Oracle® Data Masking and Subsetting

User's Guide

Release 12.1.0.8

E64335-01

July 2015

Oracle Data Masking and Subsetting User's Guide, Release 12.1.0.8

E64335-01

Contents

Preface

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This document provides information about how to manage test data using the Oracle Data Masking and Subsetting features of the Enterprise Manager for Oracle Database Plug-in. This document is intended for database administrators, application designers, and programmers who are responsible for performing real-world testing of Oracle Database.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information about some of the topics discussed in this document, see the following documents in the Oracle Database Release 12.1 documentation set:

- *Oracle Database 2 Day DBA*
- *Oracle Database 2 Day + Performance Tuning Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database Concepts*
- *Oracle Database Performance Tuning Guide*

- *Oracle Database SQL Tuning Guide*
- *Oracle Database Testing Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release for Oracle Data Masking and Subsetting User's Guide

This preface contains:

- Versioning of Oracle Data Masking and Subsetting
- Changes in Oracle Data Masking and Subsetting Release 12.1

Versioning of Oracle Data Masking and Subsetting

The version of Oracle Data Masking and Subsetting is based on the version of the Oracle Database Plug-in.

Oracle Data Masking and Subsetting is a part of the Oracle Enterprise Manager's Oracle Database Plug-in. In general, multiple Oracle Database plug-in versions are released during the life cycle of Oracle Enterprise Manager version. For example, Oracle Database Plug-in versions 12.1.0.6 and 12.1.0.7 are released during the life cycle of Oracle Enterprise Manager version 12.1.0.4.

Changes in Oracle Data Masking and Subsetting Release 12.1

The following are changes in *Oracle Data Masking and Subsetting User's Guide* Release 12.1.0.8.

- Support for Secure Shell Key-based Host Authentication
- Support for Data Masking and Subsetting in Oracle Cloud

Support for Secure Shell Key-based Host Authentication

Oracle Data Masking and Subsetting supports Secure Shell (SSH) Key-based Host authentication to the target database host. For more information, see the *Oracle Enterprise Manager Cloud Control Security Guide*.

Support for Data Masking and Subsetting in Oracle Cloud

Oracle Data Masking and Subsetting provides the ability to model, mask, and subset the databases that are hosted in Oracle Cloud (DBaaS), in addition to the on-premise databases. In order to mask and subset data in Oracle Cloud (DBaaS), you must register the database in Oracle Cloud as a target for Oracle Enterprise Manager. For more information, see the *Oracle Enterprise Manager Cloud Control Administrator's Guide*.

1

Introduction to Oracle Data Masking and Subsetting

The Data Masking and Subsetting features of the Enterprise Manager for Oracle Database Plug-in help you to securely manage test data.

When performing real-world testing, there is the risk of exposing sensitive data to non-production users in a test environment. The test data management features of Oracle Database helps to minimize this risk by enabling you to perform data masking and data subsetting on the test data.

When production data is copied into a testing environment, there is the risk of breaching sensitive information to non-production users, such as application developers or external consultants. In order to perform real-world testing, these non-production users need to access some of the original data, but not all the data, especially when the information is deemed confidential.

Oracle Database offers test data management features that help reduce this risk by enabling you to:

- Store the list of applications, tables, and relationships between table columns using Application Data Modeling. See Chapter 2, "Application Data Modeling," for more information.
- Replace sensitive data from your production system with fictitious data that can be used during testing using data masking. See Chapter 3, "Data Masking," for more information.
- Replicate information that pertains only to a particular site using data subsetting. See Chapter 4, "Data Subsetting," for more information.

> **Note:** You must have the Oracle Data Masking and Subsetting Pack license to use these security features.

2

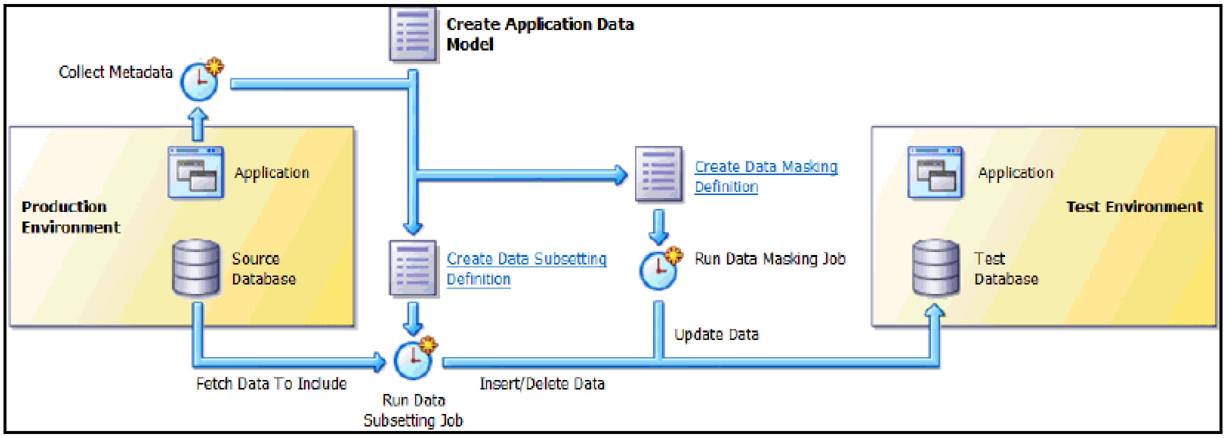
Application Data Modeling

Application Data Modeling discovers sensitive columns from Oracle database tables and corresponding referential relationships by running discovery jobs, matching on patterns and referring to Oracle application accelerators. The resulting data model is stored in the Enterprise Manager repository. Application Data Models are used by Oracle Database Security products including Oracle Data Masking and Subsetting. You must have the Oracle Data Masking and Subsetting Pack license to use these security features.

The ADM stores the list of applications, tables, and relationships between table columns that are either declared in the data dictionary, imported from application metadata, or user-specified. The ADM maintains sensitive data types and their associated columns, and is used by test data operations, such as data subsetting and data masking, to securely produce test data. Creating an ADM is a prerequisite for data masking and subsetting operations.

Figure 2–1 shows the Application Data Modeling workflow to create test data from a production environment.

Figure 2–1 Test Data Workflow



You can perform several tasks related to Application Data Modeling, including the following tasks discussed in this chapter:

- Creating an Application Data Model
- Managing Sensitive Column Types
- Associating a Database to an Application Data Model
- Importing and Exporting an Application Data Model

- Verifying or Upgrading a Source Database
- Using Self Update to Download the Latest Data Masking and Subsetting Templates
- Access Rights to Data Security Objects
- Granting Privileges on an Application Data Model

Note: The procedures in this chapter are applicable to Oracle Enterprise Manager 12.1 and higher Cloud Control only.

