# Oracle Health Insurance Back Office

## Data Management Technical Reference

version 3.6

Part number: F50672-01

January 14, 2022

# CHANGE HISTORY

| Release | Version | Changes |
|---|---|---|
| 10.15.1.0.0 | 1.4 | • Changes for database 12C<br>• Small textual changes |
| 10.15.3.0.0 | 1.5 | • Added remark for VPD and the created database user for SS/DM<br>• Added advise for parameter parallel_degree_level for subset export as well as disabling archivelog mode |
| 10.16.1.0.0 | 1.6 | • Added remark about deferred segment creation<br>• Added OHI Data Marts support |
| 10.16.2.0.0 | 1.7 | • Added changes for OEM 13C<br>• Added work-around for subset definition import bug 19828552 |
| 10.16.2.2.0 | 1.8 | • Pre-processing masking script SDM0001S.pl made compulsory (bug 25252355) |
| 10.17.1.0.0 | 1.9 | • Added paragraph 4.2 about masking flex fields |
| 10.17.1.3.0 | 2.0 | • Adjust paragraph 3.4, adding creating the subset rule definition file<br>• Removed workaround for bug 23249155 |
| 10.17.2.0.0 | 2.1 | • Updated prerequisites; OEM 13C R2 is now certified |
| 10.18.1.0.0 | 2.2 | • Added masking voor OHI Data Marts<br>• Updated prerequisites; Supported version of database plugin changed to 13.2.2.0 |
| 10.18.2.0.0 | 2.3 | • No changes except part number. |
| 10.18.2.2.0 | 2.4 | • Slightly adjusted the recommendations for IO calibration |
| 10.18.2.3.0 | 2.5 | • Added pre masking step<br>• Added 2 known issues |
| 10.19.1.0.0 | 2.6 | • Added option for configuration only subset<br>• Some minor textual corrections and changes |
| 10.19.1.1.0 | 2.7 | • Updated prerequisites; OEM 13C R3 is now certified |
| 10.19.2.0.0 | 2.8 | • No changes, republished with new part number. |
| 10.20.1.0.0 | 2.9 | • No changes, republished. |
| 10.20.3.0.0 | 3.0 | • Adapted for impact of DB 19c certification |
| 10.20.6.0.0 | 3.1 | • Small improvements.<br>• Added the step-by-step execution guide appendix. |

| Release | Version | Changes |
|---|---|---|
| 10.20.8.0.0 | 3.2 | • Added a note to state that the OHI BO Data Management data subsetting and masking processes cannot be executed on a database instance (PDB) with an active Oracle Database Vault installation.<br>• Updated prerequisites; OEM 13C R4 is now certified |
| 10.21.1.0.0 | 3.3 | • No changes, republished with new part number. |
| 10.21.4.0.0 | 3.4 | • Data masking revised |
| 10.21.5.0.0 | 3.5 | • Modified Datamasking instructions<br>• Added Apendix 2 “Masking a custom database schema” |
| 10.22.1.0.0 | 3.6 | • No changes, republished with new part number. |

# Contents

# 1 Introduction

Welcome to the technical reference guide for the Oracle Health Insurance Back Office suite (OHI BO and OHI DM) Data Management application. This reference guide is intended to help you in using the data subsetting and data masking functionality as provided by the OHI Data Management product for OHI Back Office and OHI Data Marts.

## 1.1 Subsetting and Masking

This guide describes how you can create a subset of a given OHI BO data store. The subset contains less data than the given data store but at the same time is still fully functional for the OHI BO application (that is all data integrity and consistency rules are upheld). Using data subsetting you create smaller versions of, for instance, production-size OHI BO databases. These smaller databases can then be provisioned to development teams, test teams, or deployed in user-acceptance environments. Through data subsetting you can significantly reduce the total amount of disk storage required to support the various, and often many, non-production OHI BO environments in your organization.

There is no separate subset process for an OHI DM environment, instead a OHI DM subset environment for an OHI BO subset environment needs to be created through an initial full load of the OHI BO subset.

Next to data subsetting, this guide describes how OHI BO and/or OHI DM data stores can be masked. Due to privacy regulations, organizations are obliged to deal with privacy sensitive data in a secure manner. Production environments usually have stringent data access control and auditing mechanisms in place to ensure that only those who need to access privacy sensitive data can do so. Typically those accessing the various non-production environments are not authorized to see privacy sensitive data or the data access control and auditing mechanisms are less stringent, or even absent, in these environments. With data masking you can mask (scramble, anonymize, pseudonymize) the privacy sensitive data elements in non-production OHI BO and/or OHI DM environments. Development and test teams that use these masked environments are therefore not able to see privacy sensitive data. The masked data store is still fully functional for the OHI BO and/or OHI DM application (that is all data integrity and consistency rules are upheld).

The intended audience for this technical reference guide is the DBA-group that administers the various OHI BO and OHI DM environments inside an organization. This technical reference guide contains four chapters.

1. *Introduction*
   The chapter you are currently reading. This chapter introduces you to the data subsetting and data masking packages of the OHI BO application.

2. *High Level Design*
   In chapter 2 you find a high level design of the data subsetting and data masking processes as they have been designed to operate on a data store of the OHI BO application.
3. *Detailed Process*
   In chapters 3 and 4 you find a step-by-step description of the data subsetting and data masking processes. By following these steps you can create a subset of the OHI BO application data store and mask this data store.

Data masking is implemented in the database and does not require Oracle Enterprise Manager (OEM) or additional management packs.

Data subsetting is implemented through functionality as provided through additional packs in Oracle Enterprise Manager (OEM). Data subsetting is accessed through the Quality Management submenu under the Enterprise menu of the Oracle Enterprise Manager. In this submenu you will find the two items to access the subsetting  packs:

1. Application Data Modeling
2. Data Subsetting Definitions

See Figure 1.1 for a screenshot of this submenu.

The data subsetting  pack depends upon an Application Data Model (ADM). ADMs are managed in the Application Data Modeling menu. An ADM captures all tables of the OHI BO data store involved in the subsetting  processes. For these tables the ADM defines:

- *The referential relationships between these tables*
  Subsetting processes use these to "walk through" the data model.
- *The type of table*
  Tables can either be of type transactional or config (configuration). The transactional tables are usually subsetted and the config tables are usually not subsetted.

*Figure 1.1: The Quality Management submenu in OEM13c.*

Note: The OHI BO masking process does not make use of the Data Masking Formats (the last entry in the submenu).

## 1.2 Prerequisites

The OHI BO Data Management data masking processes requires the following components:

- *OHI BO environment with the appropriate release*

The OHI BO Data Management data subsetting  processes requires the following components:

- *OHI BO environment with the appropriate release*
  This environment is usually a dedicated copy of your production environment.
- *OEM Cloud Control 13c  release 4, version 13.4.0.0.0*
- *Oracle Database Plug-in release 13.3.1.0.0*

  See section 1.3 for instructions on how to check and upgrade this component.

Note: In order to use the subsetting  pack (which is part of the Oracle Database Plug-in), you must have a license for this packs.

It is recommended to use a dedicated OEM Cloud Control instance for use with subsetting. A production monitoring instance of OEM Cloud Control might require a different release of the Database Plug-in which is not certified for use in combination with subsetting.

Note: The OHI BO Data Management data subsetting and masking processes cannot be executed on a database instance (PDB) with an active Oracle Database Vault installation. Any possible realms should be deleted and Database Vault should be disabled before executing the data subsetting and masking processes. In the Oracle Health Insurance Back Office Installation, Configuration and DBA Manual a description can be found about disabling Database Vault and any possible realms (10.3.3. Deleting the OHI realm and Disabling Database Vault in the PDB).

## 1.3 Known issues

For the 10.18.2.3.0+ release there is 1 known issue with the subsetting solution which needs additional attention.

**Database incorrectly compiles XML processing packages**

Due to a database issue, packages that contain an xmltable with columns in a cursor and which are compiled using reuse settings, with a session that runs with nls_length_semantics in byte, do use byte length semantics for these cursors. Thay may lead to ORA-06502 errors (also named ‘value errors’) when processing XML files by the OHI application when strings contain more byte than the maximum allowed character length.

Due to the way a subset environment is created this may typically occur in such an environment when no action is taken.

To prevent this it is best to recompile these packages in a preventive action. For doing this run the following pl/sql block as the OHI table owner when the subset action is finished.

```
begin
  -- run as OHI table owner, enable serveroutput when you like feedback;
  -- script might take between 30 - 90 seconds, depending on environment capacity;
  -- code can be rerun without problem if you are not sure whether it is still needed;
  dbms_output.enable(null); -- remove limit on output buffer
  for rec in
  (select 'alter '||obj.object_type||' '||obj.object_name||' compile reuse settings' as cmd
   from   dba_objects obj
   where  obj.owner = user
   and    obj.object_type = 'PACKAGE'
   and    obj.object_name like 'Z______X%PCK'
   order by
          obj.object_name
   ,      obj.object_type
  )
  loop
    dbms_output.put_line(rec.cmd);
    execute immediate rec.cmd;
  end loop;
end;
```

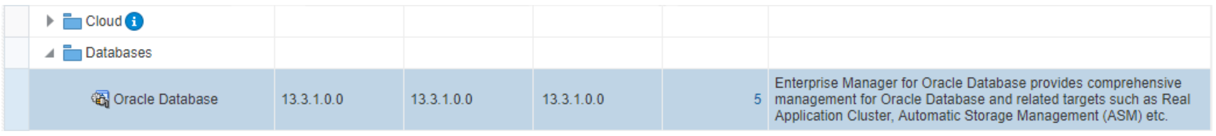
When you are not sure whether the action is necessary or already has been executed there is no harm in running it more than once.

As the cause of this problem lies in the database it is not sure yet whether a work-around will be applied to a future release of the subsetting code or that a mandatory database patch will be required.

## 1.4 Oracle Database Plug-in

You should ensure that the data subsetting pack is at the required release level for OHI BO subsetting by installing the Oracle Database Plug-in release 13.3.1.0.0.

Select Setup → Extensibility → Plug-ins and verify the version that is installed on the Management Server. See Figure 1.2.



| | | | | | |
|---|---|---|---|---|---|
| Cloud | | | | | |
| Databases | | | | | |
| Oracle Database | 13.3.1.0.0 | 13.3.1.0.0 | 13.3.1.0.0 | 5 | Enterprise Manager for Oracle Database provides comprehensive management for Oracle Database and related targets such as Real Application Cluster, Automatic Storage Management (ASM) etc. |

*Figure 1.2: Verifying current version of database plug-in.*

If the required version is not yet installed and not visible in the Latest Available column, you can use the Self Update functionality of OEM13c to download the new version. See Figure 1.3.

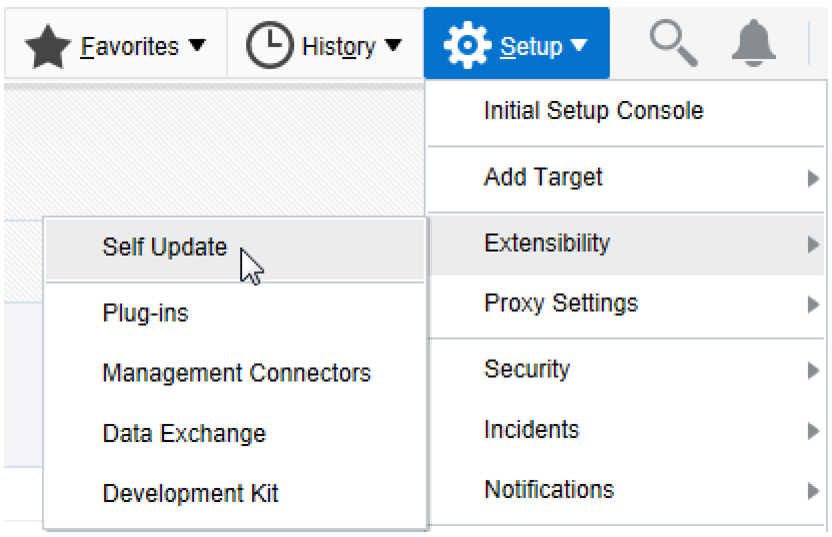


*Figure 1.3: OEM13c Self Update.*

Please note that during upgrade of the Oracle Database Plug-in on the Management Server you are required to schedule downtime of the OEM13c instance.

# 2 High Level Design

This chapter contains a high level overview of the data subsetting and data masking processes as they apply to the OHI BO application.

## 2.1 Subsetting Process

Subsetting in general can be performed in two distinct modes:

1. *All data that is not part of the subset is deleted from an existing data store.*
   In this case database reorganization is required after running the subset process to reclaim the empty free space in the data files.
2. *All data that is part of the subset is exported from an existing data store.*
   The export file is imported into an empty (smaller) database after running the subset process.

The OHI BO subsetting process as supported and implemented through the OHI Data Management product implements the second mode.

Before starting with the subset process, you first have to import two XML-files into OEM:

- The ADM xml file holding a description of the OHI BO tables to be subset. This file is named *SDM_OHI_[release] _SUBSET_ADM.xml.* This file should be generated through a utility that is part of the OHI Data Management product and delivered in an OHI Back Office release.
- The subsetting xml file holding the specification of the subset process. This file is named *SDM_OHI_[release](_CONFIG)_SUBSET_DEF.xml (where "_CONFIG" is only used if a configuration only subset is created)*. This file is another main component of the OHI Data Management product and also needs to be gerenated by a utility that is delivered in an OHI Back Office release.

Through the subsetting pages in OEM you can start the export job. This export job creates a dump file that contains the relevant subset of the source OHI BO data store. You then have to prepare an empty target OHI BO data store in a smaller database. Finally run an import job to load the subset into this smaller database. Figure 2.1 shows a high level overview of this process.

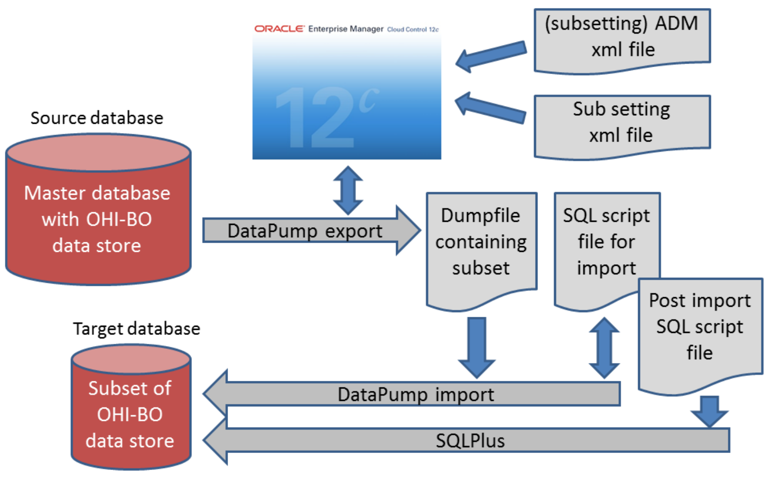

ORACLE Enterprise Manager Cloud Control 12c
12c
(subsetting) ADM xml file
Sub setting xml file
Source database
Master database with OHI-BO data store
DataPump export
Dumpfile containing subset
SQL script file for import
Post import SQL script file
Target database
Subset of OHI-BO data store
DataPump import
SQLPlus

*Figure 2.1: High level overview of subsetting process.*

Next to the dump file, the subsetting export job also generates a SQL script file for import and a post import SQL script file (as shown in Figure 4). This import SQL script file contains the commands that import the dump file into the empty subset database. This script file can be run using SQL*Plus. The post import SQL script file needs to be run after the data pump import has completed.

Note: The source database needs to be a quiescent database, as the export job runs for some time and the dump file needs to be a read-consistent snapshot of the source database. The source database is usually a dedicated RMAN copy (or Dataguard copy opened in snapshot standby mode) from your production environment.

The target database needs to be correctly prepared before starting the import job. This is described in Chapter 3.

## 2.2 Data Masking Process

Data masking is performed in-place. The masking process is run on a target database, which is then masked. Masking is performed by either executing a stored procedure or scheduling a database job in the target environment.

Figure 2.2 shows a high level overview of this process.

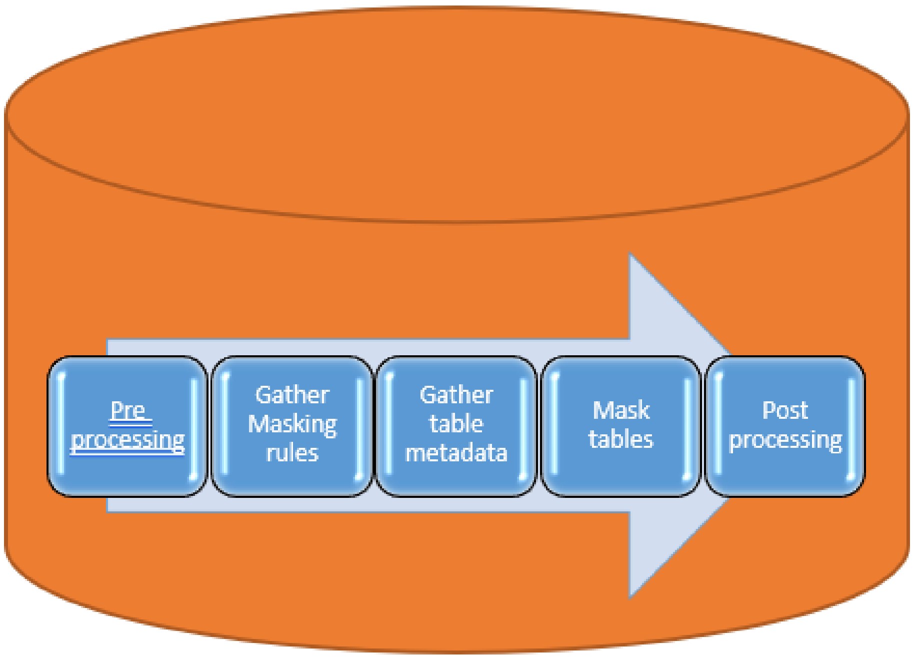


*Figure 2.2: High level overview of masking process.*

For the set of tables, based on the masking rules, the metadata is gathered and stored in a processing table. The metadata contains the definition of the table and its indexes, constraints and triggers.
During the masking process this data is processed per table.
If a tablespace name is provided the original table is first moved to this tablespace so no additional space is used in the original tablespace. The specified tablespace can be created temporarily and may be removed afterwards.

The next step is to rename the original table (either in the original tablespace or in the specified tablespace) and create a new table, in the original tablespace, based on the stored metadata of the original table (that has bee nrenamed). This newly created table is than filled with the data from the original table with the masking rules applied to it. Next the constraints and indexes are added to the table and if all steps are successful the original (renamed) table is dropped and the next table will be processed.

Chapter 4 contains a detailed description of the data masking process.

# 3 Detailed Process - Subsetting

As mentioned in Chapter 2.1 you need to acquire the two XML files that describe how the OHI BO data store is to be subset, before starting the subsetting process.

- Subsetting application data model (*SDM_OHI_[release]_SUBSET_ADM.xml*)
- Subsetting definitions (*SDM_OHI_[release](_CONFIG)_SUBSET_DEF.xml*)

As part of the subsetting process you need to generate the ADM xml, and import both this generated ADM and the subsetting definition into OEM.

## 3.1 Provision Source Database to be Subset

The subsetting process starts by designating a source database as the database from which the subset is to be created. Oracle strongly advises that you do not use a live production database for the following reasons:

- The subsetting process places considerable load on the source database.
- The data pump export sub process that performs the subset extraction, needs a single read consistent snapshot of the source database. Therefore, if you use a non-quiescent database as the source database, the data pump export produces more load as rollback segments are heavily used to recreate a read-consistent snapshot.
- Subsetting can run in a *delete* mode as mentioned in Section 2.1. You run the risk of accidentally choosing the *delete* mode instead of the *data pump export* mode as described in step 6 of the subsetting process. In which case you would submit a job that starts deleting data from your live production database.

You would usually use a dedicated RMAN backup copy of the production database as your subsetting source database. Another alternative could be to use a Data Guard (snapshot) environment of your production database.

**Note**: As part of the subsetting process, you have to load a few test data management packages into the source database. This implies that the Data Guard environment has to be converted into a Snapshot Standby database first. When the subsetting process has completed, you can convert the Data Guard environment back into a (Physical) Standby database. See the "Data Guard Concepts and Administration" guide for more information.

**Separate database user and 'temporary' tablespaces for subsetting**

We recommend to create a separate database user and separate tablespace(s) (which can be deleted afterwards and is 'temporary' in that sense) for running the subset export process with. The subset export creates temporary working tables during the subsetting  job. These tables are created in the *default tablespace* of the user credentials you supply when submitting the subset export job.

The default tablespace might need up to tens of GB's for an OHI Back Office database of a few terabytes. The temporary tablespace may grow much more; sizing advise is given later on in this document.

The user can be created as follows:

```
create user <USER> identified by <PASSWORD> default tablespace <SCRATCH TABLESPACE>
temporary tablespace <TEMP TABLESPACE THAT MAY GROW; SEE LATER>;
grant dba to <USER>;
grant execute on DBMS_AQADM to <user>;
```

By employing a separate database user and separate tablespace for its objects and potentially also a separate TEMP tablespace, you can easily reclaim the disk space occupied by the working tables and temporary segments by dropping these tablespaces afterwards.

