WHEN OTHERS THEN
    err_num := SQLCODE;
    err_msg := SUBSTR(SQLERRM, 1, 3800);
    DBMS_OUTPUT.PUT_LINE('possibly raised PLSQL/SQL error: ' ||err_num||' -
'||err_msg);

    UTL_HTTP.END_RESPONSE(resp);
    data := UTL_HTTP.GET_DETAILED_SQLERRM ;
    IF data IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('possibly raised HTML error: ' ||data);
    END IF;
END;
/

set serveroutput on
BEGIN
    &clouduser..GET_PAGE('https://objectstorage.eu-
frankfurt-1.oraclecloud.com');
END;
/

set serveroutput off
drop procedure &clouduser..GET_PAGE;
```

Correct any errors in the configuration. This procedure will run successfully if you have configured your user or role correctly.