See Also:

- Chapter 3, "Data Masking," for information about data masking
- Chapter 4, "Data Subsetting," for information about data subsetting

Creating an Application Data Model

The following procedure enables you to:

- Initiate creation of an Application Data Model (ADM)
- View and edit application tables
- View referential relationships
- Manually add a referential relationship
- Discover sensitive columns
- Set the type for sensitive columns

Before proceeding, ensure that you have the following privileges:

- `EM_ALL_OPERATOR` for Enterprise Manager Cloud Control users

Note: The EM_ALL_OPERATOR privilege is not required, if you have the following privileges:

Target Privileges (applicable to all targets):

1. Connect to any viewable target
2. Execute Command Anywhere

Resource Privileges:

1. Job System
2. Named Credential
3. Oracle Data Masking and Subsetting resource privilege

- `SELECT_CATALOG_ROLE` for database users
- Select Any Dictionary privilege for database users

> **Note:** When you create an ADM, the PL/SQL metadata collection packages are automatically deployed on the target database. The Database user must have DBA privileges to auto-deploy the packages.

To create an Application Data Model:

1. From the **Enterprise** menu, select **Quality Management**, then select **Application Data Modeling**. The selection is also available from the **Security** menu on the Databases page (**Database Load Map**).

   As the diagram shows, the first step is to create an ADM.

2. Create an ADM:

   a. Click **Create**.

      A pop-up window requesting general properties information appears.

   b. Provide the required Name and Source Database.

      The Source Database is the source from which the metadata is to be extracted.

   c. Select an Application Suite:

      If you select **Custom Application Suite**:

      – By default, metadata collection is enabled for the ADM creation process.

      – If you uncheck “Create One Application For Each Schema,” you create a shell ADM and will need to edit the ADM later to add applications and tables. Also, no metadata collection job is submitted, but you can initiate metadata collection by submitting a verification job as described in "Verifying or Upgrading a Source Database" on page 2-9.

      If you select **Oracle Application Suite**:

      – **Oracle E-Business Suite**–You provide database credentials for APPS user (or equivalent) and submit a job to create the ADM.

      – **Oracle Fusion Applications**–You provide database credentials for FUSION user (or equivalent) and submit a job to create the ADM.

      Note the following points about metadata collections:

      – The metadata collection for the selected application suite populates the ADM with the applications and tables in the suite.

      – The ADM can collect metadata for one or more schemas. An ADM application typically represents a schema. Each schema you select becomes an ADM application, and the ADM becomes populated with the tables in the schema, particularly in the case of custom applications. Note, however, that multiple applications can also map to a single schema, as in the case of Fusion Applications. The actual mapping depends on the application metadata discovered by the metadata collection job.

   d. Click **Continue**.

      Assuming that you selected Custom Application Suite, a Schemas pop-up appears in which you select schemas to include from the Available list.

   e. Click **Continue**, provide the schedule parameters, then click **Submit** to submit the metadata collection job.

The ADM you created appears in the table on the Application Data Modeling page. The Most Recent Job Status table column indicates that the metadata collection job is running. The model is locked, and you cannot edit it during this period until the status indicates that the job is complete.

3. View and edit application tables:

   a. Select the model you created, then select **Edit**.

      The Applications and Tables subpage appears, displaying the applications found during metadata collection.

      To see the tables for an application, click the expand icon.

   b. To edit an application, select the application, then select **Add Table to Application** from the **Actions** menu.

      The Add Table to Application pop-up window appears.

   c. Click the **Table** search icon.

      The Search and Select pop-up appears, showing all of the tables from the selected schema that are not assigned to an application.

   d. Select an unassigned table, then click **OK**.

      The table name now appears in the Add Table to Application pop-up.

   e. After selecting a Table Type, click **OK**.

      The table now appears in the Applications and Tables view.

4. View referential relationships:

   a. Click the **Referential Relationships** tab.

      There are three types of referential relationships:

      – Dictionary-defined

        Upon opening this tab, this view shows the referential relationships that the metadata collection extracted, resulting from primary key and foreign key relationships. You can remove relationships from the ADM if desired.

      – Imported from template

        If there are application templates available from the vendor of the enterprise application, for example, Oracle Fusion Applications or Oracle E-Business Suite, then the ADM can be created from the application vendor-supplied template by using the Import action on the ADM home page.

      – User-defined

        See the step below about manually adding a referential relationship for more information.

   b. Open an application view by selecting it, then expand parent and dependent key relationships, or by select **Expand All** from the **View** menu to view all relationships.

5. Manually add a referential relationship:

   a. From the Referential Relationships tab, select **Add Referential Relationship** from the **Actions** menu.

      The Add Referential Relationship pop-up window appears.

   b. Select the requisite Parent Key and Dependent Key information.

c. In the Columns Name list, select a dependent key column to associate with a parent key column.

d. Click **OK** to add the referential relationship to the ADM.

   The new dependent column now appears in the referential relationships list.

6. Discover sensitive columns automatically or add them manually:

> **Note:** Oracle recommends that you gather statistics prior to submitting the sensitive column discovery job for more accurate results. For example, to determine if there empty tables in the schema before unchecking the Scan Empty Tables option on the discovery job page.

To automatically discover sensitive columns:

a. Click the **Sensitive Columns** tab, then select **Create Sensitive Column Discovery Job** from the **Actions** menu.

   The Parameters pop-up appears.

b. Select one or more applications and one or more sensitive column types.

   Each type you select is processed for each application to search for columns that match the type.

c. Click **Continue**.

   The schedule pop-up window appears.

d. Provide the required information, schedule the job, then click **Submit**.

   The Sensitive Columns subpage reappears.

e. Click **Save and Return** to return to the Application Data Modeling home page.

f. When the Most Recent Job Status column indicates that the job is Successful, select the ADM, then click **Edit**.

g. Select the **Sensitive Columns** tab, then click **Discovery Results** to view the job results.

h. To set the sensitive status of any column, select the row for the column you want to define, open the **Set Status** menu, then select either **Sensitive** or **Not Sensitive**.

i. Click **OK** to save and return to the Sensitive Columns tab.

   The sensitive columns you defined in the previous step now appear in the list.

j. Click **Save and Return** to return to the Application Data Modeling page.

To manually add sensitive columns:

a. From the Application Data Modeling page, select an ADM, then click **Edit**.

b. Select the **Sensitive Columns** tab, then click **Add**.

   The Add Sensitive Column pop-up appears.

c. Provide the required information and an optional Sensitive Column Type, then click **OK**.

   The sensitive column now appears in the table for the Sensitive Columns tab.

7. Change the type for sensitive columns:

   a. Click the **Sensitive Columns** tab.

      This view shows the sensitive columns that have already been identified.

   b. Select the sensitive column for which you want to change the type.

   c. Select **Set Sensitive Column Type** from the **Actions** menu.

      The Set Sensitive Column Type pop-up window appears.

   d. Select the new type and click **OK**.

Managing Sensitive Column Types

After you have successfully created an ADM, the next task is to create either a new sensitive column type or one based on an existing type.

To create a sensitive column type:

1. From the **Actions** menu on the Application Data Modeling page, select **Sensitive Column Types**.

   The Sensitive Column Types page appears.

2. Click **Create**.

   The Create Sensitive Column Type pop-up appears.

3. Specify a required name and regular expressions for the Column Name, Column Comment, and Column Data search patterns.

   - The Or Search Type means that any of the patterns can match for a candidate sensitive column.
   - The And Search Type means that all of the patterns must match for a candidate sensitive column.

   If you do not provide expressions for any of these parameters, the system does not search for the entity.

4. Click **OK**.

   The sensitive column appears in the table in the Sensitive Column Types page.