For performance reasons it is best to de-activate all maintenance background jobs to prevent for example a parallel statistics gathering job uses unnecessary resources that delay the subset process.

Note: When Virtual Private Database (VPD) is activated in the OHI Back Office scheme this database user needs to be exempt from the VPD policies to be able to collect all required data.
This can be done as follows:

```
grant exempt access policy to <user>
```

## 3.2 Define Policy or Claim line Selection in the Source Database

OHI BO subsetting can be used to create two kinds of subsets:

- A 'configuration only' subset. This is a 'subset' copy where all policy, claim, person and financial data is not copied but only the OHI configuration data is copied (fully). One can also refer to this as a 0% transactional subset and 100% configuration subset.
- A 'regular' subset where a selection of the 'transactional data', policies and/or claims related data, is copied from the source environment next to the 100% configuration subset.

In a 'configuration only' subset no policy or claim line selection in the source database is required. If it is required to also include transactional data in the subset, a policy or claim line selection within OHI Back Office can be used to determine which rows should be included in the subset.

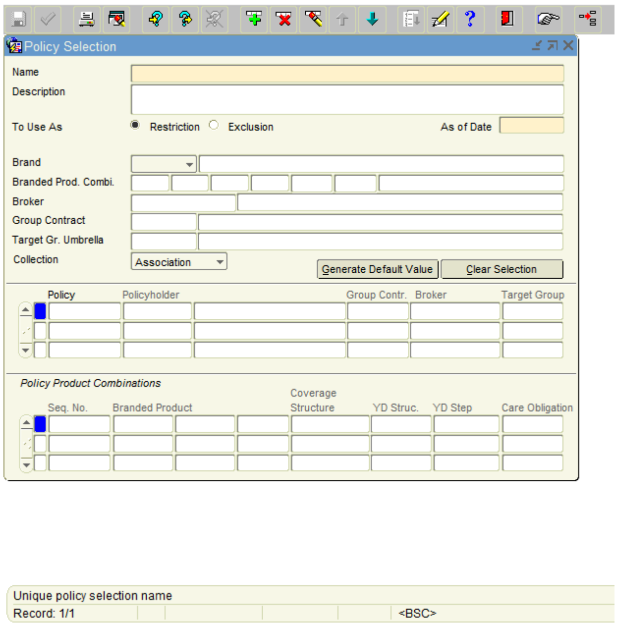
In this section the setup of the policy selection is explained in more detail, but a claim line selection can be used as well and works in a similar way.

A policy selection is a set of policy numbers. Typically you should determine a representative set of policies to be included in your target subset environment as this forms the base of the data that will be transferred to the subset database. All policy related data, including relevant claims, will be incorporated in the subset. So determining a well thought over policy selection is an important aspect in setting up your subset environment.

So you have to create a policy selection that includes all relevant policy numbers that need to be in the subset. There are two ways you can do this:

1. Use the OHI Back Office Policy Selection window to setup a policy selection.
2. Use SQL*Plus to directly populate the underlying two tables of a policy selection.

Figure 3.1 shows the Policy Selection window. You have to enter a name for the policy selection (this name will be input for one of the steps later on). The description is optional. All other items can be left empty or at their default value. In the second block you enter all required policy numbers.



To Use As: [✓ Restriction] Exclusion

Collection: Association

Unique policy selection name

Record: 1/1 <BSC>

*Figure 3.1: The Policy Selection window.*

The two underlying tables for the Policy Selection screen are depicted in Figure 3.2. They are:

- **VER#POLICY_SELECTIONS_** (Dutch name **VER_POLIS_SELECTIES**), and
- **VER#POL_PER_POS_** (Dutch name **VER_POL_PER_POS**).

It might be more convenient to use a SQL tool, such as SQL*Plus, to create the necessary rows manually in these tables.

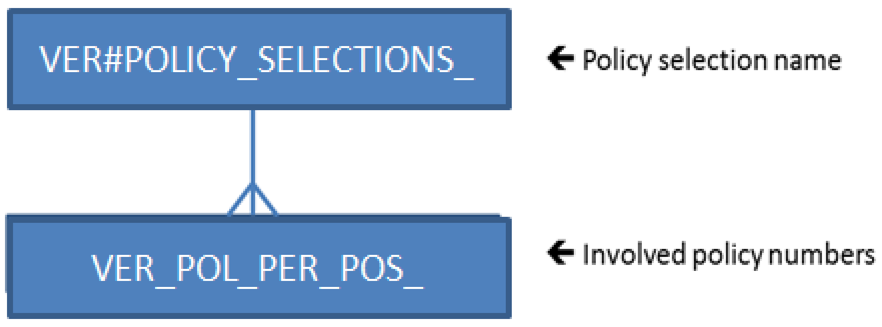


*Figure 3.2: Policy selection tables.*

For example, the following two insert statements set up a policy selection named SAMPLE_OF_5_PERCENT that holds a random sample of 5% of the total number of policies available in the source database.

```
insert into ver#policy_selections_ (name) values('SAMPLE_OF_5_PERCENT');

insert into ver#pol_per_pos_ ( pos_id, pol_num )

select ( select id from ver#policy_selections_ where name = 'SAMPLE_OF_5_PERCENT')

       , num

from ver#policies_ SAMPLE(5);


commit;
```

For running these statements you need an account with (temporary) insert privileges on these tables unless you use the OHI BO table owner account, which is strongly discouraged given the security impact.

Note: The subsetting process uses a policy selection named OHI_SDM_POS_SUBSET internally. This name is reserved by the OHI BO subsetting development team. You cannot use this name as it would interfere with the subsetting process.

## 3.3 Generate and import Subsetting Application Data Model into OEM

The ADM contains the data model for the OHI BO application. This model is used by the subsetting process to calculate the subset. The ADM is specific to an OHI BO release and must be generated for each new release. Prior to importing you should make sure that the ADM to be imported corresponds with the OHI BO release installed for the source database.

To generate the ADM for subsetting, connect with SQL*Plus to the source database. Invoke the following command (as OHI Back Office table owner):

```
exec sdm_adm_drv_pck.write_adm_files('DB_DIR');
```

Replace DB_DIR with the database directory of your choice.

Two files will be written to this directory:

- *SDM_OHI_[release]_MASK_ADM.xml*
- *SDM_OHI_[release]_SUBSET_ADM.xml*

Once these files have been generated, you may want to transfer these to a file system that is accessible from your Desktop, so you can import them using Enterprise Manager.
Open the *Application Data Models* page. See Figure 3.5.

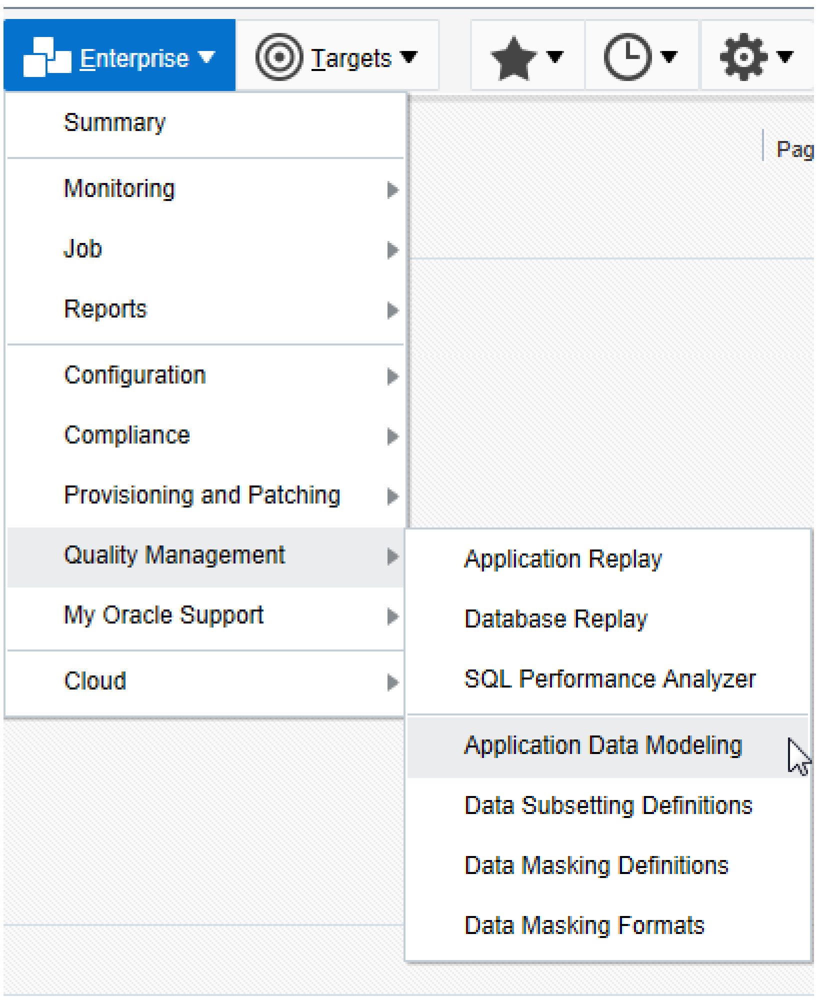


*Figure 3.5: Open Application Data Models*

Make sure the ADM XML file is on your local desktop. Then select Actions → Import → File from Desktop. See Figure 3.6.

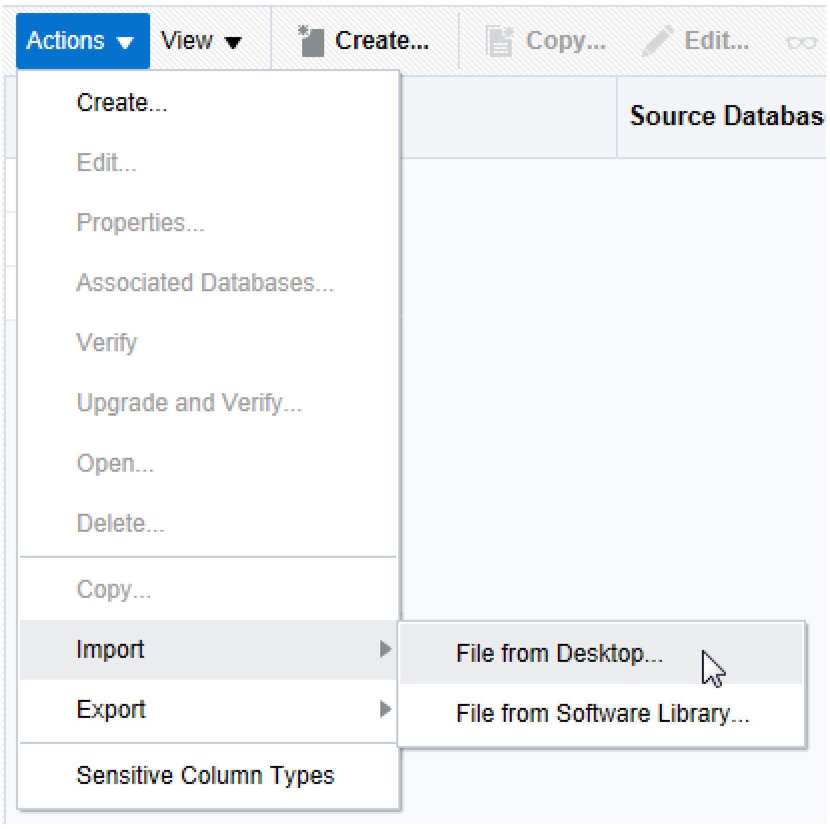
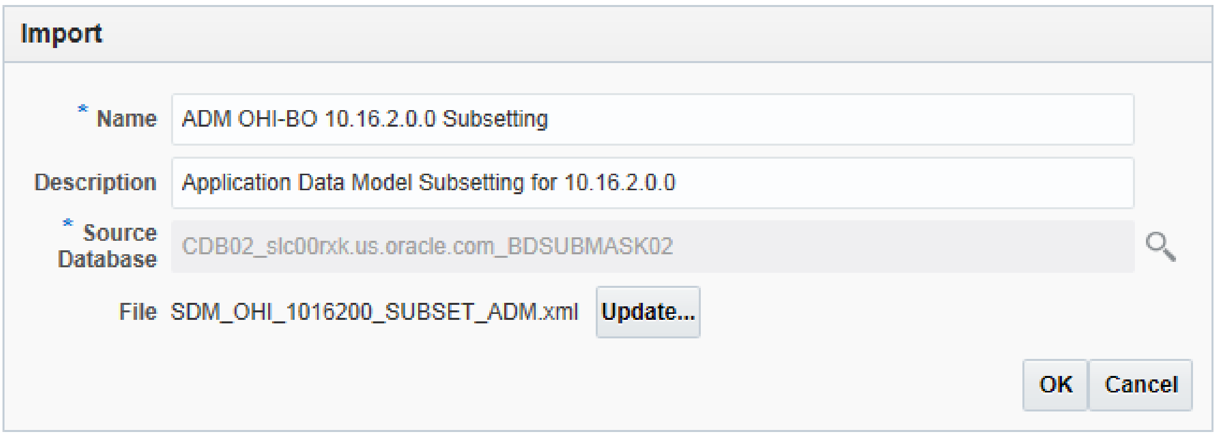


*Figure 3.6: Starting the File from Desktop import.*

Next enter a name for this ADM. Specifying the release number as part of the name of the ADM will help identify the correct ADM later on. Enter a description and select the subsetting source database. Then press Choose File and navigate to the ADM XML file on your local desktop. Finally press the OK button to start the import. See Figure 3.7.

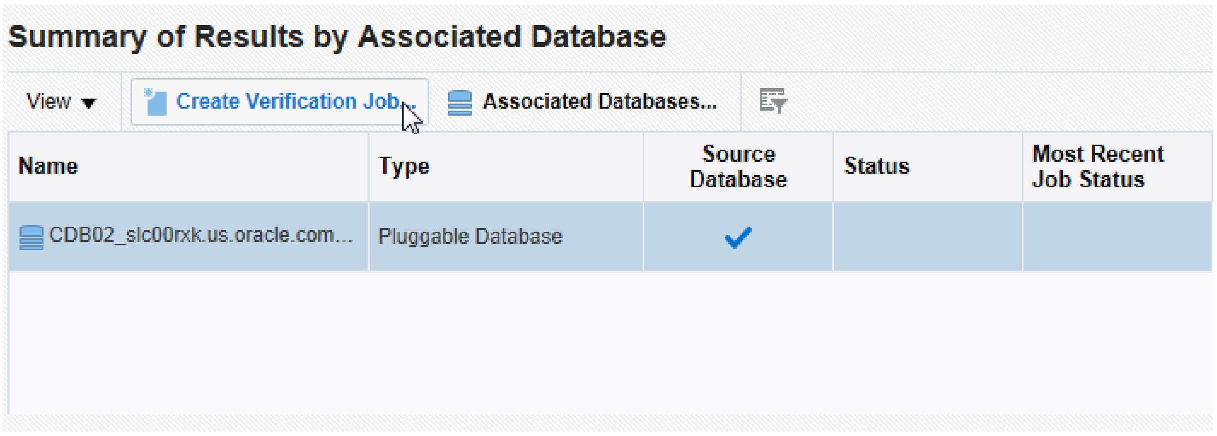

Import

* Name: ADM OHI-BO 10.16.2.0.0 Subsetting
Description: Application Data Model Subsetting for 10.16.2.0.0
* Source Database: CDB02_slc00rxk.us.oracle.com_BDSUBMASK02
File: SDM_OHI_1016200_SUBSET_ADM.xml

*Figure 3.7: Specifying Subsetting ADM XML file to be imported.*

The ADM import process may take a while.

After the import process has completed, the ADM must be verified against the source database. Select the imported ADM in the list of available ADMs. Then select Actions → Verify, and click Create Verification Job. See Figure 3.8.

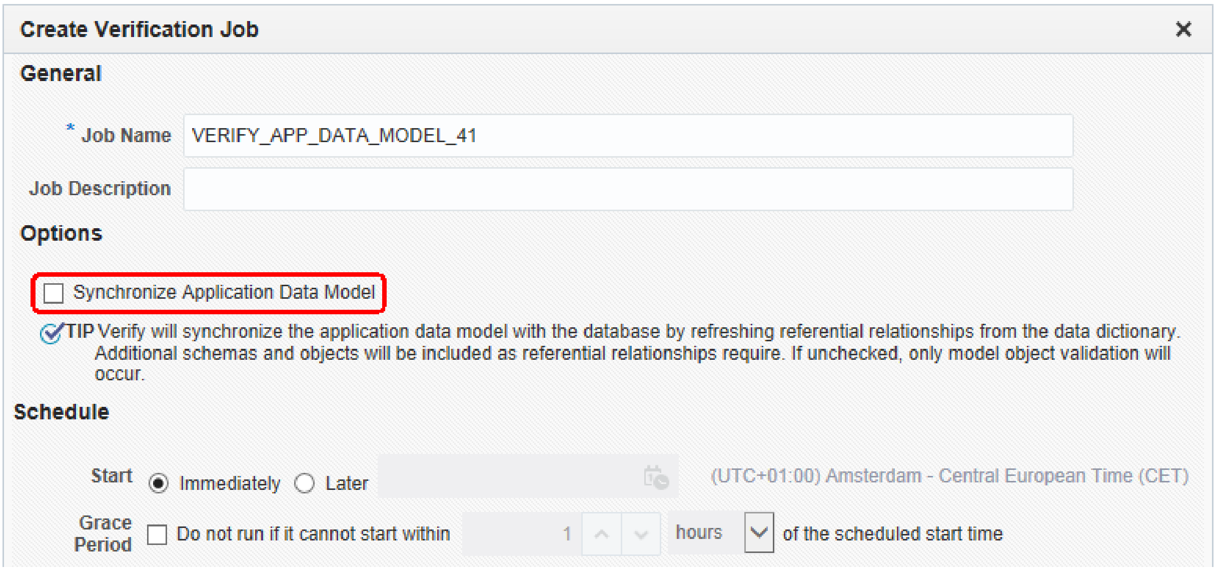


Summary of Results by Associated Database

| Name | Type | Source Database | Status | Most Recent Job Status |
|---|---|---|---|---|
| CDB02_slc00rxk.us.oracle.com... | Pluggable Database | ✓ | | |

*Figure 3.8: Create Verification Job.*

Make sure to **uncheck** Synchronize Application Data Model before submitting, see Figure 3.9.



Create Verification Job

General

* Job Name: VERIFY_APP_DATA_MODEL_41

Job Description:

Options

☐ Synchronize Application Data Model

TIP Verify will synchronize the application data model with the database by refreshing referential relationships from the data dictionary. Additional schemas and objects will be included as referential relationships require. If unchecked, only model object validation will occur.

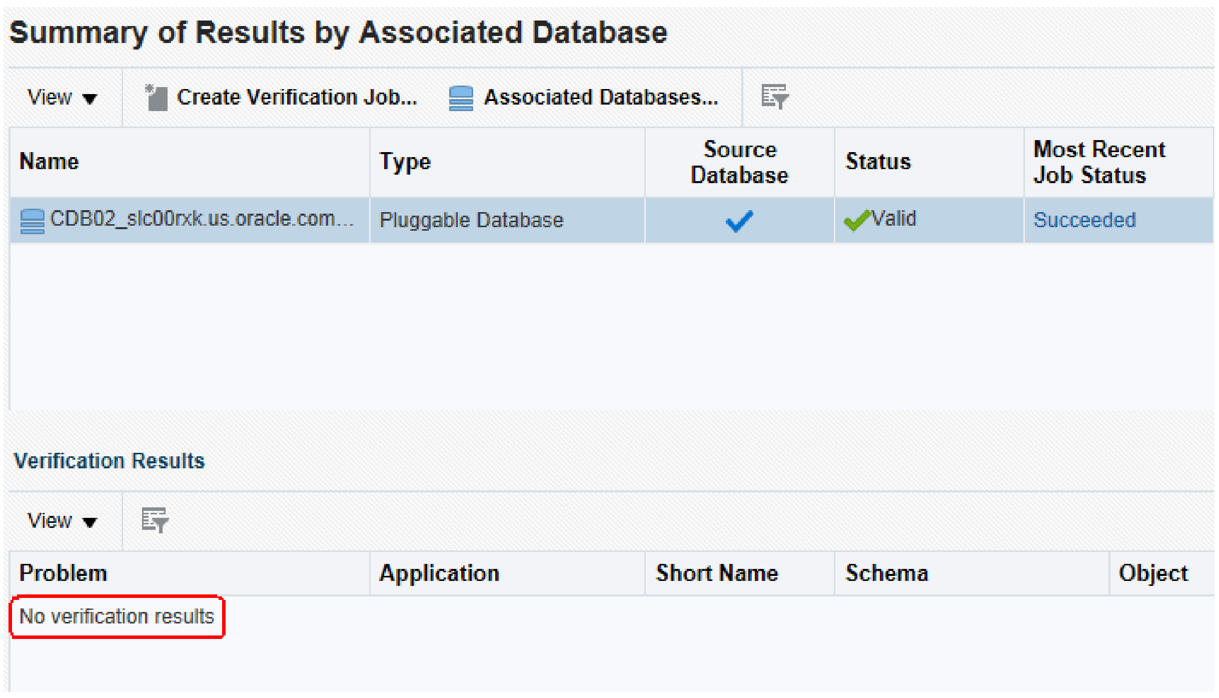
Schedule

Start [✓ Immediately] Later (UTC+01:00) Amsterdam - Central European Time (CET)

Grace Period ☐ Do not run if it cannot start within 1 hours of the scheduled start time

*Figure 3.9: Uncheck Synchronize Application Data Model.*

When the verification job has completed, you need to check the verification results. For this select the ADM from the list of available ADMs, and select Action → Verify. Now select the row with the database that you associated with the verification job. Look at the Verification Results. It should say: No verification results. See Figure 3.10.


