Installing DATACONS_SFCONNECTOR Pre-Requisites

To use connector following you must follow these pre-requisites:

- Oracle Cloud
  - Oracle Database 18c (min. supported version)
- On-Premise Oracle Database
  - Oracle Database 18c (min. supported version)
  - Oracle Wallet with SSL certificates for Snowflake sites has to be set up (see sections bellow)
  - Network ACL for snowflake source has to be set up

Oracle Cloud installation

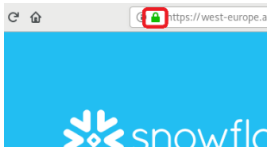
Use script DATACONS_SFCONNECTOR_INSTALL_ATP.sql

Oracle On-Premises installation

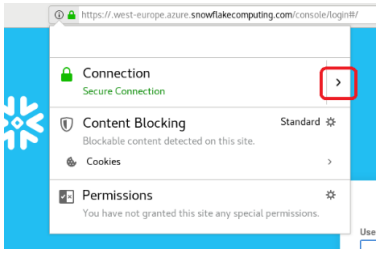
SSL certificates

Before using the Connector on On-Premise Oracle Database, it is required to obtain Snowflake SSL certificates. Bellow you can find an example of steps on how to obtain the certificates.

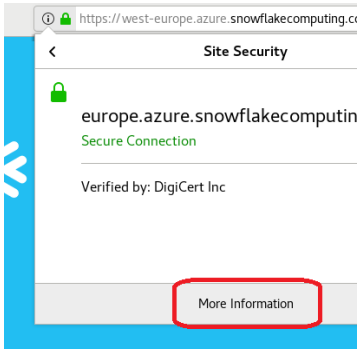
1. Go to Snowflake web console: https://<account_name>.snowflakecomputing.com
2. Click green lock next to url



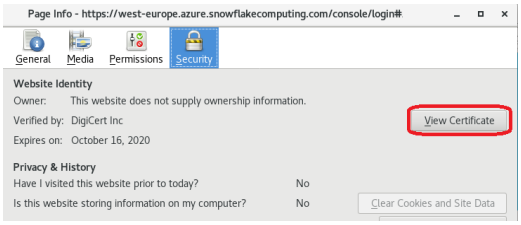
3. Click arrow on right site of Connection section



4. Click: More Information

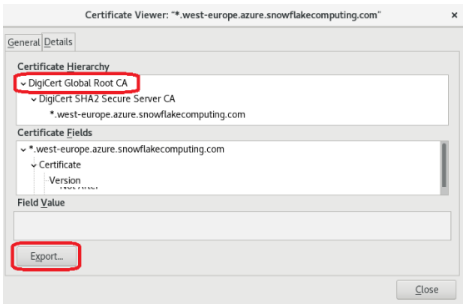


5. Click button: View Certificate



6. Click tab: Details

7. Select root certificate and click button Export to save the certificate on disk



To use multi chunk queries you need to obtain certificates for Object Store of your Vendor (AWS S3, BLOB Azure). For further instructions contact us: office@datacons.co.uk.

Setup Oracle Wallet

Before using the Connector on On-Premise Oracle Database, it is required to setup Oracle Wallet to store SSL certificates for Snowflake connection. Bellow you can find an example of how to set up Oracle Wallet.

Setting up Oracle Wallet:

```
$ mkdir -p sample_wallet
$ orapki wallet create -wallet /home/oracle/sample_wallet \
 -pwd some_password -auto_login
$ orapki wallet add -wallet /home/oracle/sample_wallet \
 -trusted_cert -cert AmazonRootCA1.crt -pwd some_password
```

Verify certificates installed in Oracle Wallet:

```
$ orapki wallet display -wallet /home/oracle/sample_wallet
```

Installing DATACONS_SFCONNECTOR

For On-Premise Oracle Database:

```
SQL> @DATACONS_SFCONNECTOR_INSTALL_ATP.sql
```

Setup Network ACL

Before using the Connector on On-Premise Oracle Database, it is required to setup Network ACL in the Database. Bellow you can find an example of how to set up Network ACL allowing all network traffic.

```
DECLARE
  l_principal VARCHAR2(20) := 'SAMPLE_SCHEMA';
BEGIN
  DBMS_NETWORK_ACL_ADMIN.CREATE_ACL (
    acl => 'http_permissions.xml',
    description => 'HTTP Access',
    principal => l_principal,
    is_grant => TRUE,
    privilege => 'connect',
    start_date => null,
    end_date => null
  );
  DBMS_NETWORK_ACL_ADMIN.ADD_PRIVILEGE(acl => 'http_permissions.xml',
    principal => l_principal,
    is_grant => true,
    privilege => 'connect'
  );
  DBMS_NETWORK_ACL_ADMIN.ADD_PRIVILEGE(acl => 'http_permissions.xml',
    principal => l_principal,
    is_grant => true,
    privilege => 'resolve'
  );
  DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL (
    acl => 'http_permissions.xml',
    host => '*', --adjust to your needs
    lower_port => null,
    upper_port => null
  );
END;
/
```

Using the DATACONS_SFCONNECTOR

Establishing a Connection

Procedure: login

To execute SQL on Snowflake from Oracle Database it is necessary to establish connection to Snowflake first. To establish the connection to Snowflake use procedure: datacons_sfconnector.login.

```
procedure login(pi_account  in varchar2,
                pi_user     in varchar2,
                pi_password in varchar2,
                pi_wallet   in varchar2);
```

Parameter	Description
pi_account	Snowflake account name
pi_user	Snowflake user
pi_password	Snowflake user password
pi_wallet	Oracle wallet location

Example:

```
SQL> exec datacons_sfconnector.login('some_account.west-europe.azure',
                                     'some_login',
                                     'some_password',
                                     'file:/home/oracle/some_wallet');
```

Query Data

query

To run a SQL query on Snowflake from Oracle Database use function datacons_sfconnector.query. The results will be piped as JSON rows.