To create a sensitive column type based on an existing type:

1. From the **Actions** menu on the Application Data Modeling page, select **Sensitive Column Types**.

   The Sensitive Column Types page appears.

2. Select either a sensitive column type you have already defined, or select one from the out-of-box types that the product provides.

3. Click **Create Like**.

   The Create Sensitive Column Type pop-up appears.

4. Specify a required name and alter the existing expressions for the Column Name, Column Comment, and Column Data search patterns to suit your needs.

5. Click **OK**.

   The sensitive column appears in the table in the Sensitive Column Types page.

Associating a Database to an Application Data Model

After you have created an Application Data Model (ADM), you can select additional databases to be associated databases of an ADM, as explained in the following procedure. See "Creating an Application Data Model" on page 2-2 for instructions on creating an ADM.

To associate a database to an ADM:

1. From the Application Data Modeling page, select an ADM, then select **Associated Databases** from the **Actions** menu.

   This dialog lists all of the databases associated with this ADM and the schemas assigned to each application per database. You can add more databases that give you a choice of data sources when subsetting and databases to mask during masking.

2. Click **Add**, then select a database from the pop-up.

   The selected database now appears in the Database section of the Associated Databases dialog.

3. To change a schema, select the associated database on the left, select the application on the right for which the schema is to be changed, then click **Select Schema**.

4. Select the missing schema from the list in the pop-up, then click **Select**.

> **Note:** You also can associate an ADM to a target remotely or through a script using the EMCLI verb `associate_target_to_adm`. See the *Oracle Enterprise Manager Command Line Interface* manual for details.

Importing and Exporting an Application Data Model

You can share Application Data Models (ADM) with other Enterprise Manager environments that use a different repository by exporting an ADM, which can subsequently be imported into the new repository.

An exported ADM is by definition in the XML file format required for import. You can edit an exported ADM XML file prior to import. When exporting an ADM for subsequent import, it is best to have one that uses most or all of the features—applications, tables, table types, referential relationships, sensitive columns. This way, if you are going to edit the exported file prior to import, it is clear which XML tags are required and where they belong in the file.

- Importing an ADM
- Exporting an ADM

> **Note:** You also can export and import an ADM remotely or through a script using the EMCLI verbs `export_adm` and `import_adm`, respectively. See the *Oracle Enterprise Manager Command Line Interface* manual for details.

Importing an ADM

There are two methods of import:

- Import an ADM XML file from the desktop

- Import an ADM XML file from the Software Library

To import an ADM XML file from your desktop:

1. From the **Actions** menu, select **Import**, then select **File from Desktop**.
2. In the pop-up that appears, specify a name for the ADM, the source database you want to assign to the ADM, and location on your desktop from which you want to import the ADM.
3. Click **OK**.

   The ADM now appears on the Application Data Modeling page.

To import an ADM XML file from the Software Library:

1. From the **Actions** menu, select **Import**, then select **File from Software Library**.
2. In the Export File from Software Library pop-up that appears, select the desired ADM XML file on the left, then specify a name and the source database you want to assign to the ADM on the right.
3. Click **Import**.

   The ADM now appears on the Application Data Modeling page.

After importing an ADM, you may want to discover sensitive columns or run a verification job. In the process of performing these tasks, the PL/SQL metadata collection packages are automatically deployed on the target database. The Database user must have DBA privileges to auto-deploy the packages.

Exporting an ADM

A user with Operator or Designer privileges can export an ADM. There are three methods of export:

- Export a selected ADM to the desktop
- Export an ADM from the Software Library
- Export an ADM to a TSDP Catalog

To export an ADM as an XML file to your desktop:

1. From the Application Data Models page, select the ADM you want to export.
2. From the **Actions** menu, select **Export**, then select **Selected Application Data Model**.
3. In the File Download pop-up that appears, click **Save**.
4. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The system converts the ADM into an XML file that now appears at the specified location on your desktop.

To export an ADM from the Software Library:

1. From the **Actions** menu, select **Export**, then select **File from Software Library**.
2. In the Export File from Software Library pop-up that appears, select the desired ADM and click **Export**.
3. In the File Download pop-up that appears, click **Save**.
4. In the Save As pop-up that appears, navigate to a file location and click **Save**.

The system converts the ADM into an XML file that now appears at the specified location on your desktop.

To export an ADM to a Transparent Sensitive Data Protection (TSDP) Catalog:

1. From the Application Data Modeling page, select the ADM you want to export.
2. From the **Actions** menu, select **Export**, then select **Export to TSDP Catalog.**
3. The Application Data Modeling page displays a table of associated databases. Select a database and click the **Export Sensitive Data** button.
4. In the Export Sensitive Data pop-up that appears, provide credentials for the selected database and click **OK**.

   A message appears on the Application Data Modeling page confirming that the sensitive data was copied to the database.

For detailed information on TSDP, see the *Oracle Database Security Guide*.

Verifying or Upgrading a Source Database

After you have created an Application Data Model (ADM), the Source Database Status column can indicate Valid, Invalid, Needs Verification, or Needs Upgrade.

- **Invalid status**–Verify the source database to update the referential relationships in the application data model with those found in the data dictionary, and to also determine if each item in the application data model has a corresponding object in the database.
- **Needs Verification status**–You have imported an Oracle supplied template and you must verify the ADM before you can use it. This is to ensure that necessary referential relationships from data dictionary are pulled into the ADM.
- **Needs Upgrade status**–You have imported a pre-12*c* masking definition, so you now need to upgrade the ADM.

To verify a source database:

1. Select the ADM to be verified, indicated with an Invalid status.
2. From the **Actions** menu, select **Verify**.
3. Select the source database with the Invalid status, then click **Create Verification Job**.
4. Specify job parameters in the Create Verification Job pop-up, then click **Submit**.
5. After the job completes successfully, click the source database and note the object problems listed.
6. Fix the object problems, rerun the Verification Job, then check that the Source Database Status is now Valid.

> **Note:** You also can submit a verification job remotely or through a script using the EMCLI verb `verify_adm`. See the *Oracle Enterprise Manager Command Line Interface* manual for details.

To upgrade an ADM:

1. Select the ADM to be upgraded, indicated with a Needs Upgrade status.
2. From the **Actions** menu, select **Verify and Upgrade**.

3. Specify job parameters in the Create Upgrade Job pop-up, then click **Submit**.
4. After the job completes successfully, check that the Source Database Status column now indicates Valid. If the column indicates Invalid, see the previous procedure.

Using Self Update to Download the Latest Data Masking and Subsetting Templates

Use the Self Update feature to get the latest data masking and subsetting templates available from Oracle, out of band of the next major release cycle. With the auto-download feature enabled, new templates appear in the Software Library as they become available. Otherwise, you can access them manually as follows:

1. From the **Setup** menu, select **Extensibility**, then select **Self Update**.
2. On the Self Update page, scroll down and select Data Masking and Subsetting templates.
3. In the available updates table, select the templates you want to download and select **Download** from the **Actions** menu.

   This action downloads the templates to the Software Library from where you can import them into your Data Masking and Subsetting environment or save them locally for editing.
4. To save a downloaded template:
   a. Navigate to the appropriate home page (ADM, masking, or subset).
   b. From the **Actions** menu, select **Export from the Software Library**.
   c. Select a template in the pop-up window and click **Save**.
   d. Specify a location where to save the XML file.