**Summary of Results by Associated Database**

| Name | Type | Source Database | Status | Most Recent Job Status |
|---|---|---|---|---|
| CDB02_slc00rxk.us.oracle.com... | Pluggable Database | ✓ | ✓Valid | Succeeded |

**Verification Results**

| Problem | Application | Short Name | Schema | Object |
|---|---|---|---|---|
| No verification results | | | | |

*Figure 3.10: Checking verification results.*

## 3.4 Generate and Import Subsetting Definitions into OEM

The subsetting definitions XML file contains all the rules to create a subset of OHI Back Office.
To generate the definition file for subsetting, connect with SQL*Plus to the source database. Invoke the following command (as OHI Back Office table owner):

```
exec sdm_subset_drv_pck.write_file
     ( pi_directory   => 'DB_DIR'
     , pi_config_only => 'Y or N'
     , pi_including   => 'Y or N'
     );
```

Replace DB_DIR with the database directory of your choice.
For pi_config_only (default value is ‘N’), you can enter:
- Y: when a configuration only subset is required (no policies and claims)
- N: when a subset is required with transactional data and configuration data

pi_including (default value is ‘Y’) only works when pi_config_only is ‘Y’, you can enter:
- Y: brokers and group contracts are included in the configuration only subset
- N: brokers and group contracts are not included in the configuration only subset

Once this file has been generated, you may want to transfer this to a file system that is accessible from your Desktop, so you can import them using Enterprise Manager.

The subsetting definitions XML file is imported from the Data Subset Definitions page. Navigate to this page by selecting Quality Management->Database Subset Definitions as shown in Figure 3.11. Make sure you have the XML file loaded on your desktop.

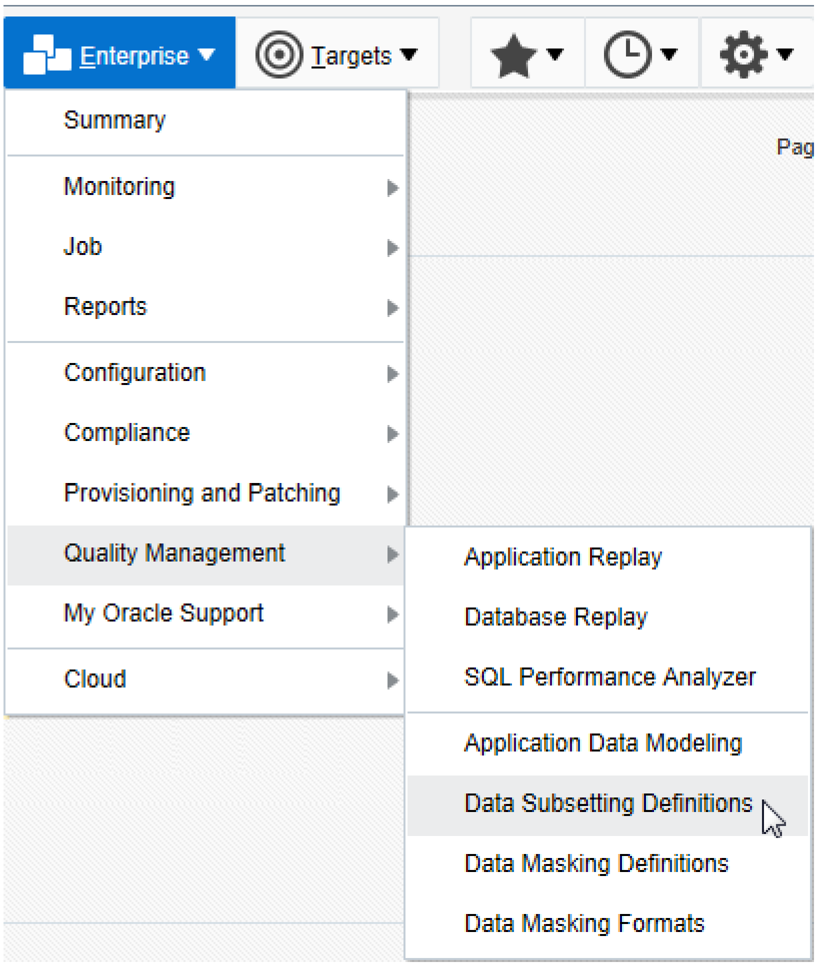


*Figure 3.11: Open Data Subset Definitions*

Select Actions → Import → File from Desktop in the Data Subset Definitions page. See Figure 3.12.

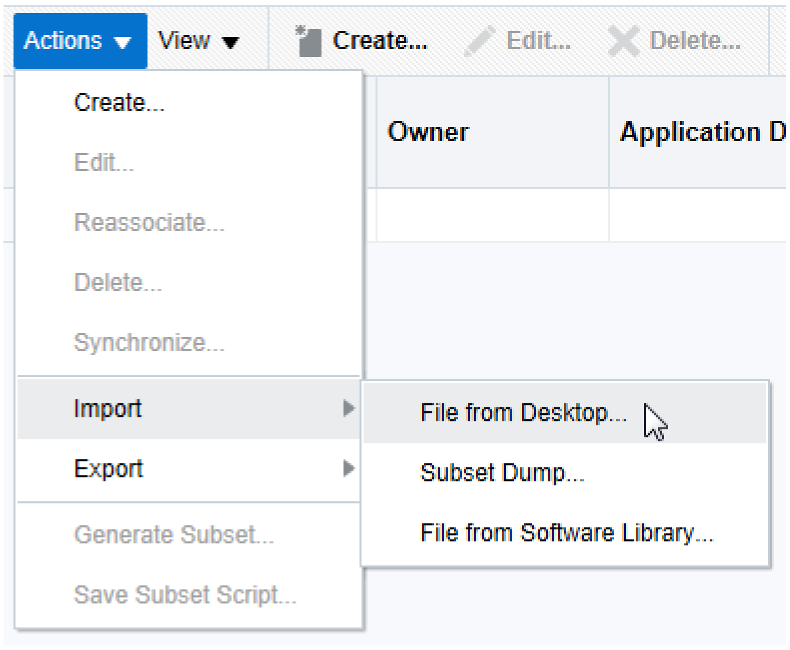
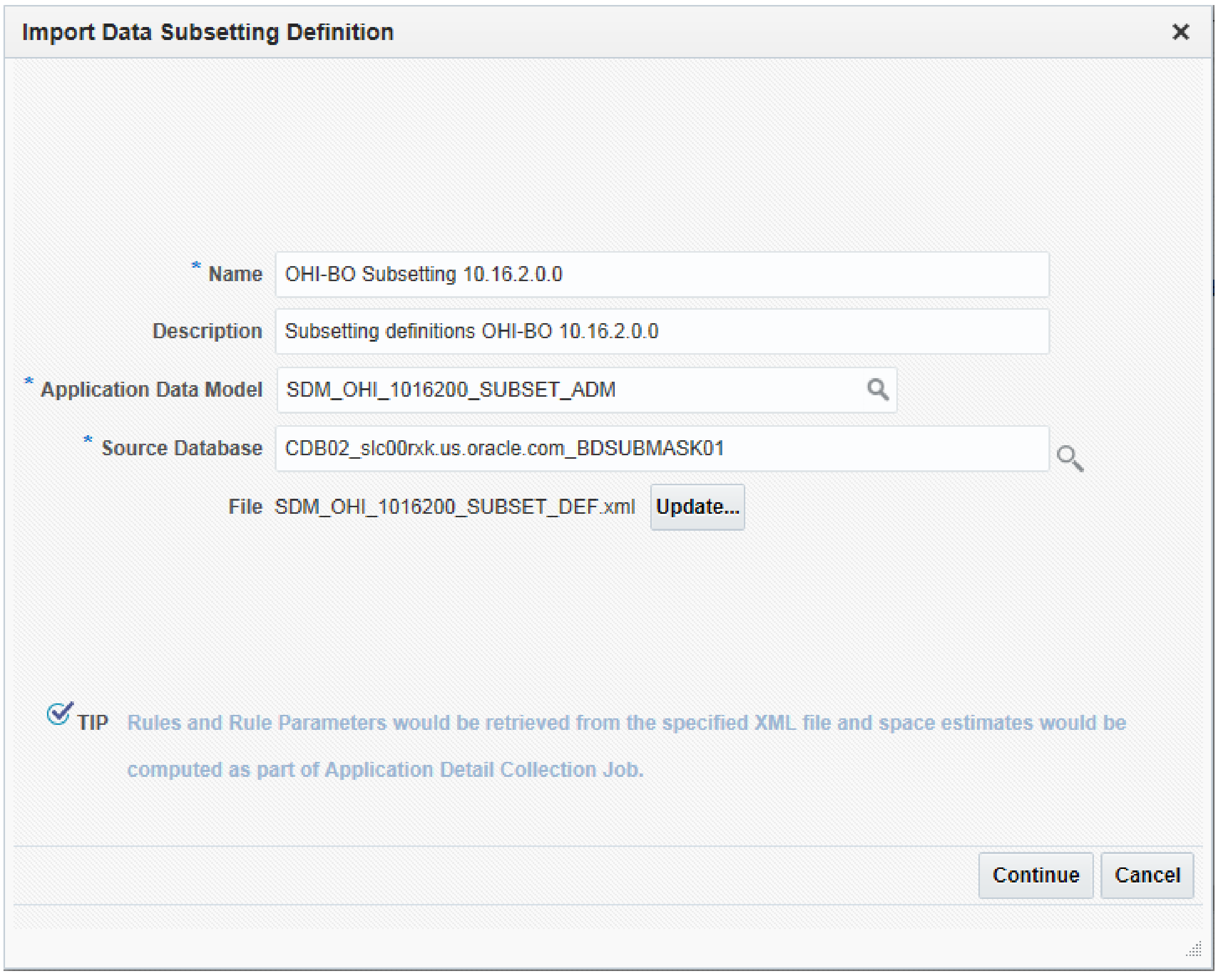


*Figure 3.12: Starting the File from Desktop import.*

In the Import Data Subset Definition page enter a name (preferably include release name here too) and optionally a description. Select the **corresponding ADM** and select the source database. Finally select the subsetting XML file from your local desktop and press the Continue button. See Figure 3.13.

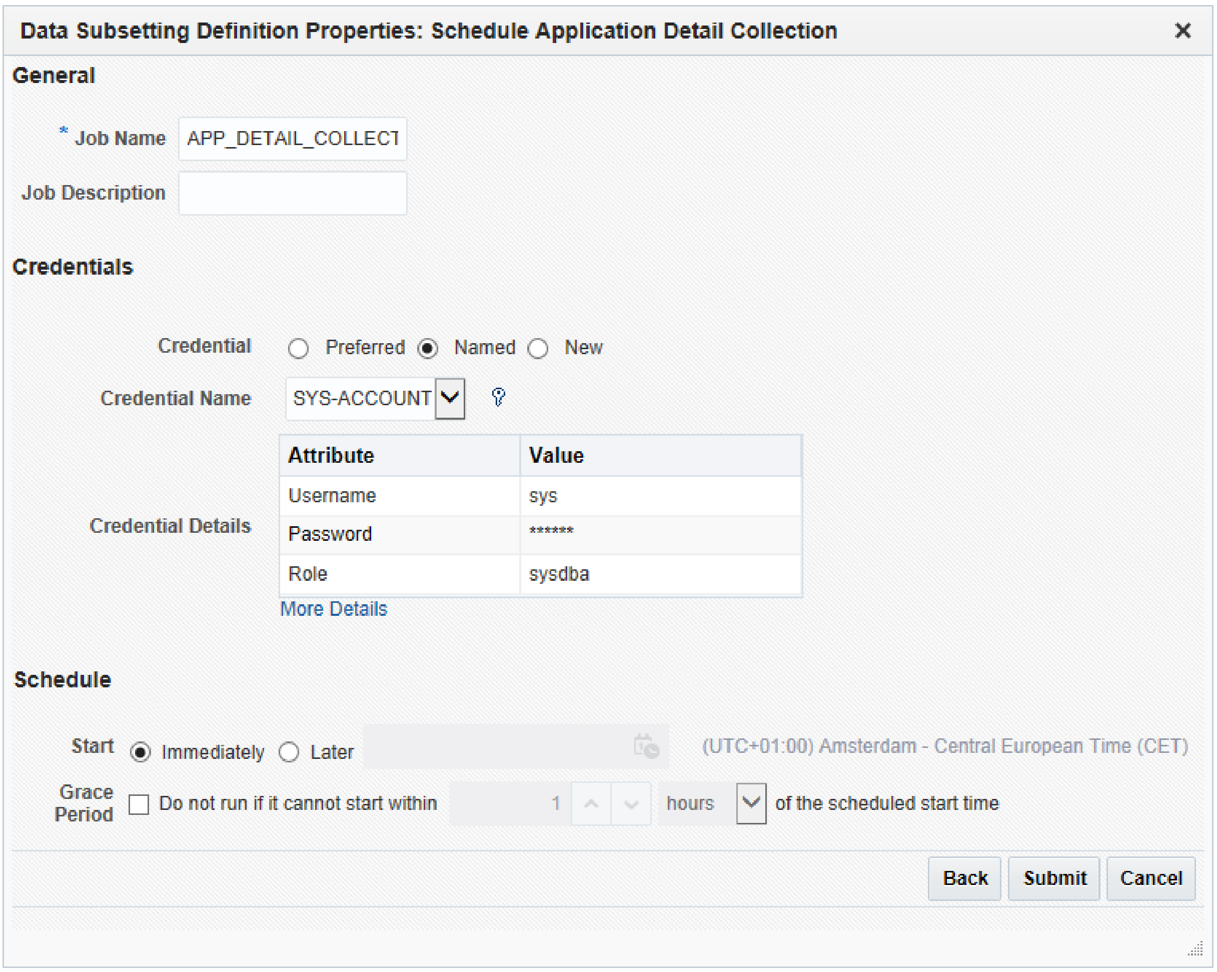

Import Data Subsetting Definition

| | |
|---|---|
| * Name | OHI-BO Subsetting 10.16.2.0.0 |
| Description | Subsetting definitions OHI-BO 10.16.2.0.0 |
| * Application Data Model | SDM_OHI_1016200_SUBSET_ADM |
| * Source Database | CDB02_slc00rxk.us.oracle.com_BDSUBMASK01 |
| File | SDM_OHI_1016200_SUBSET_DEF.xml |

TIP Rules and Rule Parameters would be retrieved from the specified XML file and space estimates would be computed as part of Application Detail Collection Job.

*Figure 3.13: Import Data Subset Definition.*

Specify the subset user credentials of the source database in the next page and press the Submit button. See Figure 3.14.


Data Subsetting Definition Properties: Schedule Application Detail Collection

General

* Job Name: APP_DETAIL_COLLECT

Job Description:

Credentials

Credential: ( ) Preferred [✓ Named] ( ) New

Credential Name: SYS-ACCOUNT

Credential Details:

| Attribute | Value |
|---|---|
| Username | sys |
| Password | ****** |
| Role | sysdba |

More Details

Schedule

Start: [✓ Immediately] ( ) Later — (UTC+01:00) Amsterdam - Central European Time (CET)

Grace Period: ☐ Do not run if it cannot start within 1 hours of the scheduled start time

*Figure 3.14: Specifying credentials for import job.*

A job is created to import the subsetting definitions from the XML file. Verify the job succeeded. This job should only run for a couple of minutes.

**Note**: When there is a need to import the subset definition for a second time after a subset export has been run, purging the internal SDM selection tables will speed up the import job. During the import, an estimation on the size of the subset is performed. When these internal tables are filled, this estimation will take a considerable amount of time.

The internal SDM tables can be purged with the following command (as OHI Back Office table owner):

```
exec sdm_util_pck.purge_sdm_data;
```

## 3.5 Generate Subset Export File

**Recommended database configuration during export**

Before starting the actual export job, the source database should be configured to maximize performance. The following parameters are recommended (and deviate from parameters required for OHI Back Office runtime):

- `parallel_degree_limit` = IO
  To limit the number of parallel query workers to the optimum found during **IO Calibration**. This calibration must be performed before setting this parameter to 'IO' (see below).
- `parallel_max_servers` - default value (>1), to enable parallel query during subset export
- `parallel_degree_level` - default value (100), as reducing the Degree Of Parallellism as wished for OHI Back Office during regular use is not wanted for this export job
- `optimizer_mode` – default value (ALL_ROWS)
- `optimizer_index_cost_adjust` – default value (100)

With respect to memory allocation, make sure the SGA (in particular the BUFFER_CACHE value) and PGA (for in-memory sorting and hash-area) are generously sized. As an indication: for a high volume environment (5+ TB database size) an `sga_target` of 16GB and a `pga_aggregate_target` of 12GB are found to be suitable values. These values of course depend on several system properties and the optimal values should be determined when running the subset export. The Memory Advisor included in Oracle Enterprise Manager can be a guide to find the optimum.

It is also advisable to disable the archivelog mode, when active, on the source database prior to starting the export job.

**IO Calibration**

To effectuate the 'parallel_degree_limit=IO' setting, IO Calibration needs to be performed. This needs to be done at the CDB$ROOT container level and not in the PDB.

There are a number of ways to perform this calibration. One way is to start the IO Calibration from OEM: navigate to the 'Performance Home' of the database, click on the "I/O" tab and click the "I/O Calibration button" (see Figure 3.15). After specifying the number of disks and the maximum tolerable latency (10ms minimum), the calibration is started. The calibration process will take about 15 minutes.

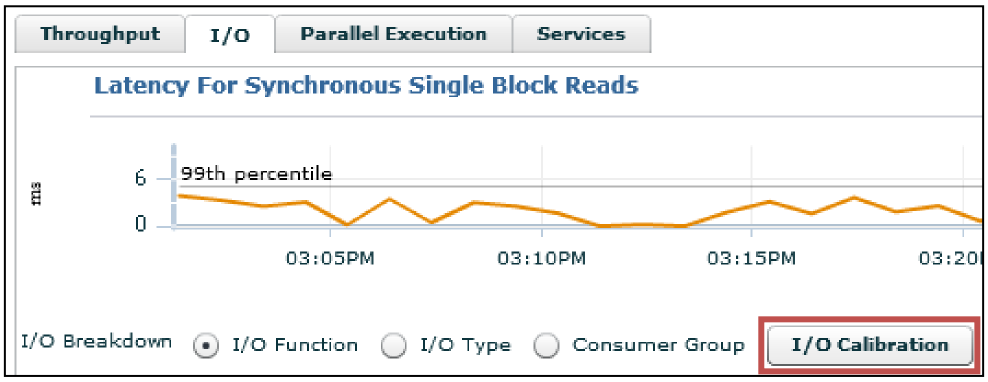


*Figure 3.15: Start IO Calibration from OEM*

Another possibility is to start calibration from SQL*Plus. See Running I/O Calibration for more information.

In MOS note **"Automatic Degree of Parallelism in 11.2.0.2 (Doc ID 1269321.1)"** a method is described to set the IO calibration data manually, without the need to run the IO Calibration. This could be useful when the used production-copy runs in a CDB$ROOT container on hardware with different characteristics where you do not want or cannot execute IO calibration for. Whether the described method can still be used in a 19c database is unclear.

**Temporary tablespace requirements**

Make sure the temporary tablespace of the subset user (so the tablespace that is specified as temporary tablespace during the create user command and not its 'scratch' tablespace) is set to auto extend, and enough disk space is available for it to grow. During the export, tables can be written to temp space first before being written to the export file. This can consume large amount of temp space. As a guideline: make sure the temporary tablespace can hold at least the "subset %" of the size of the largest application table. For example, 0.05 times the largest application table when a 5% subset is created.
When using multiple worker threads during export, temporary table space can be claimed by each worker. Multiply the subsetted size of the largest table by the number of workers.

$$TEMP\ size\ \approx (\#workers) * (size\ of\ largest\ application\ table) \\ * (estimated\ subset\ fraction)$$

For example, when using 2 worker threads for a 5% subset and the largest table being 1000GB: expect 2 * 1000 * 0.05 = 100GB of required temporary tablespace.

**Submitting the export job**

The generation of the subset export file can now be started. From the Data Subset Definitions page select the subset definition to use for generating the subset and then select Actions → Generate Subset. See figure 3.17.