Note: you can also run DML/DDL using function datacons_sfconnector.query.

```
function query(pi_query             in varchar2 default null,
               pi_binds             in json_key_list default null,
               pi_wait_for_result   in varchar2 default 1,
               pi_query_id          in varchar2 default null)
 return t_json_tab  pipelined;
```

Parameter	Description
pi_query	SQL statement
pi_binds	Binds for SQL statement
pi_wait_for_result	Flag wait for results 1 (default) - function will run until results are received 0 - function will throw an exception when query in Snowflake become asynchronous query. In exception details Query Id is provided to allow further monitoring and retrieve results
pi_query_id	Query Id. When provided allows to retrieve results for query. Parametrs pi_query_id and pi_query must not be used together.

Example 1: Running SQL query

```
select
     json_value(column_value,'$[0]') AS o_orderkey
    ,json_value(column_value,'$[1]') AS o_orderstatus
    ,json_value(column_value,'$[2]') AS o_totalprice
    ,json_value(column_value,'$[3]') AS o_comment
from table(datacons_sfconnector.query(
    pi_query => q'[SELECT
                     o_orderkey
                    ,o_orderstatus
                    ,o_totalprice
                    ,o_comment
                  FROM orders o
                  WHERE o_orderdate BETWEEN to_date(?,'YYYY-MM-DD') AND
                                            to_date(?,'YYYY-MM-DD')]'
    , pi_binds => json_key_list('1998-01-01','1998-12-31')));
```

Example 2: Running DML using datacons_sfconnector.query and returning number of inserted rows.

```
select *
from table(datacons_sfconnector.query(
   pi_query => 'INSERT INTO sample_table (id, data) VALUES (?,?)',
   pi_binds => json_key_list('1','New row')));
```

Example 3: Connecting to asynchronous query using query id.

```
select
   json_value(column_value,'$[0]') as id
  ,json_value(column_value,'$[1]') as data
from table(datacons_sfconnector.query(pi_query_id => '00000000-0000-0000-0000-000000000000'));
```

Running Statements

execute

To execute SQL statements on Snowflake from Oracle Database use procedure datacons_sfconnector.execute.

To execute bulk insert on Snowflake from Oracle Database use overloaded procedure datacons_sfconnector.execute allowing to pass json_array as a multi bind argument.

Overload 1

```
procedure execute(pi_stmt   in varchar2);
```

Parameter	Description
pi_stmt	SQL statement

Overload 2

```
procedure execute(pi_stmt   in varchar2,
                  pi_binds  in json_key_list);
```

Parameter	Description
pi_stmt	SQL statement
pi_binds	Binds for SQL statement

Overload 3

```
procedure execute(pi_stmt   in varchar2,
                  pi_binds  in json_array_t);
```

Parameter	Description
pi_stmt	SQL statement
pi_binds	Binds for SQL statement (JSON Array)

Example 1: Running DML

```
begin
  datacons_sfconnector.execute(pi_stmt => 'INSERT INTO sample_table (id, data) VALUES (?,?)',
                               pi_binds => json_key_list('1','New row'));
end;
/
```

Example 2: Running bulk insert

```
begin
  datacons_sfconnector.execute(pi_stmt => 'INSERT INTO sample_table (id, data) VALUES (?,?)',
                               pi_binds => json_array_t('[[2,"New row 2"],[3,"New row 3"]]'));
end;
/
```

Getting query status

get_query_status

To check query status on Snowflake from Oracle Database use function datacons_sfconnector.get_query_status.

```
function get_query_status(pi_query_id  in varchar2) return varchar2;
```

Parameter	Description
pi_query_id	Query Id for query that status to be checked

Example:

```
select
  datacons_sfconnector.get_query_status('00000000-0000-0000-0000-000000000000')
from dual;
```

Additional Configuration Options

set_max_wait_for_async_query

To configure maximum wait time in Oracle session for an asynchronous Snowflake query use procedure datacons_sfconnector.set_max_wait_for_async_query.

```
procedure set_max_wait_for_async_query(pi_seconds in pls_integer);
```

Parameter	Description
pi_seconds	Approximate time-out time in seconds. After pi_seconds query that runs asynchronously and wait for results will rise an exception. See exception details for a Query Id.

Example:

```
exec datacons_sfconnector.set_max_wait_for_async_query(60);
```

get_max_wait_for_async_query

To get maximum wait time in Oracle session for an asynchronous Snowflake in seconds use function datacons_sfconnector.get_max_wait_for_async_query

```
function get_max_wait_for_async_query return pls_integer;
```

Example:

```
select datacons_sfconnector.get_max_wait_for_async_query from dual;
```

set_verbose

To configure connector verbose mode use procedure datacons_sfconnector.set_verbose.

```
procedure set_verbose(pi_value in pls_integer);
```

Parameter	Description
pi_value	When set to 1 some information will be written to server output.

Example:

```
exec datacons_sfconnector.set_verbose(1);
```

get_verbose

To get connector verbose status use function datacons_sfconnector.get_verbose

```
function get_verbose return pls_integer;
```

Example:

```
select datacons_sfconnector.get_verbose from dual;
```