   The template file is available for editing prior to being imported into Application Data Modeling.
5. To import a downloaded template:
   a. Navigate to the appropriate home page (ADM, masking, or subset).
   b. From the **Actions** menu, select **Import from the Software Library**.
   c. Select a template in the pop-up window and specify appropriate values for the input parameters.
   d. Click **Import**. Template definitions are imported into the respective tables.

Reapply Templates if Upgrading to Database Plug-in 12.1.0.5 from Release 12.1.0.3 or 12.1.0.4

If you previously applied Data Masking and Subsetting templates and are upgrading from release 12.1.0.3 or 12.1.0.4, you have to remove the templates and then reapply them; otherwise, the templates will not be visible when importing or exporting from the Software Library.

1. Go to the Self Update page.
2. Select the Data Masking and Subsetting templates and remove them.
3. Return to the Self Update page.
4. Select the Data Masking and Subsetting templates and check for updates.

5. In the available updates table, select the templates you want and then download and apply them using the respective actions from the **Actions** menu.

   The templates should now be visible in the Software Library.

Access Rights to Data Security Objects

By default, Enterprise Manager Administrators can access the primary data security object pages:

- Application Data Modeling
- Data Subsetting Definitions
- ns
- Data Masking Formats

This is by virtue of having the TDM_ACCESS privilege, which is included in the PUBLIC role. The Super Administrator can revoke this privilege for designated administrators, thereby restricting access to the these pages. Without the privilege, the respective menu items do not appear in the Cloud Control console.

Additionally, Enterprise Manager provides a privilege access model that enables Super Administrators and administrators to limit access to data security objects (ADMs, ns, and data subsetting definitions) to authorized users only. The model involves the ability to grant Operator or Designer privileges to selected users.

Operator Privileges

Those granted Operator privileges can perform data masking and subsetting operations. Privileges can be granted on data security objects; that is, on ADMs, data subsetting definitions, and ns. Operator privileges do not include the ability to edit and delete these objects.

- ADM–a user (other than Super Administrator) with ADM Operator privileges can view an ADM and export it, but cannot edit and delete it, nor view its properties. To enforce this, the Edit and Delete icons, and the Properties menu are disabled. Additionally, the Sync option on the Create Verification Job page is disabled.
- Data subsetting definition–a user (other than Super DSD Administrator) with Operator privileges can view but not edit and delete a subset definition. To enforce this, the Edit and Delete icons are disabled.

  A user with Data Subsetting Definition Operator privileges can do any other operation except edit and delete the data subset definition and has the following rights:

  – View the data subsetting definition.

  – Create a data subset to export files.

  – Create a data subset on a database.

  – Save the subset script.
- n–a user with n Operator privileges can do any other operation except edit and delete the n and has the following rights:

  – View the n.

  – Generate a data masking script.

  – Schedule a data masking job.

- Export a n.

Designer Privileges

Those granted Designer privileges can enhance, modify, and manage data security objects. These users can also grant and revoke Operator and Designer privileges to others. Designer privileges imply the corresponding Operator privileges on a data security object.

- ADM–a user with Designer privileges can perform all operations on an ADM including delete.
- Data subsetting definition–a user with Designer privileges can perform all operations on a subset definition including delete.
- n–a user with Designer privileges can perform all operations on a masking definition including delete.

Granting Privileges on an Application Data Model

You can grant privileges on an Application Data Model that you create so that others can have access. To do so, you must be an Enterprise Manager Administrator with at least Designer privileges on the ADM.

1. From the **Enterprise** menu, select **Quality Management**, then select **Application Data Modeling**. The selection is also available from the **Security** menu on the Databases page (**Database Load Map**).
2. Select the ADM to which you want to grant privileges.
3. From the **Actions** menu, select **Grant**, then select as follows:
   - **Operator**–to grant Operator privileges on the ADM to selected roles or administrators, which means the grantees can view and copy but not edit and delete the definition.
   - **Designer**–to grant Designer privileges on the ADM to selected roles or administrators, which means the grantees can view, edit, and delete the definition.
4. In the dialog that opens, select the type (administrator or role, or both). Search by name, if desired. Make your selections and click **Select**.

   Those selected now have privileges on the ADM.
5. Use the **Revoke** action if you want to deny privileges previously granted.

3

Data Masking

This chapter provides conceptual information about the components that comprise Oracle Data Masking, and procedural information about performing the task sequence, such as creating masking formats and masking definitions. Data masking presupposes that you have created an Application Data Model (ADM) with defined sensitive columns.

Topics discussed in this chapter include:

The procedures in this chapter are applicable to Oracle Enterprise Manager 12.1 and higher Cloud Control only. You must have the Oracle Data Masking and Subsetting Pack license to use data masking features.

Notes: Performing masking on an 11.2.0.3 database that uses Database Plug-in 12.1.0.3 and higher requires that database patch # 16922826 be applied for masking to run successfully.

The Mask in Export feature (also known as At Source masking) works with Oracle Database 11.1 and higher.

Overview of Oracle Data Masking

Enterprises run the risk of breaching sensitive information when copying production data into non-production environments for the purposes of application development, testing, or data analysis. Oracle Data Masking helps reduce this risk by irreversibly replacing the original sensitive data with fictitious data so that production data can be shared safely with non-production users. Accessible through Oracle Enterprise Manager, Oracle Data Masking provides end-to-end secure automation for provisioning test databases from production in compliance with regulations.

Data Masking Concepts

Data masking (also known as data scrambling and data anonymization) is the process of replacing sensitive information copied from production databases to test non-production databases with realistic, but scrubbed, data based on masking rules. Data masking is ideal for virtually any situation when confidential or regulated data needs to be shared with non-production users. These users may include internal users such as application developers, or external business partners such as offshore testing companies, suppliers and customers. These non-production users need to access some of the original data, but do not need to see every column of every table, especially when the information is protected by government regulations.

Data masking enables organizations to generate realistic and fully functional data with similar characteristics as the original data to replace sensitive or confidential information. This contrasts with encryption or Virtual Private Database, which simply hides data, and the original data can be retrieved with the appropriate access or key. With data masking, the original sensitive data cannot be retrieved or accessed.

Names, addresses, phone numbers, and credit card details are examples of data that require protection of the information content from inappropriate visibility. Live production database environments contain valuable and confidential data—access to this information is tightly controlled. However, each production system usually has replicated development copies, and the controls on such test environments are less stringent. This greatly increases the risks that the data might be used inappropriately. Data masking can modify sensitive database records so that they remain usable, but do not contain confidential or personally identifiable information. Yet, the masked test data resembles the original in appearance to ensure the integrity of the application.

Security and Regulatory Compliance

Masked data is a sensible precaution from a business security standpoint, because masked test information can help prevent accidental data escapes. In many cases, masked data is a legal obligation. The Oracle Data Masking and Subsetting Pack can help organizations fulfill legal obligations and comply with global regulatory requirements, such as Sarbanes-Oxley, the California Database Security Breach Notification Act (CA Senate Bill 1386), and the European Union Data Protection Directive.