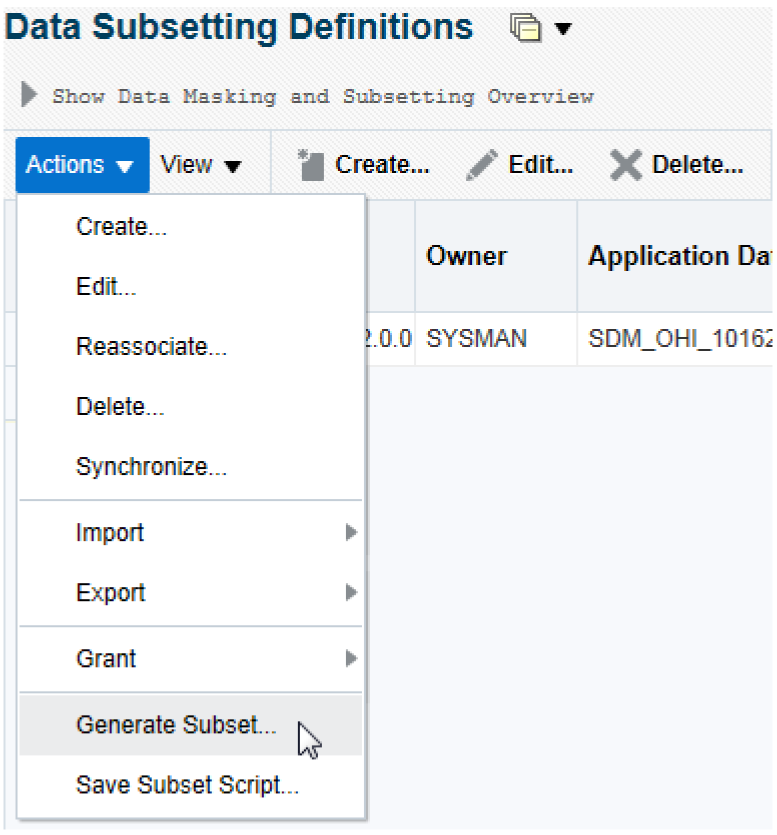
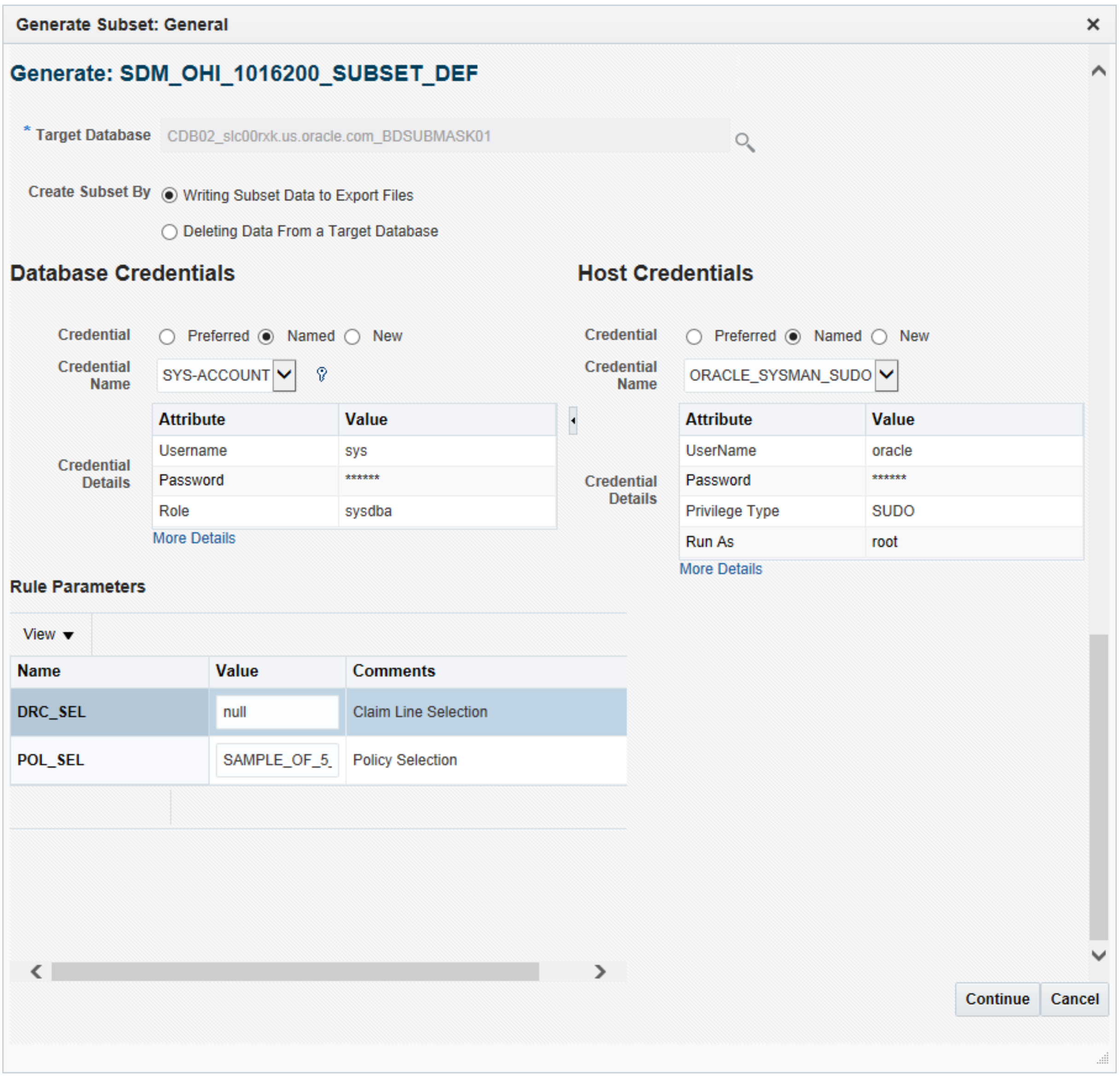


*Figure 3.17: Generate Subset.*

Specify the database from which the subset is to be exported (this is called the target database here). Keep the ‘Create Subset By’ radio button default (Writing Subset Data to Export Files). Select the database and host credentials and if applicable, specify the name of the policy or claim line selection as created in step 2 (Section 3.1.2) to drive the subset generation (for a configuration only subset, enter “null” for both selections). Then press the Continue button. See Figure 3.18.



**Generate: SDM_OHI_1016200_SUBSET_DEF**

* Target Database: CDB02_slc00rxk.us.oracle.com_BDSUBMASK01

Create Subset By: [✓ Writing Subset Data to Export Files] Deleting Data From a Target Database

**Database Credentials**

Credential: Preferred [✓ Named] New

Credential Name: SYS-ACCOUNT

Credential Details:

| Attribute | Value |
|---|---|
| Username | sys |
| Password | ****** |
| Role | sysdba |

More Details

**Host Credentials**

Credential: Preferred [✓ Named] New

Credential Name: ORACLE_SYSMAN_SUDO

Credential Details:

| Attribute | Value |
|---|---|
| UserName | oracle |
| Password | ****** |
| Privilege Type | SUDO |
| Run As | root |

More Details

**Rule Parameters**

View ▾

| Name | Value | Comments |
|---|---|---|
| DRC_SEL | null | Claim Line Selection |
| POL_SEL | SAMPLE_OF_5_ | Policy Selection |

*Figure 3.18: Specify general parameters for subset process.*

**Note**: If Enterprise Manager freezes after pressing the continue button this could be due to the internal SDM selection tables. When there is a need to import the subset definition for a second time after an earlier subset export has already been run, purging the internal SDM selection tables will solve this problem. During the import, an estimation on the size of the subset is performed. When these internal tables are filled, this estimation will take a considerable amount of time.

The internal SDM tables can be purged with the following command (as OHI Back Office table owner):

```
exec sdm_util_pck.purge_sdm_data;
```

You enter the data pump parameters in the next page and specify the directory object that data pump must use to create the dump file in. This must be an existing directory object (one that the DBA must have created on beforehand, using the CREATE DIRECTORY command) inside the source database and the OHI BO application owner schema, usually OZG_OWNER, must have read/write access on this directory object.

Provide the export file name, maximum file size and the number of worker threads that data pump must use whilst querying the source database and generating the export files.

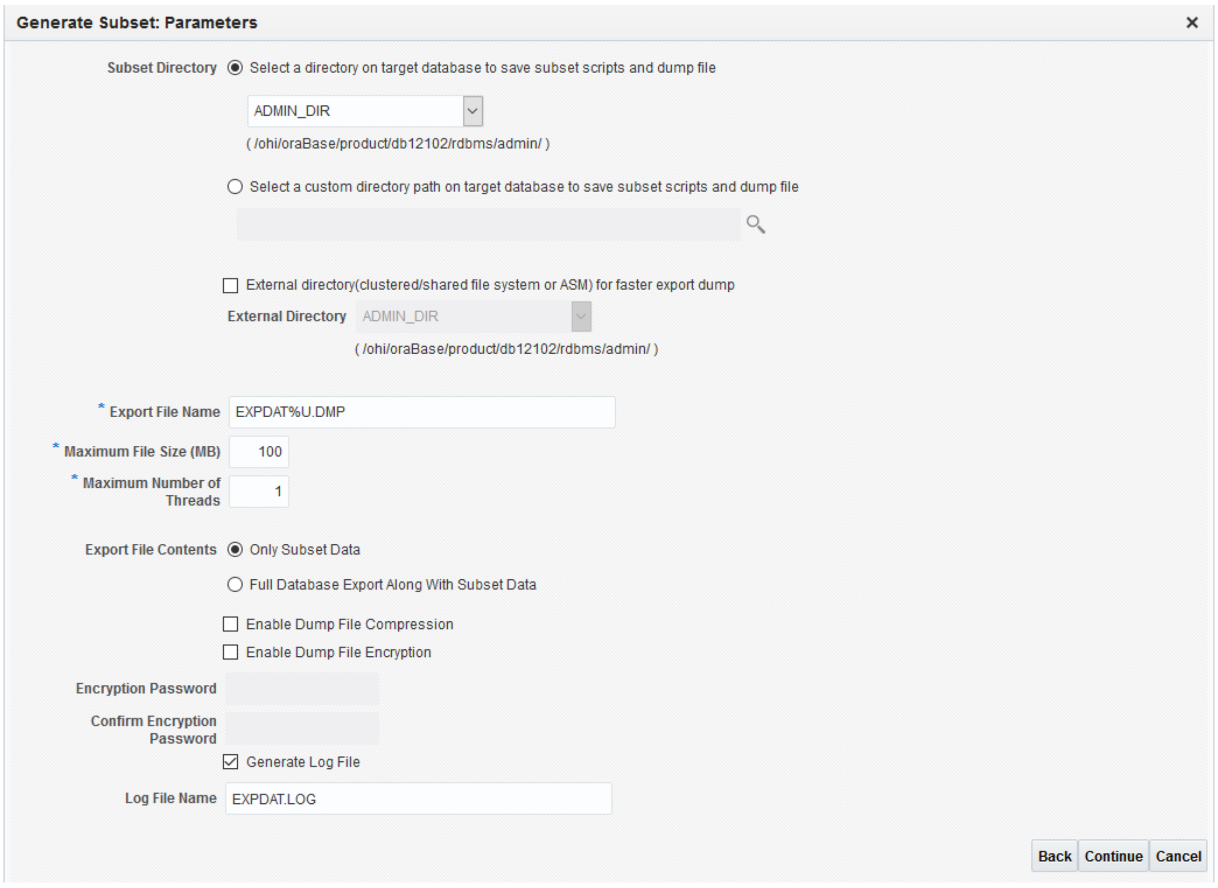
The maximum file size should not be set too low, because the total export size is limited by the nr of files, which is 100 files maximum to be used and those files should together be large enough to contain the subset export dump. This means the exported database dump should fit into 100 * {maximum file size}. Hence the default file size of 100M accommodates for a 10G maximum export size.

Consider setting the number of worker threads to 2. Depending on the number of available CPU's and IO capacity it will decrease the time needed for the export. When using two workers the table data is exported in parallel.

Make a choice between the subset only or full database including subset export. The first will only replace the OZG_OWNER scheme and expects the rest of the schemes to be present, e.g. the custom development schemes, the (webservice) user schemes etc. Typically this will be used when you want to load the subset in an already cloned instance. The second will export import the whole database.

**note:** Advise for OHI Back Office is to select the first option: "Only subset data".

Please note that the Advanced Compression and Advanced Security licenses are required in order to use the compress and encrypt options. Specify the log file name and press the Continue button. See Figure 3.19.

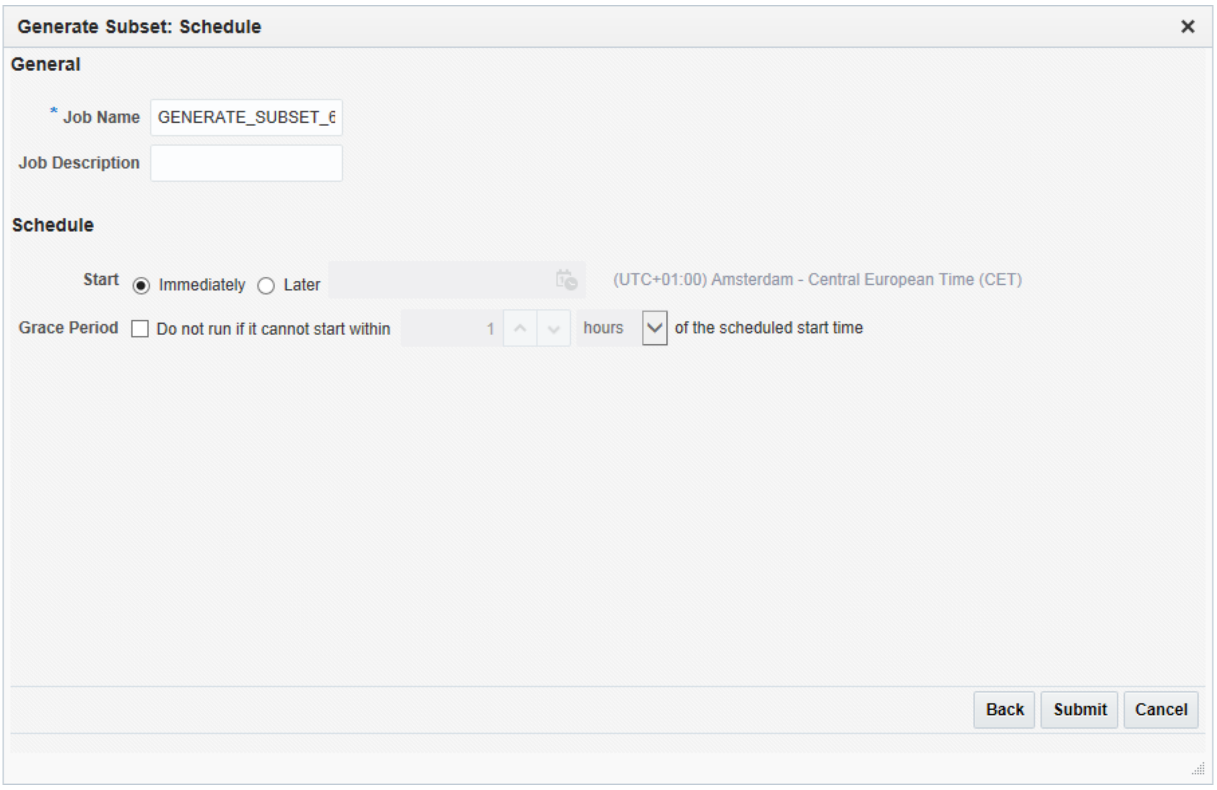


**Generate Subset: Parameters**

| | |
|---|---|
| Subset Directory | [✓ Select a directory on target database to save subset scripts and dump file] |
| | ADMIN_DIR |
| | ( /ohi/oraBase/product/db12102/rdbms/admin/ ) |
| | Select a custom directory path on target database to save subset scripts and dump file |
| | ☐ External directory(clustered/shared file system or ASM) for faster export dump |
| External Directory | ADMIN_DIR |
| | ( /ohi/oraBase/product/db12102/rdbms/admin/ ) |
| * Export File Name | EXPDAT%U.DMP |
| * Maximum File Size (MB) | 100 |
| * Maximum Number of Threads | 1 |
| Export File Contents | [✓ Only Subset Data] |
| | Full Database Export Along With Subset Data |
| | ☐ Enable Dump File Compression |
| | ☐ Enable Dump File Encryption |
| Encryption Password | |
| Confirm Encryption Password | |
| | ✓ Generate Log File |
| Log File Name | EXPDAT.LOG |

*Figure 3.19: Specify data pump parameters for the subset process.*

Finally schedule the job and submit it. See Figure 3.20.



* Job Name: GENERATE_SUBSET_6

Job Description:

Start: (● Immediately) ○ Later — (UTC+01:00) Amsterdam - Central European Time (CET)

Grace Period: ☐ Do not run if it cannot start within 1 hours of the scheduled start time

*Figure 3.20: Schedule and submit the subset process job.*

This job generates the following:

- *One or more data pump export files and a corresponding log file*
  These contain the subset of the data in the source database.
- *An import sql script (tdm_import.sql)*
  This script contains the commands that are used later to import the data pump export files
- *A post import sql script (tdm_post_script.sql)*
  This script contains commands that must be run on the target database after the data pump import has completed.

Once the subset generation job has completed, check the data pump log file for any errors.

Note: don't forget to revert the changes made to the database parameters when (re-)using the source database again as an OHI Back Office runtime or standby environment.

## 3.6 Provision Target Database

The data pump export that was created in the previous step, will have to be imported into a smaller target database. The export contains a user-mode export of the owner-schema of the OHI BO application. This is usually the OZG_OWNER schema. You have to prepare an 'empty' target database so that this user-mode export file can be successfully imported.

- *Make sure the database is loaded with the required options.*
  Those are currently the XDB and JVM options.
- *Make sure the instance is running with all the required (s)pfile settings.*
  Refer to the Oracle Health Insurance - Installation, Configuration and DBA Manual to verify these settings.
- *Make sure the database has all the necessary OHI BO tablespaces.*
  In addition to the SYSTEM, TEMP, UNDO, USER and SYSAUX tablespaces, there are currently eighteen required application tablespaces (ozg_fact_rel_tab, ozg_fact_rel_ind, etc.). Create these as small tablespaces and ensure that they can grow (autoextend on).
- *Make sure the database has the required OHI BO schemas and roles.*
  Running script *$OZG_BASE/sql/*OZGI001S*.sql* will create these. The import job ensures that these schemas and roles receive all the object privileges again.
- *Optionally create your own custom schemas and roles.*
  When your OHI BO environments have additional (custom) schemas, roles or both that have object grants from the OHI BO application owner schema, the import job successfully restores the object grants to these schemas also, provided that they are created in advance.

It is also advisable to disable the archivelog mode on the target database prior to starting the import job.

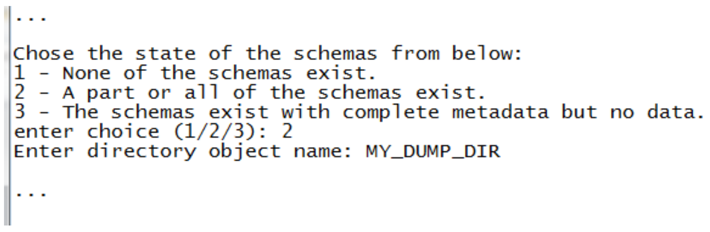
Note 1: It might be an idea to use a copy of an existing OHI BO database of the same OHI release as target database, drop all objects within the OHI BO owner schema (use an efficient ordering preventing exceptions and unnecessary invalidations), and possible recreate or resize the OHI BO related tablespaces.

Note 2: The above describes only the database-side of the target environment. Of course you also need a client environment (contents of $OZG_BASE) with the forms, reports, batch scheduler and so forth. This client environment can simply be a one-on-one copy of the client environment for the source.

## 3.7 Import Export File

The subset can now be loaded into the target database that was prepared in the previous section. You have to transfer the export dump files from the source database host to the target database host and create a directory object in the target database that points to the directory on the host holding the export files. Also transfer the *tdm_import.sql* and *tdm_post_script.sql* scripts that were generated in step 6.

Next start up SQL*Plus on the target host, connect as SYS and start the *tdm_import.sql* script. This script creates a few objects and then requests two inputs: the state of the schemas and the directory object name. Select option 2 for the first input and enter the name of the directory object that holds the export files. See Figure 3.21.



```
...

Chose the state of the schemas from below:
1 - None of the schemas exist.
2 - A part or all of the schemas exist.
3 - The schemas exist with complete metadata but no data.
enter choice (1/2/3): 2
Enter directory object name: MY_DUMP_DIR

...
```

*Figure 3.21: Enter data pump import parameters.*

Errors are logged during the import which causes objects in the OHI BO application schema to remain invalid due to a known issue in the way the subsetting pack interacts with data pump. These errors are resolved by the post import script.

You supplied a password for the application owner schema (typically OZG_OWNER) during preparation of the target database (as described in the previous Section). The tdm_import.sql script drops this schema and has it recreated by the data pump import process. This means that you lose the password that you have supplied: after the import, it is again set to the password that was in place in the source environment.

## 3.8 Run the Post Import Script File

It is assumed this step is run from the application server that is prepared for the target environment as a connection to the OHI BO table owner account is made using the wallet. OHI environment variables should be present.

Start SQL*Plus on the application server for the target database, connect as SYS and run the post import script: *tdm_post_script.sql*. To run the post import script on the target, first set the environment correctly, i.e.:

```
. ozg_init.env <env>

. ozg_init.env $OZG_ORATAB_DB19
```

Make sure the [SID]_install wallet entry exists and connects to the OHI BO table owner schema of the target database (you may want to reset the application owner password in the target database at this stage).

After running this script all stored PL/SQL objects should be valid. You now have a subset of the OHI BO data store. Note however that some tasks still remain. For instance, you still need to set up the application users. This target database will have the ALG#USERS_ table contents (Dutch name: ALG_FUNCTIONARISSEN) of the source database. You may want to update this table and create the corresponding Oracle application user schemas for it.