The legal requirements vary from country to country, but most countries now have regulations of some form to protect the confidentiality and integrity of personal consumer information. For example, in the United States, The Right to Financial Privacy Act of 1978 creates statutory Fourth Amendment protection for financial records, and a host of individual state laws require this. Similarly, the U.S. Health Insurance Portability and Accountability Act (HIPAA) created protection of personal medical information.

Roles of Data Masking Users

The following types of users participate in the data masking process for a typical enterprise:

- Application database administrator or application developer

This user is knowledgeable about the application and database objects. This user may add additional custom database objects or extensions to packaged applications, such as the Oracle E-Business suite.

- Information security administrator

  This user defines information security policies, enforces security best practices, and also recommends the data to be hidden and protected.

Related Oracle Security Offerings

Besides data masking, Oracle offers the following security products:

- Virtual Private Database or Oracle Label Security–Hides rows and data depending on user access grants.
- Transparent Data Encryption–Hides information stored on disk using encryption. Clients see unencrypted information.
- `DBMS_CRYPTO`–Provides server packages that enable you to encrypt user data.
- Database Vault–Provides greater access controls on data.

Agent Compatibility for Data Masking

Data masking supports Oracle Database 9*i* and newer releases. If you have a version prior to 11.1, you can use it by implementing the following workaround.

Replace the following file...

```
AGENT_HOME/sysman/admin/scripts/db/reorg/reorganize.pl
```

... with this file:

```
OMS_HOME/sysman/admin/scripts/db/reorg/reorganize.pl
```

Supported Data Types

The list of supported data types varies by release.

- Grid Control 10*g* Release 5 (10.2.0.5), Database 11*g* Release 2 (11.2), and Cloud Control 12*c* Release 1 (12.1.0.1) and Release 2 (12.1.0.2)

  – Numeric Types

    The following Numeric Types can use Array List, Delete, Fixed Number, Null Value, Post Processing Function, Preserve Original Data, Random Decimal Numbers, Random Numbers, Shuffle, SQL Expression, Substitute, Table Column, Truncate, Encrypt, and User Defined Function masking formats:

    * `NUMBER`

    * `FLOAT`

    * `RAW`

    * `BINARY_FLOAT`

    * `BINARY_DOUBLE`

  – String Types

    The following String Types can use Array List, Delete, Fixed Number, Fixed String, Null Value, Post Processing Function, Preserve Original Data, Random Decimal Numbers, Random Digits, Random Numbers, Random Strings, Shuffle, SQL Expression, Substitute, Substring, Table Column, Truncate, Encrypt, and User Defined Function masking formats:

  - `CHAR`
  - `NCHAR`
  - `VARCHAR2`
  - `NVARCHAR2`

- Date Types

  The following Date Types can use Array List, Delete, Null Value, Post Processing Function, Preserve Original Data, Random Dates, Shuffle, SQL Expression, Substitute, Table Column, Truncate, Encrypt, and User Defined Function masking formats:

  - `DATE`
  - `TIMESTAMP`

- Grid Control 11*g* Release 1 (11.1) and Cloud Control 12*c* Release 1 (12.1.0.1) and Release 2 (12.1.0.2)

  - Large Object (LOB) Data Types

    The following Data Types can use Fixed Number, Fixed String, Null Value, and SQL Expression masking formats:

    - BLOB
    - CLOB
    - NCLOB

Format Libraries and Masking Definitions

To mask data, the Oracle Data Masking and Subsetting Pack provides two main features:

- Data masking format library

  The format library contains a collection of ready-to-use masking formats. The library consists of format routines that you can use for masking. A masking format can either be one that you create, or one from the list of Oracle-supplied default masking formats.

  As a matter of best practice, organizations should create masking formats for all commonly regulated information so that the formats can be applied to the sensitive data regardless of which database the sensitive data resides in. This ensures that all sensitive data is consistently masked across the entire organization.

- Data masking definitions

  A masking definition defines a data masking operation to be implemented on one or more tables in a database. Masking definitions associate table columns with formats to use for masking the data. They also maintain the relationship between columns that are not formally declared in the database using related columns.

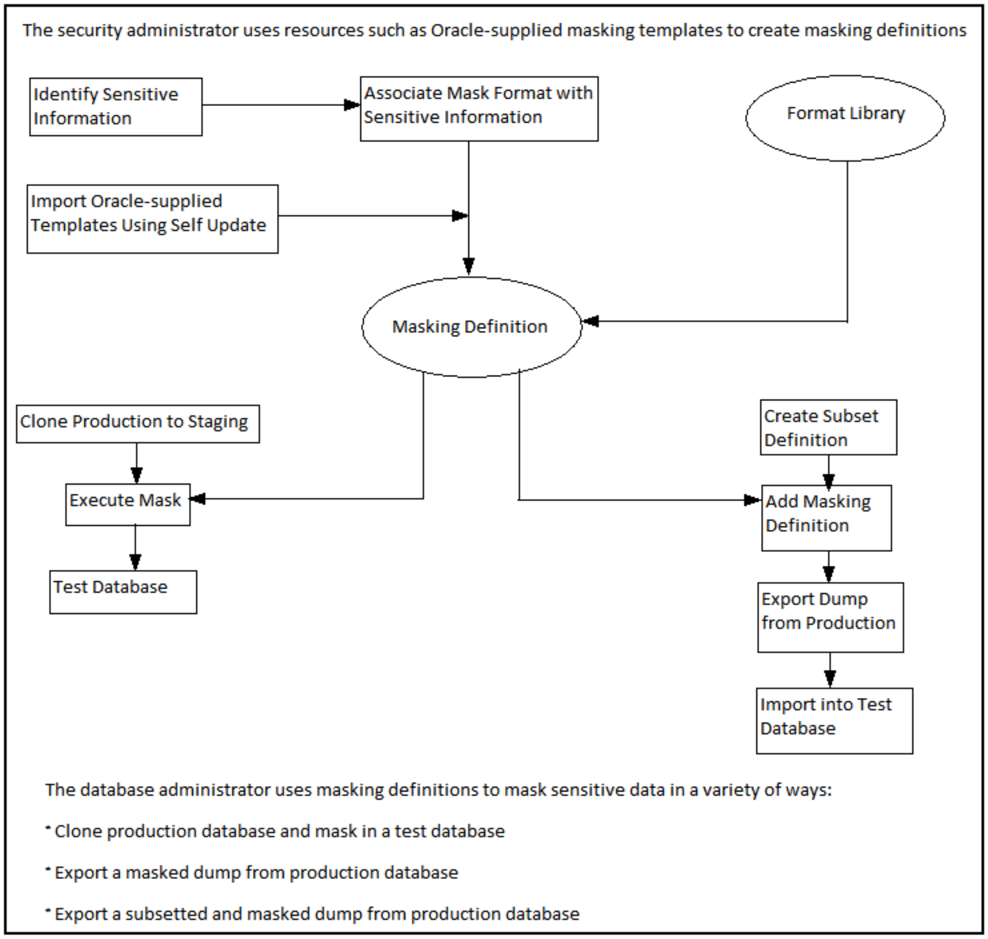
  You can create a new masking definition or use an existing definition for a masking operation. To create a masking definition, you specify the column of the table for which the data should be masked and the format of masked data. If the columns being masked are involved in unique, primary key, or foreign key constraints, data masking generates the values so that the constraints are not violated. Masking ensures uniqueness per character using decimal arithmetic. For example, a 5-character string generates a maximum of only 99999 unique values. Similarly, a 1-character string generates a maximum of only 9 unique values.

You would typically export masking definitions to files and import them on other systems. This is important when the test and production sites reside on different Oracle Management Systems or on entirely different sites.

Recommended Data Masking Workflow

Figure 3–1 shows that the production database is cloned to a staging region and then masked there. During the masking process, the staging and test areas are tightly controlled like a production site.

Figure 3–1 Data Masking Workflow


The security administrator uses resources such as Oracle-supplied masking templates to create masking definitions
Identify Sensitive Information
Associate Mask Format with Sensitive Information
Format Library
Import Oracle-supplied Templates Using Self Update
Masking Definition
Clone Production to Staging
Execute Mask
Test Database
Create Subset Definition
Add Masking Definition
Export Dump from Production
Import into Test Database
The database administrator uses masking definitions to mask sensitive data in a variety of ways:
• Clone production database and mask in a test database
• Export a masked dump from production database
• Export a subsetted and masked dump from production database

Data masking is an iterative and evolving process handled by the security administrator and implemented by the database administrator. When you first configure data masking, try out the masking definition on a test system, then add a greater number of columns to the masking definition and test it to make sure it functions correctly and does not break any application constraints. During this process, you should exercise care when removing all imbedded references to the real data while maintaining referential integrity.

After data masking is configured to your satisfaction, you can use the existing definition to repeatedly mask after cloning. The masking definition, however, would need to evolve as new schema changes require new data and columns to be masked.

After the masking process is complete, you can distribute the database for wide availability. If you need to ship the database to another third-party site, you are required to use the Data Pump Export utility, and then ship the dump file to the remote site. However, if you are retaining the masked data in-house, see "Data Masking Task Sequence" on page 3-6.

You also can apply data masking definitions while creating a data subsetting definition. See "About Integrated Subset and Mask" on page 4-13 for more information.

Data Masking Task Sequence

The task sequence in this section demonstrates the data masking workflow and refers you to additional information about some of the tasks in the sequence. Before reviewing this sequence, note that there are two options for completing this process:

- Exporting/importing to another database

  You can clone the production database to a staging area, mask it, then export/ import it to another database before delivering it to in-house testers or external customers. This is the most secure approach.

- Making the staging area the new test region

  You can clone the production database to a mask staging area, then make the staging area the new test region. In this case, you should not grant testers SYSDBA access or access to the database files. Doing so would compromise security. The masked database contains the original data in unused blocks and in the free list. You can only purge this information by exporting/importing the data to another database.

The following basic steps guide you through the data masking process, with references to other sections for supporting information.

1. Review the application database and identify the sources of sensitive information.
2. Define data masking formats for the sensitive data. The formats may be simple or complex depending on the information security needs of the organization.

   For more information, see "Creating New Data Masking Formats" on page 3-7 and "Using Oracle-supplied Predefined Data Masking Formats" on page 3-9.
3. Create a data masking definition to associate table columns to these data masking formats. Data masking determines the database foreign key relationships and adds foreign key columns to the mask.

   For more information, see "Masking with an Application Data Model and Workloads" on page 3-17.
4. Save the data masking definition and generate the masking script.
5. Verify if the masked data meets the information security requirements. Otherwise, refine the masking definition, restore the altered tables, and reapply the masking definition until the optimal set of masking definitions has been identified.
6. Clone the production database to a staging area, selecting the masking definition to be used after cloning. Note that you can clone using Enterprise Manager, which enables you to add masking to the Enterprise Manager clone workflow. However,

if you clone outside of Enterprise Manager, you must initiate masking from Enterprise Manager after cloning is complete. The cloned database should be controlled with the same privileges as the production system, because it still contains sensitive production data.

After cloning, be sure to change the passwords as well as update or disable any database links, streams, or references to external data sources. Back up the cloned database, or minimally the tables that contain masked data. This can help you restore the original data if the masking definition needs to be refined further.

For more information, see "Cloning the Production Database" on page 3-28.

7. After masking, test all of your applications, reports, and business processes to ensure they are functional. If everything is working, you can export the masking definition to keep it as a back-up.

8. After masking the staging site, make sure to drop any tables named MGMT_DM_TT before cloning to a test region. These temporary tables contain a mapping between the original sensitive column value and the mask values, and are therefore sensitive in nature.

   During masking, Enterprise Manager automatically drops these temporary tables for you with the default "Drop temporary tables created during masking" option. However, you can preserve these temporary tables by deselecting this option. In this case, you are responsible for deleting the temporary tables before cloning to the test region.

9. After masking is complete, ensure that all tables loaded for use by the substitute column format or table column format are going to be dropped. These tables contain the mask values that table column or substitute formats will use. It is recommended that you purge this information for security reasons.

   For more information, see "Deterministic Masking Using the Substitute Format" on page 3-16.

10. Clone the database to a test region, or use it as the new test region. When cloning the database to an external or unsecured site, you should use Export or Import. Only supply the data in the database, rather than the database files themselves.

11. As part of cloning production for testing, provide the data masking definition to the application database administrator to use in masking the database.

Defining Data Masking Formats

A data masking definition requires one or more data masking formats for any columns included in the masking definition. When adding columns to a masking definition, you can either create masking formats manually or import them from the format library. It is often more efficient to work with masking formats from the format library.

Creating New Data Masking Formats

This section describes how to create new data masking formats using Enterprise Manager.

To create a data masking format in the format library:

1. From the **Enterprise** menu, select **Quality Management**, then select **Data Masking Formats**. The selection is also available from the **Data Masking and Subsetting** submenu of the **Security** menu on the all databases home page or on a particular database’s home page.

The Format Library appears with predefined formats that Oracle Enterprise Manager provides.

2. Click **Create**.

   The Create Format page appears, where you can define a masking format.

3. Provide a required name for the new format, select a format entry type from the **Add** list, then click **Go**.

   A page appears that enables you to provide input for the format entry you have selected. For instance, if you select Array List, the subsequent page enables you to enter a list of values, such as New York, New Jersey, and New Hampshire.

4. Continue adding additional format entries as needed.

5. When done, provide an optional user-defined or post-processing function (see "Providing User-defined and Post-processing Functions" on page 3-8), then click **OK** to save the masking format.

   The Format Library page reappears with your newly created format displayed in the Format Library table. You can use this format later to mask a column of the same sensitive type.

Providing User-defined and Post-processing Functions

If desired, you can provide user-defined and post-processing functions on the Create Format page. A user-defined choice is available in the Add list, and a post-processing function field is available at the bottom of the page.

- User-defined functions

  To provide a user-defined function, select **User Defined Function** from the **Add** list, then click **Go** to access the input fields.

  A user-defined function passes in the original value as input, and returns a mask value. The data type and uniqueness of the output values must be compatible with the original output values. Otherwise, a failure occurs when the job runs. Combinable, a user-defined function is a PL/SQL function that can be invoked in a `SELECT` statement. Its signature is returned as:

  ```
  Function udf_func (rowid varchar2, column_name varchar2, original_value
  varchar2) returns varchar2;
  ```

  – `rowid` is the min (rowid) of the rows that contain the value `original_value` 3rd argument.