As a final verification you can run the object check script for the OHI BO application (script *SYS9006S*). Note: This script assumes that the batch scheduler has been started. Whether the client and database are in a correct state is verified by Script *SYS9006S*.

## 3.9 Install and Configure Other Custom Localizations

Any adjacent schemas containing for instance custom code and data should be installed and configured as the last step for local customizations. If you need to subset the custom data also you need to develop your own custom subset implementation for this.

# 4 Detailed Process - Data Masking

Contrary to subsetting, where an empty target database is filled with a subset of data, the data masking process executes "in-place" within an existing OHI BO or OHI DM data store.

## 4.1 Preparing to-be-masked Database (Target Database)

Data masking can be performed on a full-size OHI BO or OHI DM data store, or on a subset OHI BO or OHI DM data store.
**Note:** A subset of an OHI DM data store can be obtained by performing an initial load of a subset OHI BO environment, see chapter 5. Typically, you would first mask that OHI BO data store before performing an initial load to construct the companion OHI DM data store. Masking an OHI DM data store is typically used for a full sized OHI DM environment.

Obviously the masking process takes more time on full-size data stores. During masking, tables with sensitive columns are temporarily duplicated (this is further explained in Section 4.3). For this reason it is necessary to check that tablespaces holding the table segments either have enough free space available, or are able to grow (autoextend) during the masking process. Upon completion of data masking a database (or tablespace) reorganization of some kind will have to be performed to reclaim the then remaining free space. Oracle describes an approach in Section 4.3 to prevent having to execute such reorganization.

Obviously, masking should never be executed on a production environment. Therefore it is required to adjust the value of the application environment parameter in the OHI system parameter record to “Test” instead of “Production” ( alg_systeem_parameters.applicatie_omgeving ).

Before masking the target database, it is advisable to create a backup of the database. Depending on the size of the archivelog destination file system, it may be advisable to disable archive logging during the masking process, and, when required, re-enable it afterwards.

**Recommended database configuration during masking**

Before running the masking process on the target, it should be configured to maximize performance. The following parameters are recommended (and might deviate from parameters required for OHI Back Office or OHI Data Marts at runtime):

- `parallel_max_servers` - default value (>1) or (for example) twice the number of available CPU threads, to enable parallel query during masking
- `pga_aggregate_target` – parallel_max_servers * 1GB

**Masking a Autonomous Data Warehouse**

When masking is performed for OHI Data Marts on an ADW environment it is recommended to use the “high” connection service.

**Checks before masking**

The masking program processes one table at a time. It creates a copy of the original table and deletes that copy after the table has been masked successfully. This means that the masking program requires a certain amount of free space in the original tablespace or in the specified “temporary” tablespace. If the masking of a table fails, the copy is not removed (to support a quick retry after the cause of the failure has been addressed), and processing continues with the next table.

Therefore, the free space needed will be at least the size of the largest table, plus a reserve for any tables that fail to be processed successfully.

To get an indication of the minimum free space that needs to be available, a pre-check can be executed on the target database before the masking is started
This check will check some environment settings and will provide a list of the top 5 tables in size.

The pre-check can be executed by running the following command as schema owner in SQL*Plus:

```
set serveroutput on
exec sdm_driver_pck.pre_check;
```

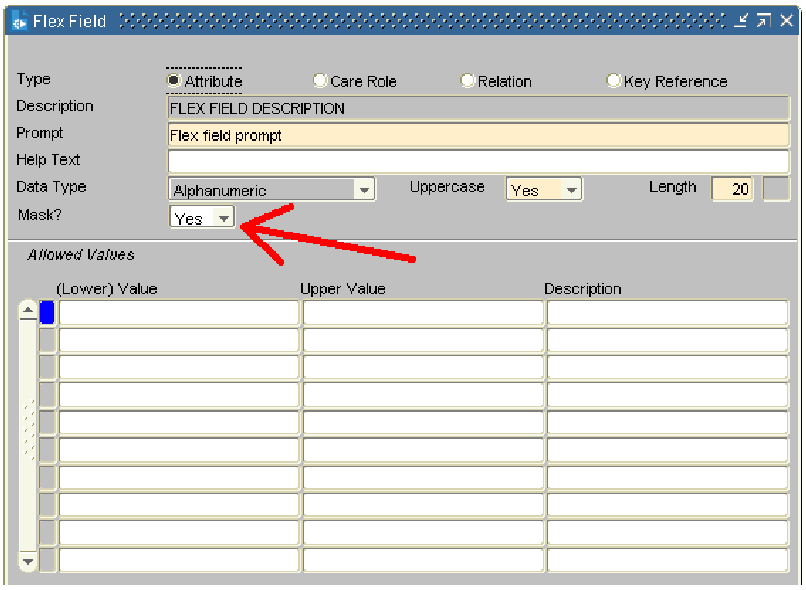
The output looks like the following:

```
Pre masking checks
Batch scheduler is stopped (ok).
Environment is not a production environment (ok)
Back Office parameter 'Default country code' is set to 'NL' (ok).
Database is in restricted session mode

Top 5 tables to be masked based on size
---------------------------------------
GEB_DER_EIGENSCHAP_WAARDEN: 38,075 GB
FVS_FINANCIEEL_FEIT_REGELS: 18,169 GB
GEB_DER_CORRECT: 16,9 GB
FSA_VERPL_JOURN_REGELS: 13,008 GB
GEB_DUS_RISICOFACTOREN: 12,723 GB
```

## 4.2 Masking flex fields

Flex fields are flexible, like the name says, and your organization can determine what they are used for and what is stored in them. As such, it may be that certain flex fields contain sensitive information that should also be masked.

In the OHI BO application it is possible to indicate if the value for a specific flex field should be masked. In window ZRG7019F (Flex field) the indication “Mask?” can be set to “Yes” in order to mask the value for this flex field. If set to “No” (default value) masking will not take place. Masking the OHI DM application will also use this setup in OHI BO.



Type: [✓ Attribute] Care Role Relation Key Reference
Description: FLEX FIELD DESCRIPTION
Prompt: Flex field prompt
Help Text:
Data Type: Alphanumeric
Uppercase: Yes
Length: 20
Mask?: Yes

*Allowed Values*

| (Lower) Value | Upper Value | Description |
|---|---|---|

**Note:** The value of a flex field can be used to influence the logic of processes like the claims processing. Masking of these flex fields can interfere with this logic and may lead to unpredictable and/or undesirable results. Therefore, the advice is to mask only those flex fields that are actually privacy-sensitive.

## 4.3 Running Masking on Target Database

To start the masking process a database job can be created. To do this execute the following command as the OHI database scheme owner:

```
begin
  sdm_driver_pck.job
  ( pi_seed                    => <your 'secret' integer value>
  , pi_tablespace              => 'SCRATCH_TBS'
  , pi_job_class               => 'DEFAULT_JOB_CLASS'
  , pi_force_query             => 'N'
  , pi_force_dml               => 'Y'
  , pi_force_ddl               => 'Y'
  , pi_parallel_policy_manual => 'Y'
  );
end;
/
```

This will start a database scheduler job with the name OHI_MASKING and run the SDM_DRIVER_PCK.MASK procedure.
It is also possible to start the mask procedure directly in the calling session, but this will require you to keep the executing SQL*Plus session open and alive.

The parameters do have the following impact/effect during the masking process:

- PI_SEED: this can have a value between 0 and 4294967295. This will randomize the masking output and prevent identical masked strings for identical source strings for different OHI Back Office customers.
  **Note:** If you want to mask an OHI Back Office environment and a OHI Data Marts

environment that should be coupled together make sure the value of the seed is the same, so make sure to store it in a safe place.

- PI_TABLESPACE: when provided each table to be masked will be moved to this tablespace before it is masked. For this an existing but empty tablespace should be provided.
  If no tablespace is provided the temporary table object is created in the same tablespace as the table to be masked. Providing a tablespace might increase the masking time, because the tables to be masked will first have to be moved to the chosen tablespace, but will have the benefit (after dropping this tablespace) that the size of the original tablespaces is not significantly increased after masking.
- PI_JOB_CLASS: The scheduler job class this job will use as a resource group. For an Autonomous Data Warehouse environment use the job class best fit for the resource-intensive masking job, e.g. 'HIGH'.
- PI_FORCE_QUERY: when Y(es) the session is set to FORCE parallel query, when N(o) the session is set to ENABLE parallel query
- PI_FORCE_DML: when Y(es) the session is set to FORCE parallel DML, when N(o) the session is set to ENABLE parallel dml
- PI_FORCE_DDL: when Y(es) the session is set to FORCE parallel DDL, when N(o) the session is set to ENABLE parallel DDL
- PI_PARALLEL_POLICY_MANUAL: when Y(es) the session is set to MANUAL for the parameter PARALLEL_DEGREE_POLICY, when N(o) the session is set to AUTO for the parameter PARALLEL_DEGREE_POLICY.
  **Note:** This parameter will not be set in an Autonomous Data Warehouse environment.

All parameters are provided with a default which is the OHI recommended setting, except for the tablespace name.

These settings will make maximum use of the available resources. If it is not possible or if you don't want to use the maximum available resources you can set these parameters to the other possible value.

## 4.4 Monitoring the masking run

There is no direct output during the masking run. However information about the run is stored in a table.

The current action is shown in the first row in the following query which can be run under the OHI schema owner. The schema owner in the queries is the owner of the tables that are (being) masked:

```
select mlg.*
from sdm#masking_log mlg
where mlg.schema_owner = '<schema owner>'
order by mlg.id desc;
```

After the initialization phase is done, the overall progress can be monitored by looking at the status of the tables with the following query:

```
select mdd.base_table
,      mdd.status
from sdm#statement_data ssd
where ssd.object_type = 'TABLE'
```

```
and   ssd.schema_owner = '<schema owner>'
order  by mdd.id;
```

Tables with status D(one) have been processed successfully.

## 4.5 Checking the outcome

After the masking job and/or procedure is finished a report can be created to check on the process and possible errors. To create the report the following script can be run under the masked schema owner

```
col spool_tijdstip new_value l_spool_tijdstip noprint
select to_char(sysdate,'YYYYMMDD_HH24_MI_SS') spool_tijdstip
from dual
/

set trimspool on
set trim on
set pages 0
set linesize 1000
set long 10000000
set longchunksize 10000000
set feedback off
set showmode off
set flush off
set verify off
spool MASKING_REPORT_&l_spool_tijdstip..html

select * from table(sdm_driver_pck.report(pi_full => 'N'))
/
spool off
undefine l_spool_tijdstip
```

This will create an HTML file with the following information

- Provided parameter values
- Instance and session information
- Generic run information
- Top 20 long running actions
- Any errors that occurred

When the parameter pi_full is provided with the value 'Y' the report will provide a full report of all executed steps. This option can be requested when contacting OHI support.

## 4.6 Processing errors and restarting

If one or more errors occurred the process can be restarted by using the same parameters, after mitigating the cause of the error, e.g. by increasing a tablespace size.
In that case, the process will detect an incomplete previous run and only process the tables that are not yet successfully masked.

## 4.7 Cleaning up

If you had moved the tables with sensitive columns to an empty tablespace (as described earlier) you can drop that tablespace now, to release the extra disk space used by the masking process.

Revert the changes made to the database parameters before using the database as an OHI Back Office or OHI Data Marts runtime environment.

# 5 OHI Data Marts subset

From release 10.16.1.0.0 onwards OHI Data Marts (OHI DM) is certified to extract data from sub-setted and/or data-masked OHI Back Office (OHI BO) data stores.

The following types of OHI BO data stores are supported:

- subsetted
- subsetted and data-masked
- data-masked

From release 10.18.1.0.0 onwards, OHI DM can be data-masked itself as well. The process to mask OHI DM is described in chapter 4.

When a subset OHI DM environment is needed you should use the approach described in this chapter. The fact if the source OHI BO environment is already masked determines whether the resulting OHI DM environment is masked. For creating a subset OHI DM environment an initial load is needed from the OHI BO subset environment.

When you need a full size masked OHI DM environment the masking process from the previous chapter can be applied.

Theoretically you can create a subset OHI BO environment and corresponding OHI DM environment and mask them both indepently but this is more time consuming and more error-prone than using a subset masked OHI BO environment to create the corresponding OHI DM environment.

## 5.1 Preparing OHI Data Marts data-store

An empty OHI DM data store is required to be able to load data from a subsetted and/or data-masked OHI BO data store into OHI DM. The SQL script *OBDRESET.sql* (available within *OZG_TEMPLATES.zip* as of release 10.16.1.0.0) is provided to create an empty OHI DM data store by truncating all the necessary OHI DM tables. This script can be run using SQL*Plus connected to the OBD_OWN account.

It is strongly advised to create a clone from an existing OHI DM environment which is at the same OHI patch-level as the OHI BO data store, and use this cloned environment to run the SQL script *OBDRESET.sql* against. Make sure the database link SRC_OPENZORG is referring to the correct OHI BO data store.

NOTE: This SQL script should never be used within a production environment!

## 5.2 Loading OHI Data Marts

Performing loads from an subsetted and/or data-masked OHI BO environment is identical to loading from a normal OHI BO environment. See the OHI Back Office online help for information on loading (topic ‘Loading OHI Data Marts’).

You can proceed with a full initial load from the OHI BO subset environment by not specifying up to which moment to load. Or you choose to first load older data up to a specific date and divide the work in this way, by using additional incremental loads for later periods to load.

# 6 Appendix 1: Subsetting Step-by-step execution guide

This appendix provides a step-by-step guide to execute the subsetting process. In this guide there will be references to the 'source database' and the 'target database'. The source database is the database where the subset will be generated on, this database contains the data that will be included in the subset export. The target database will be the database on which the subset will be loaded.

The guide focusses on creating a subset including a policy selection. The guide can also be used for a configuration only subset by skipping steps: 6.2 and 6.3.27.

## 6.1 Preparation

For a more detailed description of all the prerequisites needed please see the paragraph Prerequisites.

1. Restore a backup on the source database that will be used as the base for the subset.
2. Run step 900 of OHIPATCH (of the corresponding release currently installed) on the source database and make sure that everything is valid.
3. Connect with SQL*Plus to the source database (as SYS or any other user that can create users and issue grants).
4. Create a (temporary) subset user for the creation of the subset export. This can be done with the following statements:

```
create user <USER> identified by <PASSWORD> default tablespace <SCRATCH TABLESPACE>

temporary tablespace <TEMP TABLESPACE THAT MAY GROW>;

grant dba to <USER>;

grant execute on DBMS_AQADM to <user>;
```

## 6.2 Creating a policy selection

For a more detailed description of the process including creating a dataset for subsetting from within OHI BO please see: Define Policy or Claim line Selection in the Source Database

1. Create a policy selection. This can be done by using the statements below (run under a user that has insert rights on these tables). In the statements the name (SAMPLE_OF_5_PERCENT) can be changed to your liking. The where clause of the query can be edited to include or exclude certain specific policies from the subset.

```
insert into ver#policy_selections_ (name) values('SAMPLE_OF_5_PERCENT');

insert into ver#pol_per_pos_ ( pos_id, pol_num )

select ( select id from ver#policy_selections_ where name = 'SAMPLE_OF_5_PERCENT')

       , num

from ver#policies_ SAMPLE(5);

commit;
```

2. But be sure to write down the chosen name of the subset since you will need it later on when generating the subset export (step 27).

## 6.3 Generating the subset export

For a more detailed description of the process including the setup of the environment and the motivation why certain steps should be executed please see: Detailed Process - Subsetting

1. Connect with SQL*Plus to the source database. Invoke the following command (as OHI Back Office table owner):

   ```
   exec sdm_adm_drv_pck.write_adm_files('DB_DIR');
   ```

   Replace DB_DIR with the database directory of your choice.

2. Transfer the resulting file (*SDM_OHI_[release]_MASK_ADM.xml*) to a file system that is accessible from your Desktop.
3. Login to Oracle Enterprise Manager.
4. Go to: Enterprise → Quality Management → Application Data Modeling. See Figure 6.1.

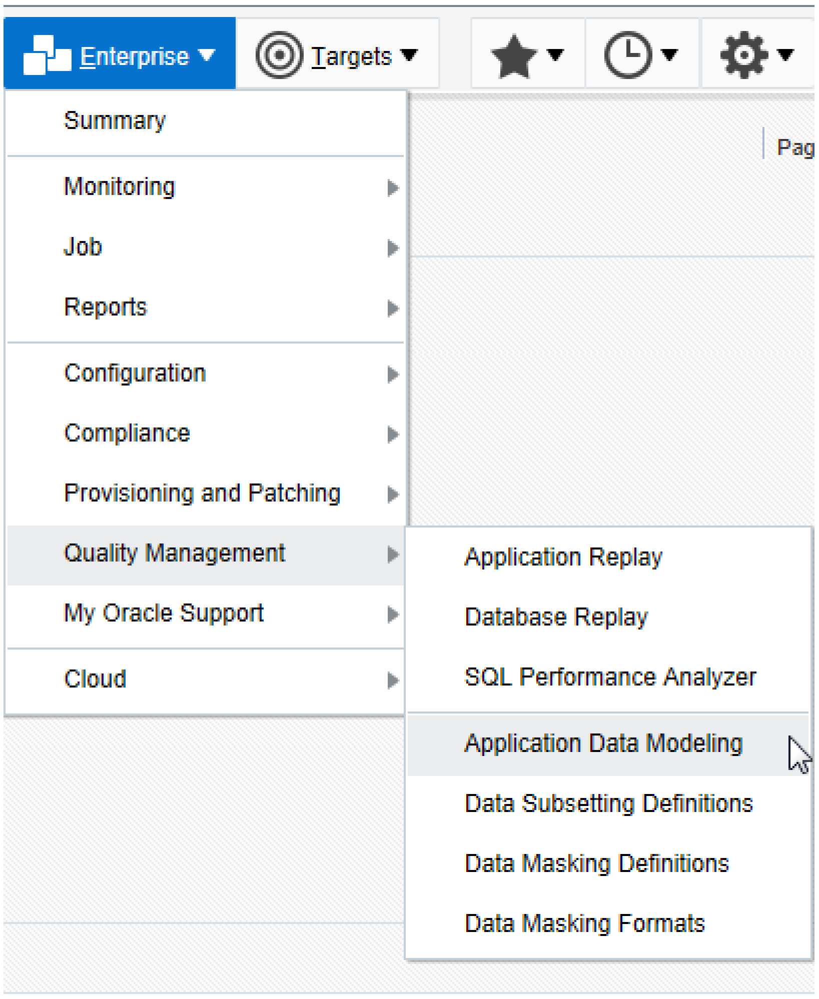


*Figure 6.1: Open Application Data Models*

5. Make sure the ADM XML file (*SDM_OHI_[release]_SUBSET_ADM.xml*) is on your local desktop. Then select Actions → Import → File from Desktop. See Figure 6.2.

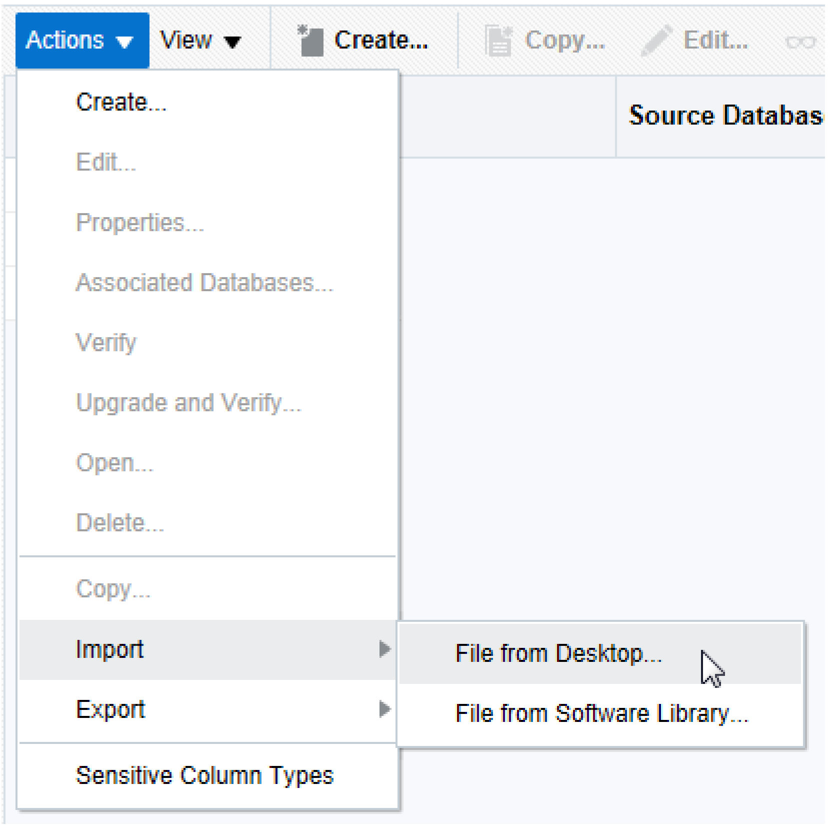
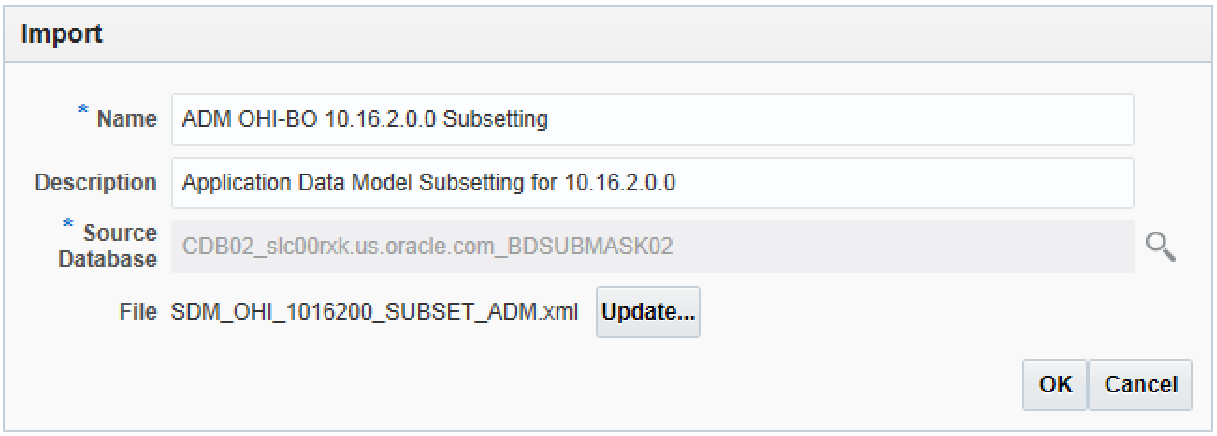


*Figure 6.2: Starting the File from Desktop import.*

6. Enter a name for this ADM in the pop-up that opens. Specifying the release number as part of the name of the ADM will help identify the correct ADM later on (e.g. "ADM OHI-BO [release] Subsetting")
7. Enter a description and select the subsetting source database.
8. Press "Choose File" and navigate to the ADM XML file (*SDM_OHI_[release]_SUBSET_ADM.xml*) on your local desktop which was generated in step 1. Finally press the OK button to start the import. See Figure 6.3.

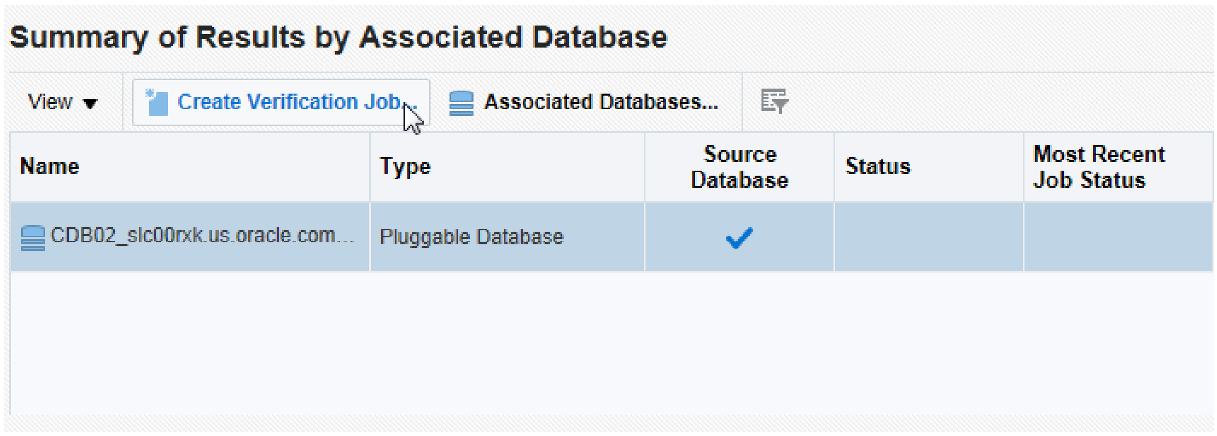


**Import**

| | |
|---|---|
| * Name | ADM OHI-BO 10.16.2.0.0 Subsetting |
| Description | Application Data Model Subsetting for 10.16.2.0.0 |
| * Source Database | CDB02_slc00rxk.us.oracle.com_BDSUBMASK02 |
| File | SDM_OHI_1016200_SUBSET_ADM.xml |

*Figure 6.3: Specifying Subsetting ADM XML file to be imported.*

9. After the import process has completed, select the imported ADM in the list of available ADMs. Then select Actions → Verify, in the next screen click Create Verification Job. See Figure 6.4.

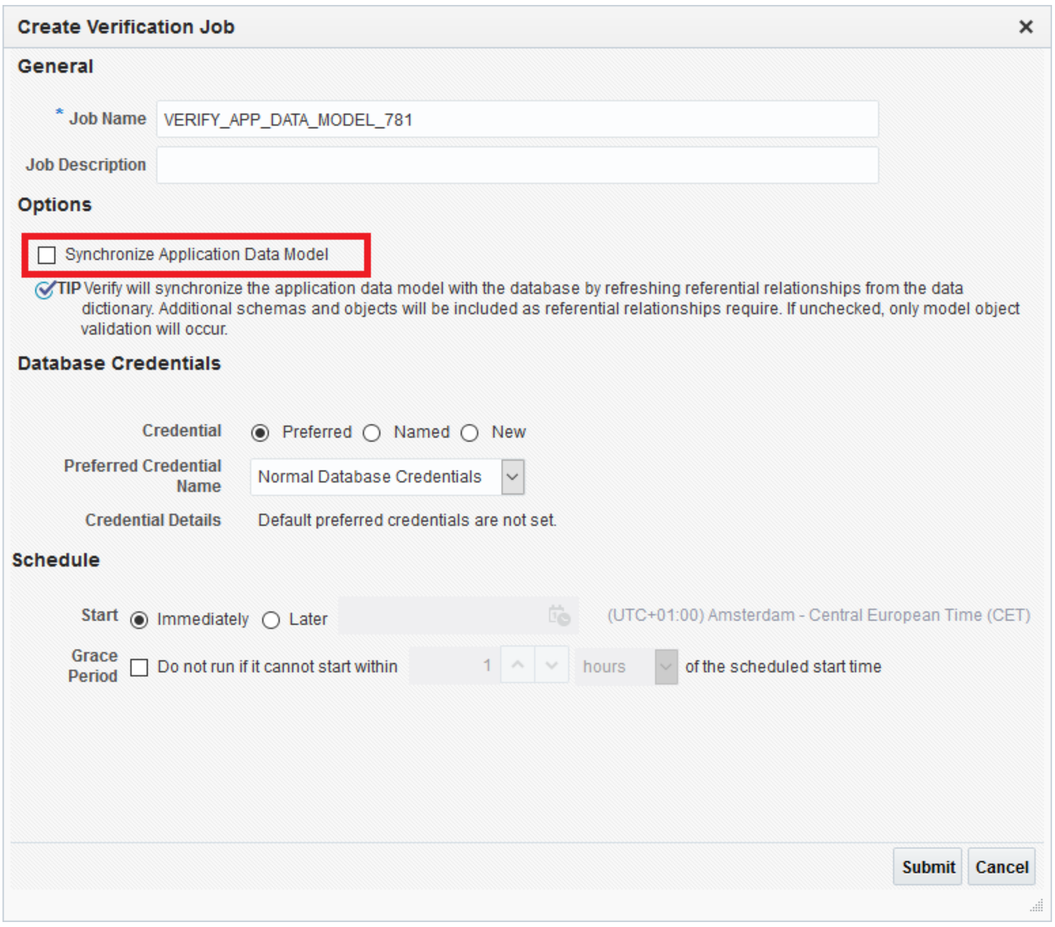


**Summary of Results by Associated Database**

| Name | Type | Source Database | Status | Most Recent Job Status |
|---|---|---|---|---|
| CDB02_slc00rxk.us.oracle.com... | Pluggable Database | ✓ | | |

*Figure 6.4: Create Verification Job.*

10. Make sure to **uncheck** Synchronize Application Data Model before submitting. Provide the correct credentials to connect to the source database. The other options can be kept default. See Figure 6.5.



**Create Verification Job**

**General**

* Job Name: VERIFY_APP_DATA_MODEL_781

Job Description:

**Options**

☐ Synchronize Application Data Model

TIP Verify will synchronize the application data model with the database by refreshing referential relationships from the data dictionary. Additional schemas and objects will be included as referential relationships require. If unchecked, only model object validation will occur.

**Database Credentials**

Credential: [✓ Preferred] Named New

Preferred Credential Name: Normal Database Credentials

Credential Details: Default preferred credentials are not set.

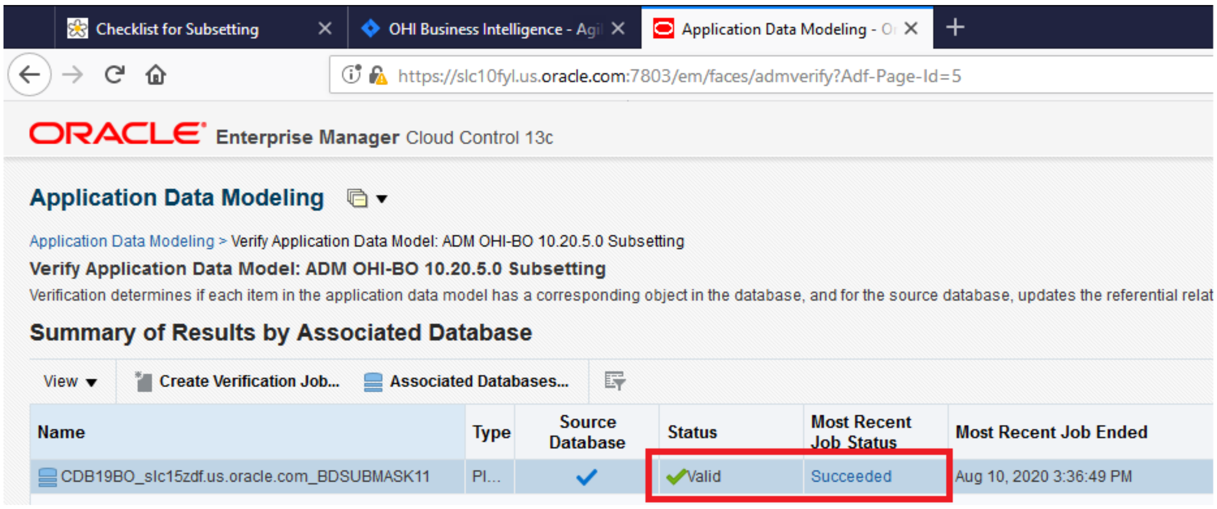
**Schedule**

Start: [✓ Immediately] Later (UTC+01:00) Amsterdam - Central European Time (CET)

Grace Period: ☐ Do not run if it cannot start within 1 hours of the scheduled start time

Figure 6.5: Uncheck Synchronize Application Data Model.

11. Wait till the verification job has completed. This can take some time. See Figure 6.6.



https://slc10fyl.us.oracle.com:7803/em/faces/admverify?Adf-Page-Id=5

ORACLE Enterprise Manager Cloud Control 13c

**Application Data Modeling**

Application Data Modeling > Verify Application Data Model: ADM OHI-BO 10.20.5.0 Subsetting

**Verify Application Data Model: ADM OHI-BO 10.20.5.0 Subsetting**

Verification determines if each item in the application data model has a corresponding object in the database, and for the source database, updates the referential relat

**Summary of Results by Associated Database**

| Name | Type | Source Database | Status | Most Recent Job Status | Most Recent Job Ended |
|---|---|---|---|---|---|
| CDB19BO_slc15zdf.us.oracle.com_BDSUBMASK11 | Pl... | ✓ | ✓Valid | Succeeded | Aug 10, 2020 3:36:49 PM |

Checklist for Subsetting
OHI Business Intelligence - Agi
Application Data Modeling - O
View
Create Verification Job...
Associated Databases...

Figure 6.6: Verification Job Succeeded

12. Connect with SQL*Plus to the source database.
13. Invoke the following command (as OHI Back Office table owner):

```
exec sdm_subset_drv_pck.write_file(DB_DIR);
```

Replace DB_DIR with the database directory of your choice.

14. Once this file (SDM_OHI_<release nr>_SUBSET_DEF.xml) has been generated, transfer this to a file system that is accessible from your Desktop, so you can import it using Enterprise Manager.
15. Within Enterprise Manager navigate to Enterprise → Quality Management → Data Subsetting Definitions. See Figure 6.7.

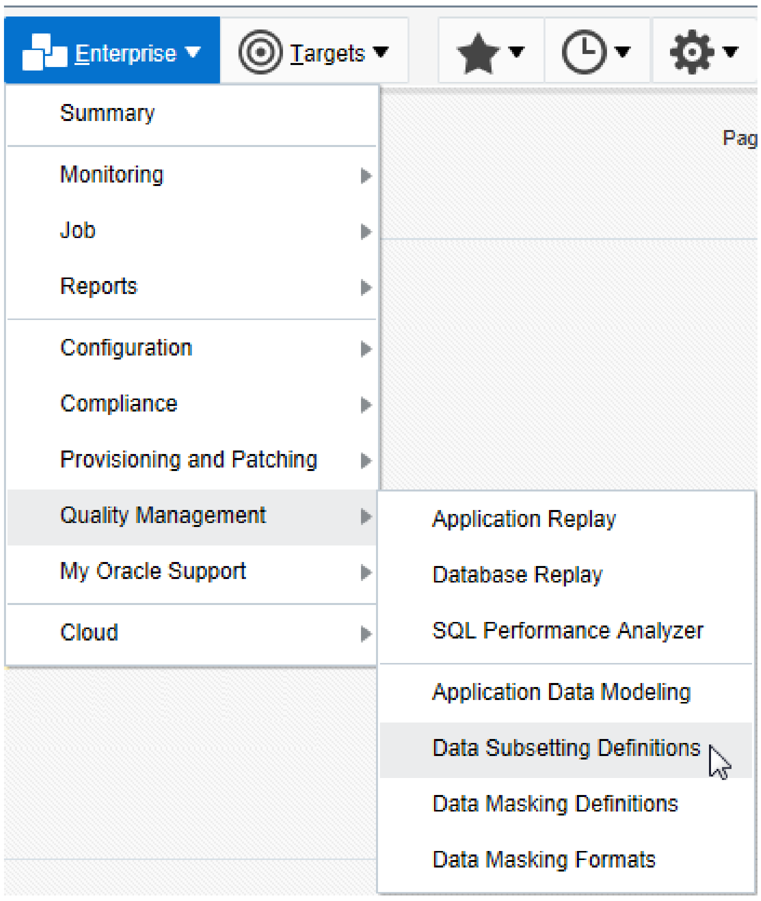


*Figure 6.7: Open Data Subset Definitions*

16. Select Actions → Import → File from Desktop in the Data Subset Definitions page. See Figure 6.8.

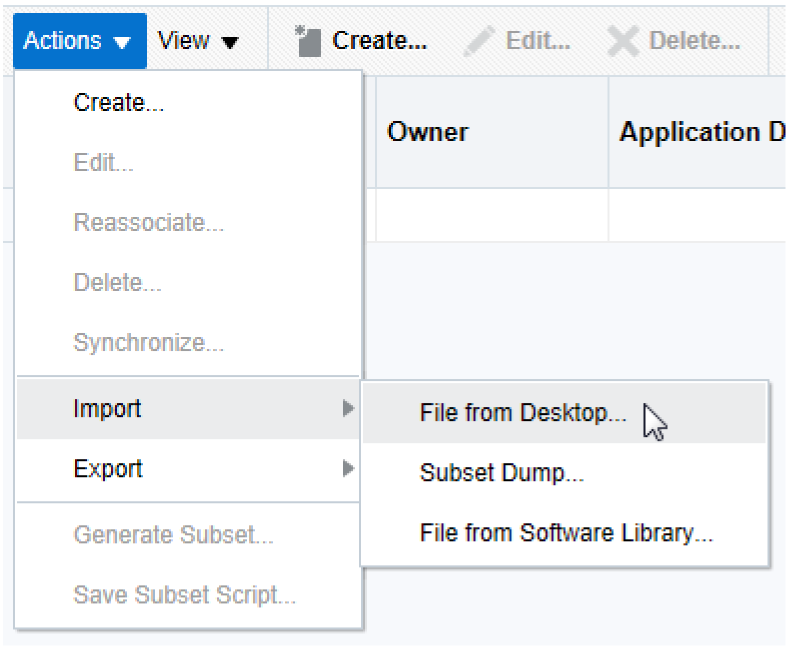
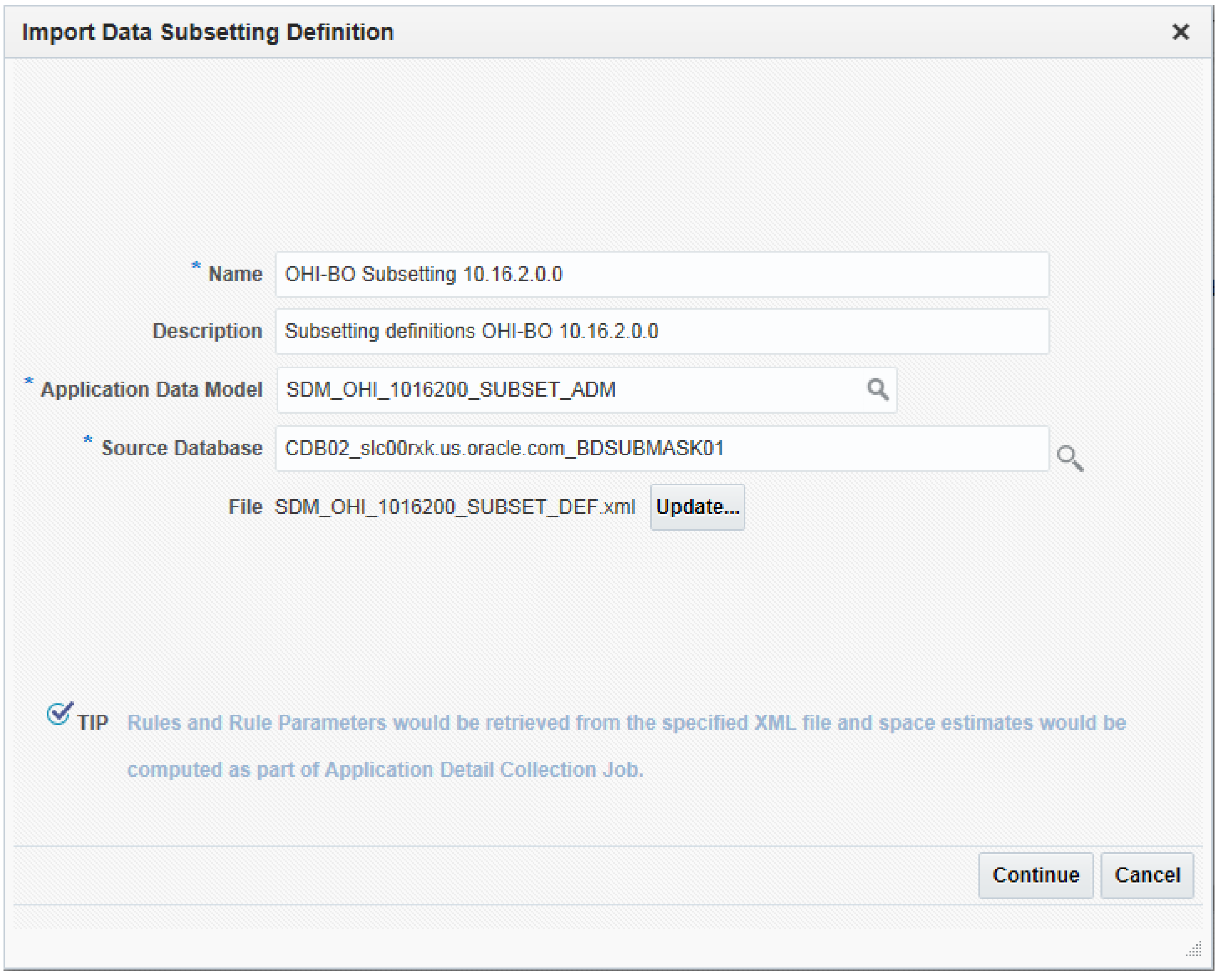


*Figure 6.8: Starting the File from Desktop import.*

17. Provide a name for the Subset Definition in the pop-up that opens. Specifying the release number as part of the name of the definition will help identify the correct subset definition later on (e.g. “*OHI-BO Subsetting [release]*”).
18. Select the **corresponding ADM** which was imported in step 8.
19. Select the source database.
20. Finally select the subsetting XML file from your local desktop and press the Continue button. See Figure 6.9.