  – `column_name` is the name of the column being masked.

  – `original_value` is the value being masked.

  That is, it accepts the original value as an input string, and returns the mask value.

  Both input and output values are varchar2. For instance, a user-defined function to mask a number could receive 100 as input, the string representation of the number 100, and return 99, the string representation of the number 99. Values are cast appropriately when inserting to the table. If the value is not castable, masking fails.

- Post-processing functions

  To provide a post-processing function, enter it in the **Post Processing Function** field.

A post-processing function has the same signature as a user-defined function, but passes in the mask value the masking engine generates, and returns the mask value that should be used for masking, as shown in the following example:

```
Function post_proc_udf_func (rowid varchar2, column_name varchar2, mask_value
varchar2) returns varchar2;
```

- `rowid` is the min (rowid) of the rows that contain the value `mask_value`.
- `column_name` is the name of the column being masked.
- `mask_value` is the value being masked.

Using Data Masking Format Templates

After you have created at least one format, you can use the format definition as a template in the Create Format page, where you can implement most of the formats using a different name and changing the entries as needed, rather than needing to create a new format from scratch.

To create a new format similar to an existing format, select a format on the Format Library page and click **Create Like**. The masking format you select can either be one you have previously defined yourself, or one from the list of out-of-box masking formats. You can use these generic masking format definitions for different applications.

For instructional details about the various Oracle-supplied predefined masking format definitions and how to modify them to suit your needs, see "Using Oracle-supplied Predefined Data Masking Formats" on page 3-9.

Using Oracle-supplied Predefined Data Masking Formats

Enterprise Manager provides several out-of-box predefined formats. All predefined formats and built-in formats are random. The following sections discuss the various Oracle-supplied format definitions and how to modify them to suit your needs:

- Patterns of Format Definitions
- Category Definitions

> **See Also:** "Installing the DM_FMTLIB Package" on page 11 for information on installing the DM_FMTLIB package so that you can use the predefined masking formats

Patterns of Format Definitions

All of the format definitions adhere to these typical patterns:

- Generate a random number or random digits.
- Perform post-processing on the above-generated value to ensure that the final result is a valid, realistic value.

For example, a valid credit card number must pass Luhn’s check. That is, the last digit of any credit card number is a checksum digit, which is always computed. Also, the first few digits indicate the card type (MasterCard, Amex, Visa, and so forth). Consequently, the format definition of a credit card would be as follows:

- Generate random and unique 10-digit numbers.
- Using a post-processing function, transform the values above to a proper credit card number by adding well known card type prefixes and computing the last digit.

This format is capable of generating 10 billion unique credit card numbers.

Category Definitions

The following sections discuss different categories of these definitions:

- Credit Card Numbers
- United States Social Security Numbers
- ISBN Numbers
- UPC Numbers
- Canadian Social Insurance Numbers
- North American Phone Numbers
- UK National Insurance Numbers
- Auto Mask

By default, these data masking formats are also available in different format styles, such as a hyphen (-) format. If needed, you can modify the format style.

Credit Card Numbers

Out of the box, the format library provides many different formats for credit cards. The credit card numbers generated by these formats pass the standard credit card validation tests by the applications, thereby making them appear like valid credit card numbers.

Some of the credit card formats you can use include:

- MasterCard numbers
- Visa card numbers
- American Express card numbers
- Discover Card numbers
- Any credit card number (credit card numbers belong to all types of cards)

You may want to use different styles for storing credit card numbers, such as:

- Pure numbers
- ’Space’ for every four digits
- ’Hyphen’ (-) for every four digits, and so forth

To implement the masked values in a certain format style, you can set the `DM_CC_FORMAT` variable of the `DM_FMTLIB` package. To install the package, see "Installing the DM_FMTLIB Package" on page 3-11.

United States Social Security Numbers Out of the box, you can generate valid U.S. Social Security (SSN) numbers. These SSNs pass the normal application tests of a valid SSN.

You can affect the format style by setting the `DM_SSN_FORMAT` variable of the `DM_FMTLIB` package. For example, if you set this variable to ’-’, the typical social security number would appear as ’123-45-6789’.

ISBN Numbers Using the format library, you can generate either 10-digit or 13-digit ISBN numbers. These numbers adhere to standard ISBN number validation tests. All of these ISBN numbers are random in nature. Similar to other format definitions, you can affect the "style" of the ISBN format by setting values to `DM_ISBN_FORMAT`.

UPC Numbers Using the format library, you can generate valid UPC numbers. They adhere to standard tests for valid UPC numbers. You can affect the formatting style by setting the `DM_UPC_FORMAT` value of the `DM_FMTLIB` package.

Canadian Social Insurance Numbers Using the format library, you can generate valid Canadian Social Insurance Numbers (SINs). These numbers adhere to standard tests of Canadian SINs. You can affect the formatting style by setting the `DM_CN_SIN_FORMAT` value of the `DM_FMTLIB` package.

North American Phone Numbers Out of the box, the format library provides various possible U.S. and Canadian phone numbers. These are valid, realistic looking numbers that can pass standard phone number validation tests employed by applications. You can generate the following types of numbers:

- Any North American phone numbers
- Any Canadian phone number
- Any U.S.A. phone number

UK National Insurance Numbers Using the format library, you can generate valid unique random UK National Insurance Numbers (NINs). These numbers adhere to standard tests of UK NINs. A typical national insurance number would appear as ’GR 12 56 34 RS’.

Auto Mask This format scrambles characters and numbers into masked characters and numbers and while retaining the format and length of the data, including special characters; for example, ’ABCD_343-ddg’ masked as ’FHDT_657-tte’.

Installing the DM_FMTLIB Package

The predefined masking formats use functions defined in the `DM_FMTLIB` package. This package is automatically installed in the `DBSNMP` schema of your Enterprise Manager repository database. To use the predefined masking formats on a target database (other than the repository database), you must manually install the `DM_FMTLIB` package on that database.

Note: Starting Release 12.1.0.7, the DM_FMTLIB packages does not need to be manually installed as the DM_FMTLIB packages are automatically deployed on the target database while creating a DMS definition or generating a script. The database user must have DBA privileges to auto-deploy the DM_FMTLIB packages.

To install the DM_FMTLIB package:

1. Locate the following scripts in your Enterprise Manager installation:

   ```
   $PLUGIN_HOME/sql/db/latest/masking/dm_fmtlib_pkgdef.sql
   $PLUGIN_HOME/sql/db/latest/masking/dm_fmtlib_pkgbody.sql
   ```

   Where PLUGIN_HOME can be any of the locations returned by the following SQL SELECT statement, executed as SYSMAN:

   ```
   select PLUGIN_HOME from gc_current_deployed_plugin where
   plugin_id='oracle.sysman.db' and destination_type='OMS';
   ```

2. Copy these scripts to a directory in your target database installation and execute them using SQL*Plus, connected as a user that can create packages in the DBSNMP schema.

   You can now use the predefined masking formats in your masking definitions.

3. Select and import any predefined masking format into a masking definition by clicking the **Import Format** button on the Define Column Mask page.