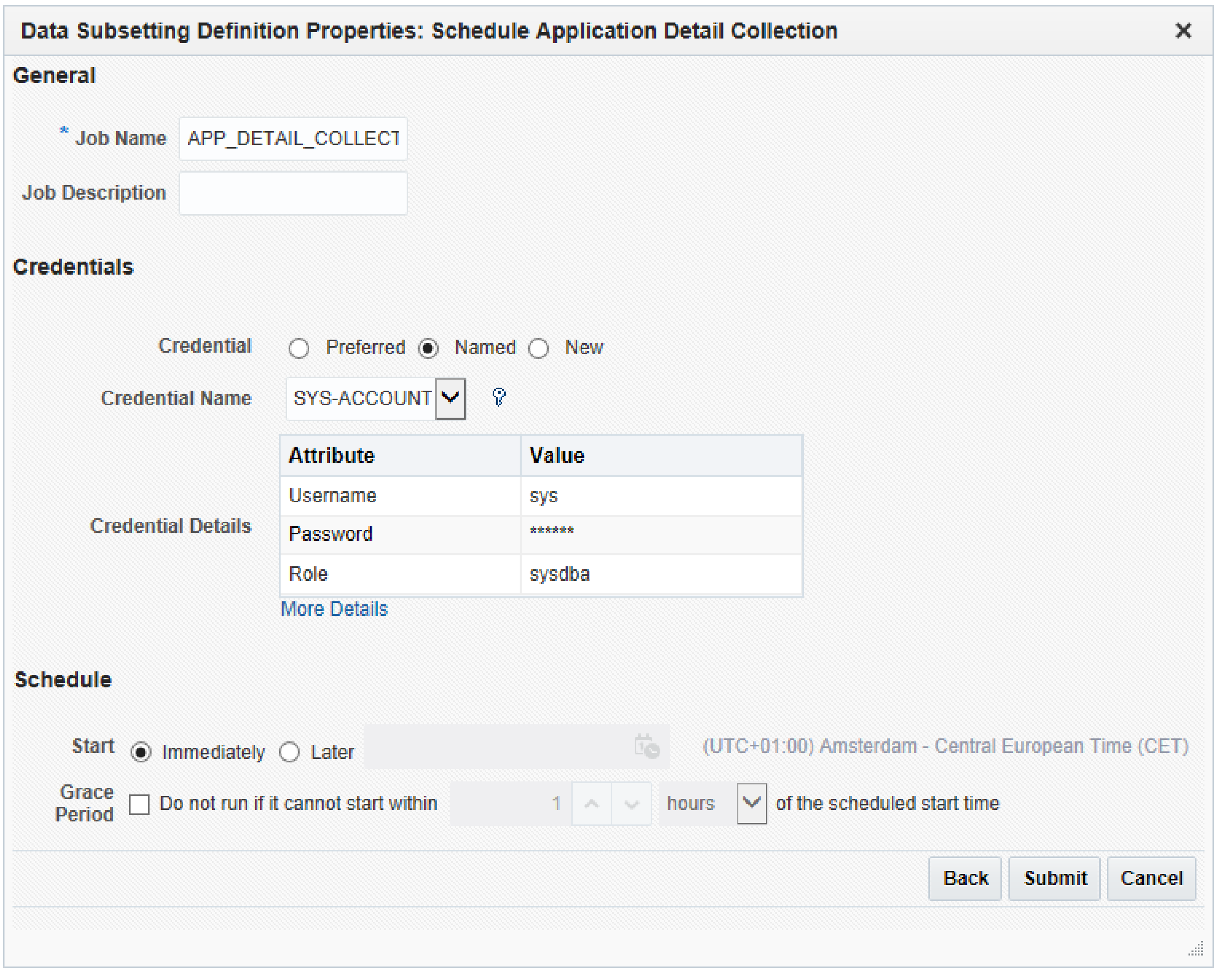

Import Data Subsetting Definition

| | |
|---|---|
| * Name | OHI-BO Subsetting 10.16.2.0.0 |
| Description | Subsetting definitions OHI-BO 10.16.2.0.0 |
| * Application Data Model | SDM_OHI_1016200_SUBSET_ADM |
| * Source Database | CDB02_slc00rxk.us.oracle.com_BDSUBMASK01 |
| File | SDM_OHI_1016200_SUBSET_DEF.xml |

TIP Rules and Rule Parameters would be retrieved from the specified XML file and space estimates would be computed as part of Application Detail Collection Job.

*Figure 6.9: Import Data Subset Definition.*

21. Specify the subset user credentials of the source database in the next page and press the Submit button. See Figure 3.14.


Data Subsetting Definition Properties: Schedule Application Detail Collection

**General**

| | |
|---|---|
| * Job Name | APP_DETAIL_COLLECT |
| Job Description | |

**Credentials**

Credential: Preferred [✓ Named] New

Credential Name: SYS-ACCOUNT

Credential Details:

| Attribute | Value |
|---|---|
| Username | sys |
| Password | ****** |
| Role | sysdba |

More Details

**Schedule**

Start: [✓ Immediately] Later (UTC+01:00) Amsterdam - Central European Time (CET)

Grace Period: ☐ Do not run if it cannot start within 1 hours of the scheduled start time

*Figure 6.10: Specifying credentials for import job.*

22. Verify the job succeeded. This job should only run for a couple of minutes.

23. Connect with SQL*Plus to the source database.
24. Invoke the following command (as OHI Back Office table owner):

```
exec sdm_util_pck.purge_sdm_data;
```

25. Within Enterprise Manager on the Data Subsetting Definition page make sure the just imported definition is selected/highlighted and select: Action → Generate Subset. See Figure 6.11.

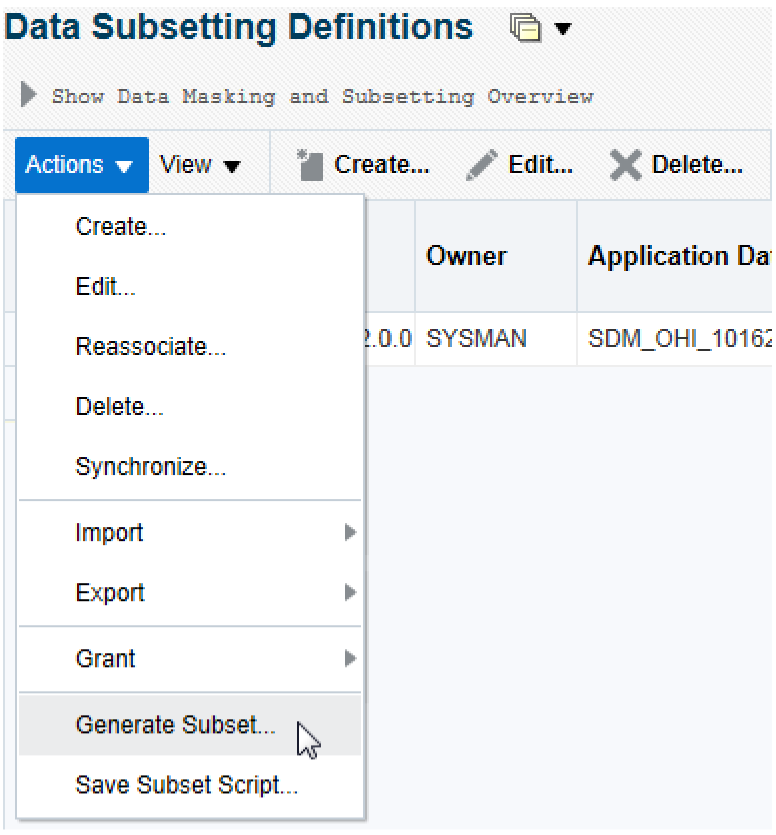
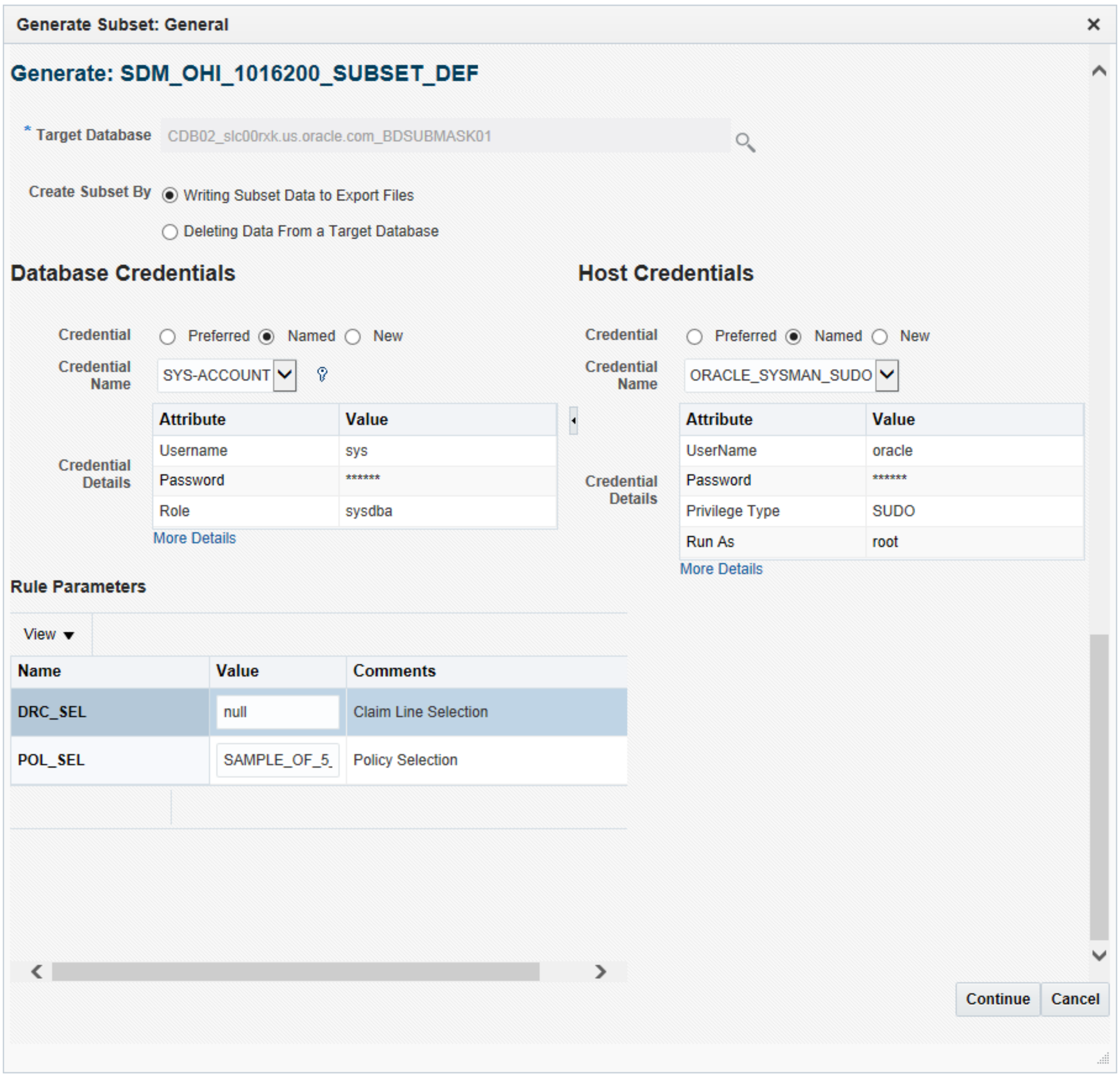


*Figure 6.11: Generate Subset.*

26. In the pop-up that opens keep the 'Create Subset By' radio button default (Writing Subset Data to Export Files).
27. Provide the database and host credentials. As database credentials use the user that was created in step 6.2.1.
28. Under 'Rule Parameter' provide the name of the created policy selection under the POL_SEL parameter. Make sure this is the exact name of the policy selection created under step 6.2.1.
29. Press the Continue button. See Figure 6.12.



Generate Subset: General

Generate: SDM_OHI_1016200_SUBSET_DEF

* Target Database: CDB02_slc00rxk.us.oracle.com_BDSUBMASK01

Create Subset By: [✓ Writing Subset Data to Export Files] Deleting Data From a Target Database

**Database Credentials**

Credential: Preferred [✓ Named] New

Credential Name: SYS-ACCOUNT

Credential Details:

| Attribute | Value |
|---|---|
| Username | sys |
| Password | ****** |
| Role | sysdba |

More Details

**Host Credentials**

Credential: Preferred [✓ Named] New

Credential Name: ORACLE_SYSMAN_SUDO

Credential Details:

| Attribute | Value |
|---|---|
| UserName | oracle |
| Password | ****** |
| Privilege Type | SUDO |
| Run As | root |

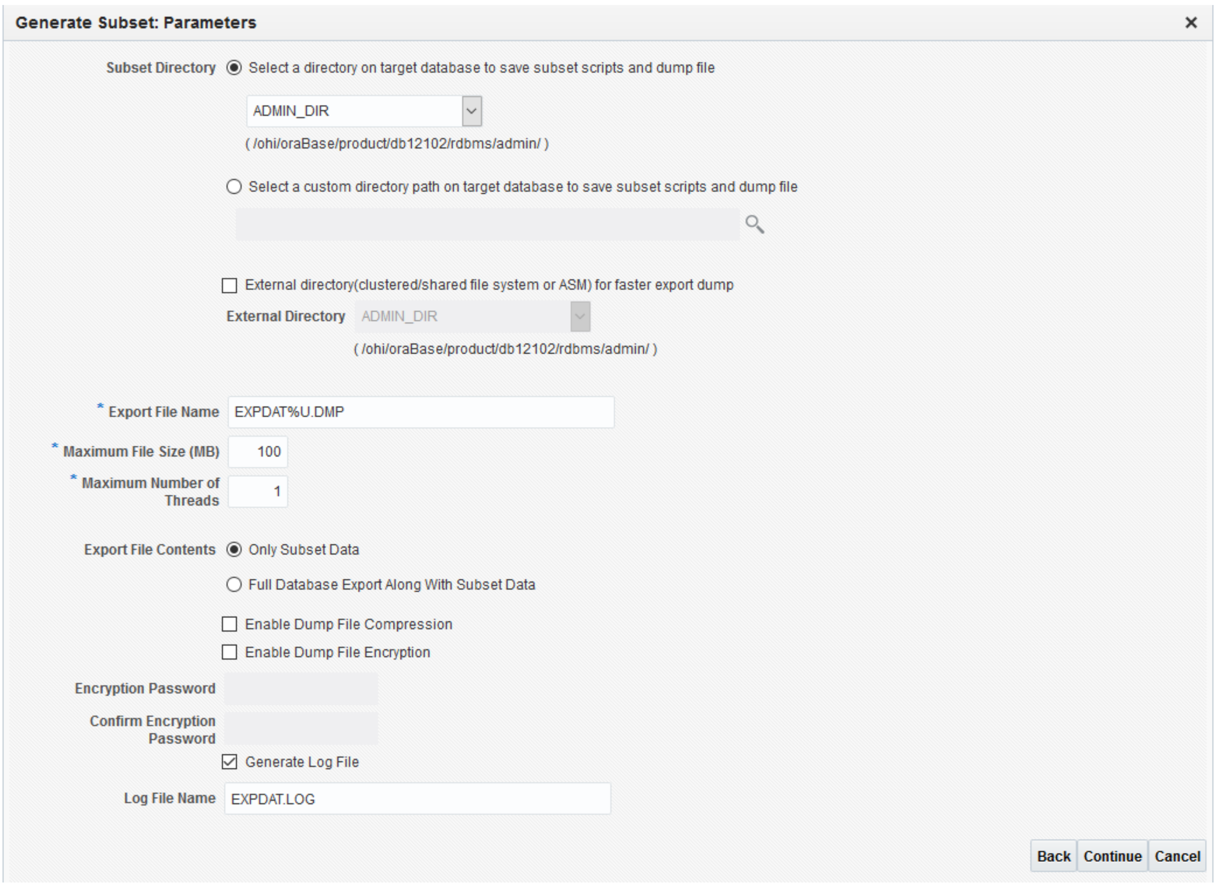
More Details

**Rule Parameters**

| Name | Value | Comments |
|---|---|---|
| DRC_SEL | null | Claim Line Selection |
| POL_SEL | SAMPLE_OF_5_ | Policy Selection |

*Figure 6.12: Specify general parameters for subset process.*

30. On the next page you will have to select the data pump parameter. Select a directory where the subset export should be placed. Make sure that the database user that was provided in step 26 has read/write access on this directory object.
31. Provide a name for the export (Export File Name).
32. Provide the maximum file size for the export files. Note that only a maximum of 100 files can be created. The provided file size *100 should be at least enough to contain the entire size of the subset.
33. Make sure that under Export File Contents ‘Only Subset Data’ is selected.
34. Click on ‘Continue’. See Figure 6.13.

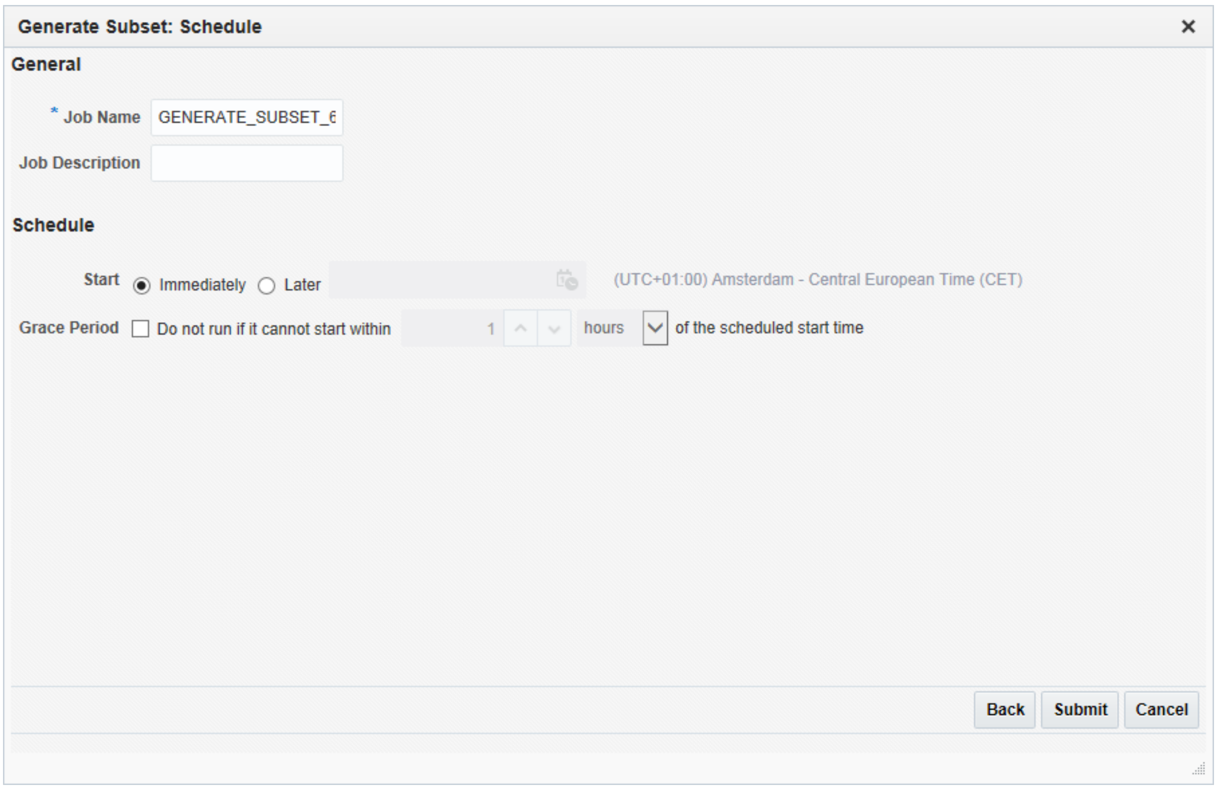


**Generate Subset: Parameters** ×

| | |
|---|---|
| Subset Directory | [● Select a directory on target database to save subset scripts and dump file] |
| | ADMIN_DIR |
| | ( /ohi/oraBase/product/db12102/rdbms/admin/ ) |
| | ○ Select a custom directory path on target database to save subset scripts and dump file |
| | ☐ External directory(clustered/shared file system or ASM) for faster export dump |
| External Directory | ADMIN_DIR |
| | ( /ohi/oraBase/product/db12102/rdbms/admin/ ) |
| * Export File Name | EXPDAT%U.DMP |
| * Maximum File Size (MB) | 100 |
| * Maximum Number of Threads | 1 |
| Export File Contents | [● Only Subset Data] |
| | ○ Full Database Export Along With Subset Data |
| | ☐ Enable Dump File Compression |
| | ☐ Enable Dump File Encryption |
| Encryption Password | |
| Confirm Encryption Password | |
| | [✓ Generate Log File] |
| Log File Name | EXPDAT.LOG |

*Figure 6.13: Specify data pump parameters for the subset process.*

35. On the next page keep everything default and press ‘Submit’. See Figure 6.14.


Generate Subset: Schedule

General

* Job Name: GENERATE_SUBSET_6

Job Description:

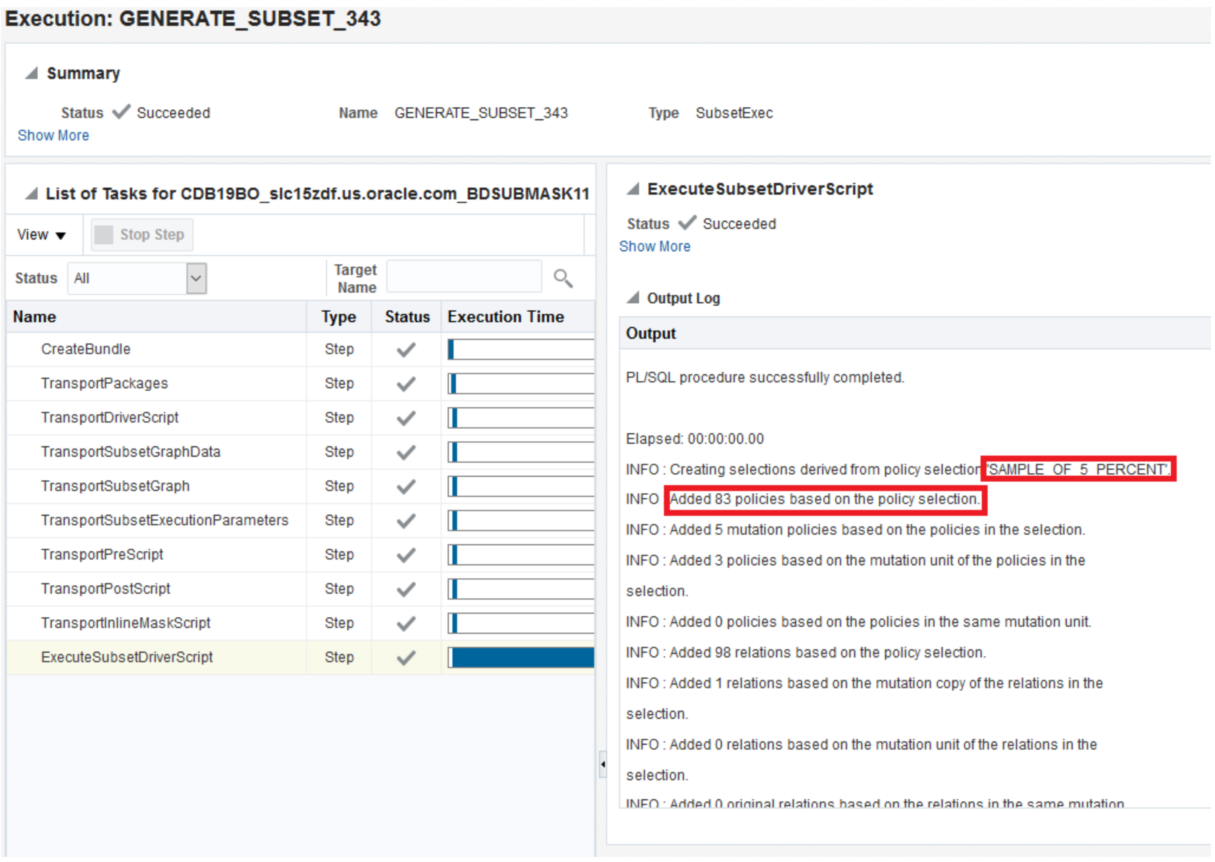
Schedule

Start: [✓ Immediately] Later (UTC+01:00) Amsterdam - Central European Time (CET)

Grace Period: ☐ Do not run if it cannot start within 1 hours of the scheduled start time

*Figure 6.14: Schedule and submit the subset process job.*

36. Once the subset generation job has completed, check the log in Enterprise Manager for any errors. Click on the “Succeeded” status under the “Most Recent Job Status” column to do so.
37. Especially for the “ExecuteSubsetDriverScript” step, check the log to see if there are any policies included in the subset and if the amount corresponds to the expected amount. See Figure 6.15


Execution: GENERATE_SUBSET_343

Summary

Status ✓ Succeeded    Name GENERATE_SUBSET_343    Type SubsetExec

Show More

List of Tasks for CDB19BO_slc15zdf.us.oracle.com_BDSUBMASK11

View ▾   Stop Step

Status All    Target Name

| Name | Type | Status | Execution Time |
|---|---|---|---|
| CreateBundle | Step | ✓ | |
| TransportPackages | Step | ✓ | |
| TransportDriverScript | Step | ✓ | |
| TransportSubsetGraphData | Step | ✓ | |
| TransportSubsetGraph | Step | ✓ | |
| TransportSubsetExecutionParameters | Step | ✓ | |
| TransportPreScript | Step | ✓ | |
| TransportPostScript | Step | ✓ | |
| TransportInlineMaskScript | Step | ✓ | |
| ExecuteSubsetDriverScript | Step | ✓ | |

ExecuteSubsetDriverScript

Status ✓ Succeeded

Show More

Output Log

Output

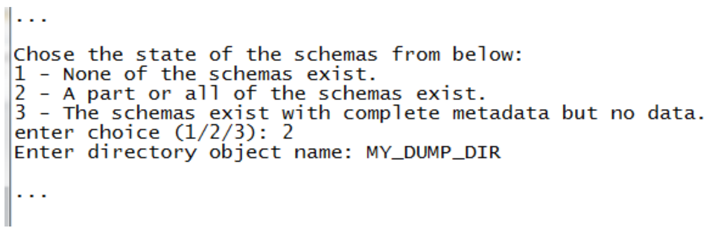
```
PL/SQL procedure successfully completed.

Elapsed: 00:00:00.00
INFO : Creating selections derived from policy selection 'SAMPLE_OF_5_PERCENT'.
INFO : Added 83 policies based on the policy selection.
INFO : Added 5 mutation policies based on the policies in the selection.
INFO : Added 3 policies based on the mutation unit of the policies in the
selection.
INFO : Added 0 policies based on the policies in the same mutation unit.
INFO : Added 98 relations based on the policy selection.
INFO : Added 1 relations based on the mutation copy of the relations in the
selection.
INFO : Added 0 relations based on the mutation unit of the relations in the
selection.
INFO : Added 0 original relations based on the relations in the same mutation
```

*Figure 6.15: Subset export job log, showing the policy selection*

## 6.4 Import the subset

For a more detailed description of the process please see: Import Export File and Run the Post Import Script File

1. The data pump export that was created in the previous step, will have to be imported into a smaller target database. Make sure that there is an ‘empty’ target database available. This could be based on the same backup that was used for the source database.
2. Copy the .dmp files, tdm_import.sql and tdm_post_script.sql files (that were created in step 6.3.34) from the source environment to a directory on the target environment.
3. Make sure that there is a directory object in the target database pointing to the directory on the host holding the export files. If this is not existing then create it.
4. Start SQL*Plus on the target host, connect as SYS and start the *tdm_import.sql* script.
5. When the script asks for input select option 2 (A part or all of the schemas exist).
6. Provide the directory object name from step 3 as input for the second question of the script. See Figure 6.16.



```
...

Chose the state of the schemas from below:
1 - None of the schemas exist.
2 - A part or all of the schemas exist.
3 - The schemas exist with complete metadata but no data.
enter choice (1/2/3): 2
Enter directory object name: MY_DUMP_DIR

...
```

*Figure 6.16: Enter data pump import parameters.*

7. The import may report errors about invalid objects. These can be ignored, they will be recompiled by the post import script. Other errors (e.g. Objects that cannot be created) should be taken more seriously and have to be further investigated.
8. Start SQL*Plus on the application server for the target database, connect as SYS and run the post import script: *tdm_post_script.sql*. Make sure to do this under a user-account that has access to the wallet wich contains the connection information for the target database.
9. After running this script all PL/SQL objects should be valid. Certain custom software might still be invalid. Make sure to validate this and resolve any possible errors.
10. A subset of an OHI DM environment can be obtained by performing an initial load of a subset OHI BO environment.

# 7 Appendix 2: Masking a custom database schema

This appendix provides a step-by-step guide to mask a custom database schema with data extracted from an OHI environment in the same way and with the same rules as the OHI environment is masked. This guide is not meant to mask any database with any dataset. Only data from an OHI environment in a custom development schema can be masked, with the limitation that no additional masking rules other than those provided and used to mask an OHI environment can be used and are supported.

## 7.1 Creating the masking rule set

It is assumed that the custom development schema is in the same database as the OHI database schema and that the custom development schema has the grants of the OZG_ROL_DIRECT role.

Each table and/or column that should be masked should be registered via the procedure sdm_driver_pck.add_custom_rule.

This procedure has the following interface:

```
procedure add_custom_rule
( pi_schema_owner     in sdm#custom_schema_masking.schema_owner%type
, pi_table_name       in sdm#custom_schema_masking.table_name%type
, pi_column_name      in sdm#custom_schema_masking.column_name%type
, pi_masking_rule     in sdm#custom_schema_masking.masking_rule%type
, pi_col_gender       in sdm#custom_schema_masking.col_gender%type default null
, pi_col_first_name   in sdm#custom_schema_masking.col_first_name%type default null
, pi_col_last_name    in sdm#custom_schema_masking.col_last_name%type default null
, pi_col_country      in sdm#custom_schema_masking.col_country%type default null
, pi_col_street       in sdm#custom_schema_masking.col_street%type default null
, pi_col_city         in sdm#custom_schema_masking.col_city%type default null
, pi_col_postal_code  in sdm#custom_schema_masking.col_postal_code%type default null
, pi_col_house_number in sdm#custom_schema_masking.col_house_number%type default null
);
```

The parameters

- **PI_SCHEMA_OWNER**: The custom development database schema that holds the table that is to be masked. It is possible to register tables from multiple database schemas, but each masking run will only process tables from one schema and run from that schema.
- **PI_TABLE_NAME**: The tablename of the table to be masked or truncated
- **PI_COLUMN_NAME**: The column name which should be masked; must be left empty when the table should be truncated
- **PI_MASKING_RULE**: The masking rule(*) that should be applied to the column or table
- **PI_COL_GENDER**: Column name (or expression) of the column containing the gender value
- **PI_COL_FIRST_NAME**: Column name (or expression) of the column containing the first name value
- **PI_COL_LAST_NAME**: Column name (or expression) of the column containing the last name value
- **PI_COL_COUNTRY**: Column name (or expression) of the column containing the country value
- **PI_COL_STREET**: Column name (or expression) of the column containing the street value
- **PI_COL_CITY**: Column name (or expression) of the column containing the city value
- **PI_COL_POSTAL_CODE**: Column name (or expression) of the column containing the postal code value

- **PI_COL_HOUSE_NUMBER**: Columnname (or expression) of the column containing the housenumber value

(*) Available masking rules for the parameter PI_MASKING_RULE are:

- **TRUNCATE**: truncate the table, in other words all rows are removed leaving an empty table
- **NULL_VALUE**: the column value is removed
- **TRANSLATE**: replaces the content with dummy data depending on the datatype of the column.
- **TRANSLATE_UNIQUE**: replaces the content of a varchar column with dummy characters, but will make sure the content will keep its uniqueness
- **DATE**: adjust the date to the 1st, 11th or 21st of the month, can be used e.g. for a date of birth
- **REL_NR**: replaces the relation number with a new calculated relation number
- **BSN**: replaces the relation number with a new calculated social security number
- **PHONE**_1: masks a phone number
- **PHONE**_2: masks a phone number with a different pattern
- **FAX**: masks a fax number
- **BANK**: replaces the bank account number with a new calculated bank account number (IBAN)
- **FIRSTNAME**: replaces the first name of a person with a new first name
- **LASTNAME**: replaces the last name of a person with a new last name
- **FIRSTNAME_LETTER**: replaces the first letter of the fist name of a person with a new first letter based on the new first name
- **EMAIL**: replaces the email of a person with a new email based on the new first and last name of that person
- **STREET**: replaces the street part of an address with a fake street
- **CITY**: replaces the city part of an address with a fake city
- **COUNTRY**: replaces the country part of an address with a fixed one if this is not the Dutch country code
- **HOUSENUMBER**: replaces the house number of an address
- **POSTAL_CODE**: replaces the postal code part of an address with a fake postal code
- **POSTAL_CODE_LETTER**: replaces the character part of a Dutch postal code with a fake character part
- **POSTAL_CODE_NUMBER**: replaces the numerical part of a Dutch postal code with a fake numerical part

## 7.2 The use of the PI_COL_* parameters

Masking is based on the data in the given row. Sometimes the name of the column itself is enough, sometimes additional data is required.
For instance a gender is needed to differentiate between male and female first names.
The table below shows which additional columns are needed for each masking rule. If the rule is not mentioned in this table no additional columns are required.

Some of the required column parameters may seem unnecessary, e.g. a pi_col_postal_code for the column with/for the postal code  but these are required for the correct implementation.

| Rule | Additional Parameters |
|---|---|
| **FIRSTNAME** | PI_COL_GENDER |
| **FIRSTNAME_LETTER** | PI_COL_GENDER, PI_COL_FIRST_NAME |
| **EMAIL** | PI_COL_GENDER, PI_COL_FIRST_NAME, PI_COL_LAST_NAME |
| **STREET** | PI_COL_COUNTRY, PI_COL_POSTAL_CODE, PI_COL_HOUSENUMBER<br><br>If addresses outside of the Netherlands are also present in the table also the following are required:<br>PI_COL_STREET, PI_COL_CITY |
| **CITY** | PI_COL_COUNTRY, PI_COL_POSTAL_CODE, PI_COL_HOUSENUMBER<br><br>If addresses outside of the Netherlands are also present in the table also the following are required:<br>PI_COL_STREET, PI_COL_CITY |
| **COUNTRY** | PI_COL_COUNTRY, PI_COL_POSTAL_CODE, PI_COL_HOUSENUMBER<br><br>If addresses outside of the Netherlands are also present in the table also the following are required:<br>PI_COL_STREET, PI_COL_CITY |
| **HOUSENUMBER** | PI_COL_COUNTRY, PI_COL_POSTAL_CODE, PI_COL_HOUSENUMBER<br><br>If addresses outside of the Netherlands are also present in the table also the following are required:<br>PI_COL_STREET, PI_COL_CITY |
| **POSTAL_CODE** | PI_COL_COUNTRY, PI_COL_POSTAL_CODE, PI_COL_HOUSENUMBER<br><br>If addresses outside of the Netherlands are also present in the table also the following are required:<br>PI_COL_STREET, PI_COL_CITY |

| | |
|---|---|
| **POSTAL_CODE_LETTER** | PI_COL_COUNTRY, PI_COL_POSTAL_CODE, PI_COL_HOUSENUMBER<br><br>If addresses outside of the Netherlands are also present in the table also the following are required:<br>PI_COL_STREET, PI_COL_CITY |
| **POSTAL_CODE_NUMBER** | PI_COL_COUNTRY, PI_COL_POSTAL_CODE, PI_COL_HOUSENUMBER<br><br>If addresses outside of the Netherlands are also present in the table also the following are required:<br>PI_COL_STREET, PI_COL_CITY |

The PI_COL columns can contain the actual column name as well as an expression.

## 7.3 Execution

Masking of a custom development schema is done with the same interface as for OHI itself.
The procedure sdm_driver_pck.job, or sdm_driver_pck.mask, should be executed under the custom development schema owner. To be able to run the job, the owner of the custom schema must have the privileges ‘CREATE JOB’ and ‘EXECUTE ON DBMS_SCHEDULER’.

## 7.4 Example script

The following script is an example showing a possible setup.
NB. The procedure SDM_DRIVER_PCK.DEL_CUSTOM_SCHEMA will remove all rules for the given custom development schema.

This example assumes your script is the “master” that is maintained, and the rulres are re-created whenever there is a change in your custom schema that impacts the masking.

For this example the following custom development tables are used:

```
create table SVS_PERSONEN
( ID                      NUMBER not null,
  BSN                     VARCHAR2(20) not null,
  ACHTERNAAM              VARCHAR2(240),
  VOORNAAM                VARCHAR2(240),
  GEBDATUM                DATE not null,
  GESLACHT                VARCHAR2(1) not null,
  REKENINGNUMMER          VARCHAR2(40),
  DEC_CODE                VARCHAR2(30),
  EMAIL                   VARCHAR2(240),
  UZOVI                   VARCHAR2(36),
  MERK                    VARCHAR2(5) not null,
  ZV_PAKKET               VARCHAR2(5),
  AV_PAKKET               VARCHAR2(5),
  NOTITIES                VARCHAR2(4000)
);

create table SVS_ADRESSEN
( ID                      NUMBER not null,
  DATUM_INGANG            DATE not null,
  BSN                     VARCHAR2(20) not null,
  PC_LETTER               VARCHAR2(2),
  PC_NR                   NUMBER(4),
  HUISNR                  NUMBER(5),
  LAND                    VARCHAR2(2),
  BUITENLAND_STRAAT       VARCHAR2(35),
  BUITENLAND__POSTCODE    VARCHAR2(15),
  BUITENLAND__PLAATS      VARCHAR2(35)
);

create table SVS_TEKSTEN
( ID                      NUMBER not null,
  BSN                     VARCHAR2(20) not null,
  TEKST                   VARCHAR2(4000)
);

begin
  -- clear the rules for the schema SVS_OWNER
  sdm_driver_pck.del_custom_schema( pi_schema_owner => 'SVS_OWNER' );

  --table SVS_TEKSTEN should be made empty
  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_TEKSTEN'
  , pi_column_name      => null
  , pi_masking_rule     => 'TRUNCATE'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_PERSONEN'
  , pi_column_name      => 'BSN'
  , pi_masking_rule     => 'BSN'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_PERSONEN'
  , pi_column_name      => 'ACHTERNAAM'
  , pi_masking_rule     => 'LASTNAME'
  );

  -- Gender in OHI is stored as 1 or 2; example shows a conversion if
  -- the custom data gender column has another domain set for the GENDER
  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_PERSONEN'
  , pi_column_name      => 'VOORNAAM'
  , pi_masking_rule     => 'FIRSTNAME'
  , pi_col_gender       => 'DECODE(GESLACHT,''M'',1,''V'',2,GESLACHT)'

  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_PERSONEN'
  , pi_column_name      => 'GEBDATUM'
  , pi_masking_rule     => 'DATE'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_PERSONEN'
  , pi_column_name      => 'REKENINGNUMMER'
  , pi_masking_rule     => 'BANK'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_PERSONEN'
  , pi_column_name      => 'EMAIL'
  , pi_masking_rule     => 'EMAIL'
  , pi_col_gender       => 'GESLACHT'
  , pi_col_first_name   => 'VOORNAAM'
  , pi_col_last_name    => 'ACHTERNAAM'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_PERSONEN'
  , pi_column_name      => 'DEC_CODE'
  , pi_masking_rule     => 'TRANSLATE_UNIQUE'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_PERSONEN'
  , pi_column_name      => 'NOTITIES'
  , pi_masking_rule     => 'TRANSLATE'
  );

  -- NOTE: although the SVS_PERSONEN.BSN is masked by a rule above, all other BSN colums
  -- should be masked in the same way; there is no automatic detection of the same type of
  -- columns or foreign key relation dependencies
  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'BSN'
  , pi_masking_rule     => 'BSN'
  );
  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'PC_LETTER'
  , pi_masking_rule     => 'POSTAL_CODE_LETTER'
  , pi_col_country      => '''NL'''
  , pi_col_postal_code  => 'PC_NR||PC_LETTER'
  , pi_col_house_number => 'HUISNR'
  );

  -- NOTE: although a country code column is present in this table (column LAND)
  -- this example uses a literal value for the country code
  -- as this specific example we ‘know’ that it is a Dutch (NL) specific postal code
  -- NOTE: In this case the postal code is in the table separated into two columns.
  -- you can use a concatenation to make these two columns act as one for the masking rule
  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'PC_NR'
  , pi_masking_rule     => 'POSTAL_CODE_NUMBER'
  , pi_col_country      => '''NL'''
  , pi_col_postal_code  => 'PC_NR||PC_LETTER'
  , pi_col_house_number => 'HUISNR'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'BUITENLAND_STRAAT'
  , pi_masking_rule     => 'STREET'
  , pi_col_country      => 'LAND'
  , pi_col_street       => 'BUITENLAND_STRAAT'
  , pi_col_city         => 'BUITENLAND_PLAATS'
  , pi_col_postal_code  => 'BUITENLAND_POSTCODE'
  , pi_col_house_number => 'HUISNR'
  );
```

```
  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'BUITENLAND_POSTCODE'
  , pi_masking_rule     => 'POSTAL_CODE'
  , pi_col_country      => 'LAND'
  , pi_col_street       => 'BUITENLAND_STRAAT'
  , pi_col_city         => 'BUITENLAND_PLAATS'
  , pi_col_postal_code  => 'BUITENLAND_POSTCODE'
  , pi_col_house_number => 'HUISNR'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'BUITENLAND_PLAATS'
  , pi_masking_rule     => 'CITY'
  , pi_col_country      => 'LAND'
  , pi_col_street       => 'BUITENLAND_STRAAT'
  , pi_col_city         => 'BUITENLAND_PLAATS'
  , pi_col_postal_code  => 'BUITENLAND_POSTCODE'
  , pi_col_house_number => 'HUISNR'
  );

  -- NOTE: besides a literal or concatenation also simple sql functions like NVL are supported
  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'HUISNR'
  , pi_masking_rule     => 'HOUSENUMBER'
  , pi_col_country      => 'LAND'
  , pi_col_street       => 'BUITENLAND_STRAAT'
  , pi_col_city         => 'BUITENLAND_PLAATS'
  , pi_col_postal_code  => 'NVL(PC_NR||PC_LETTER,BUITENLAND_POSTCODE)'
  , pi_col_house_number => 'HUISNR'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'LAND'
  , pi_masking_rule     => 'COUNTRY'
  , pi_col_country      => 'LAND'
  , pi_col_street       => 'BUITENLAND_STRAAT'
  , pi_col_city         => 'BUITENLAND_PLAATS'
  , pi_col_postal_code  => 'NVL(PC_NR||PC_LETTER,BUITENLAND_POSTCODE)'
  , pi_col_house_number => 'HUISNR'
  );

  sdm_driver_pck.add_custom_rule
  ( pi_schema_owner     => 'SVS_OWNER'
  , pi_table_name       => 'SVS_ADRESSEN'
  , pi_column_name      => 'NOTITIES'
  , pi_masking_rule     => 'NULL_VALUE'
  );
  -- NOTE: Don't forget to commit the transactions
  commit;
end;
/
```