Providing a Data Masking Format to Define a Column

When you create a data masking definition ("Masking with an Application Data Model and Workloads" on page 3-17), you will be either importing a format or selecting one from the available types in the Define Column Mask page. Format entry options are as follows:

- **Array List**

  The data type of each value in the list must be compatible with that of the masked column. Uniqueness must be guaranteed if needed. For example, for a unique key column that already has 10 distinct values, the array list should also contain at least 10 distinct values.

- **Delete**

  Deletes the specified rows as identified by the condition clauses. If a column includes a delete format for one of its conditions, a foreign key constraint or a dependent column cannot refer to the table.

- **Encrypt**

  Encrypts column data by specifying a regular expression. The column values in all the rows must match the regular expression. This format can be used to mask data consistently across databases. That is, for a given value it always generates the same masked value.

  For example, the regular expression [(][1-9][0-9]{2}[)] [_][0-9]{3}[-][0-9]{4} generates U.S. phone numbers such as (123) 456-7890.

  This format supports a subset of the regular expression language. It supports encrypting strings of fixed widths. However, it does not support * or + syntax of regular expressions.

  If a value does not match the format specified, the encrypted value may no longer produce one-to-one mappings. All non-conforming values are mapped to a single encrypted value, thereby producing a many-to-one mapping.

  Note that if the regular expression exceeds the maximum allowable value (all 64 bits set), an exception occurs advising to you to reduce the length of the expression.

  You can reverse the masking of an encrypted column. For more information, see "Data Masking Options" on page 3-22.

- **Fixed Number**

  The type of column applicable to this entry is a `NUMBER` column or a `STRING` column. For example, if you mask a column that has a social security number, one of the entries can be Fixed Number 900. This format is combinable.

- **Fixed String**

The type of column applicable to this entry is a STRING column. For example, if you mask a column that has a License Plate Number, one of the entries can be Fixed String CA. This format is combinable.

- **Null Value**

  Masks the column using a value of NULL. The column must be nullable.

- **Post-Processing Function**

  This is a special function that you can apply to the mask value that the masking engine generates. This function takes the mask value as input and returns the actual mask value to be used for masking.

  The post-processing function is called after the mask value is generated. You can use it, for instance, to add commas or dollar signs to a value. For example, if a mask value is a number such as 12000, the post processing function can modify this to $12,000. Another use is for adding checksums or special encodings for the mask value that is produced.

  In the following statement:

  ```
  Function post_proc_udf_func (rowid varchar2, column_name varchar2, mask_value
  varchar2) returns varchar2;
  ```

  – rowid is the min (rowid) of the rows that contains the value mask_value 3rd argument.

  – column_name is the name of the column being masked.

  – mask_value is the value being masked.

- **Preserve Original Data**

  Retains the original values for rows that match the specified condition clause. This is used in cases where some rows that match a condition do not need to be masked.

- **Random Dates**

  The uniqueness of the Date column is not maintained after masking. This format is combinable.

- **Random Decimal Numbers**

  If used as part of a mixed random string, these have limited usage for generating unique values. This masking format generates unique values within the specified range. For example, a starting value of 5.5 and ending value of 9.99 generates a decimal number ranging from 5.5 to 9.99, both inclusive. This masking format is combinable.

- **Random Digits**

  This format generates unique values within the specified range. For example, for a random digit with a length of [5,5], an integer between [0, 99999] is randomly generated, left padded with '0's to satisfy the length and uniqueness requirement. This is a complementary type of random number, which will not be padded. When using random digits, the random digit pads to the appropriate length in a string. It does not pad when used for a number column. This format is combinable.

  Data masking ensures that the generated values are unique, but if you do not specify enough digits, you could run out of unique values in that range.

- **Random Numbers**

If used as part of a mixed random string, these have limited usage for generating unique values. This format generates unique values within the specified range. For example, a starting value of 100 and ending value of 200 generates an integer number ranging from 100 to 200, both inclusive. Note that Oracle Enterprise Manager release 10.2.0.4.0 does not support float numbers. This format is combinable.

- **Random Strings**

  This format generates unique values within the specified range. For example, a starting length of 2 and ending length of 6 generates a random string of 2 - 6 characters in length. This format is combinable.

- **Regular Expression**

  This format enables you to use regular expressions to search for sensitive data in LOBs (BLOB, CLOB, NCLOB) and replace the data with a fixed string, a fixed number, null, or SQL Expression. Use rules to search for multiple strings within a LOB.

  Examples:

  – Use the regular expression `[0-9]{3}[.][0-9]{3}[.][0-9]{4}` to match strings of the format *nnn.nnn.nnnn* and replace with a masked value, for example, ***.***.****

  – Use the regular expression `<SALARY>[0-9]{2,6}</SALARY>` to zero out salary information by replacing with `<SALARY>0</SALARY>`

  – Use the regular expression `[A-Z]+@[A-Z]+\.[A-Z]{2,4}` to mask e-mail addresses by replacing with john.doe@acme.com

  You can also use this format with data type VARCHAR2 to mask part of a string.

- **Shuffle**

  This format randomly shuffles the original column data. It maintains data distribution except when a column is conditionally masked and its values are not unique.

  For more information, see "Using the Shuffle Format" on page 3-31.

- **Substitute**

  This format uses a hash-based substitution for the original value and always yields the same mask value for any given input value. Specify the substitution masking table and column. This format has the following properties:

  – The masked data is not reversible. That is, this format is not vulnerable to external security breaches because the original value is replaced, so it is not possible to retrieve the original value from the mask value.

  – Masking multiple times with a hash substitute across different databases yields the same mask value. This characteristic is valid across multiple databases or multiple runs assuming that the same substitution values are used in the two runs. That is, the actual rows and values in the substitution table do not change. For example, suppose the two values Joe and Tom were masked to Henry and Peter. When you repeat the same mask on another database using the same substitution table, if there were Bob and Tom, they might be replaced with Louise and Peter. Notice that even though the two runs have different data, Tom is always replaced with Peter.

  – This format does not generate uniqueness.

- **SQL Expression**

  This masking format enables you to enter a SQL Expression for masking a column. Data masking uses this expression to generate masked values to replace the original values. You cannot combine a column using this masking format type with other masking format types, such as Random Numbers or Random Strings.

  The SQL Expression can consist of one or more values, operators, and SQL functions that evaluates to a value. It can also contain substitution columns (columns from the same table as the masked column). You should specify substitution columns within percent signs (%). Use SQL Expressions with `dbms_lob` and other user-defined functions for LOB (BLOB, CLOB, NCLOB) masking.

  Examples:

  – `dbms_random.string('u', 8) || '@company.com'`

    Generates random e-mail addresses.

  – `%first_name% || '.' || %last_name% || '@company.com'`

    Generates e-mail addresses using `first_name` and `last_name` column values. In this example, `first_name` and `last_name` are the substitution columns.

  – **CLOB Masking**

    ```
    dbms_lob.empty_clob()
    ```

    Empties the CLOB.

    ```
    custom_mask_clob(%CLOB_COL%)
    ```

    Applies the custom mask function to the clob column `CLOB_COL`.

  – **Conditional Mask**

    ```
    (case when %PARTY_TYPE%='PERSON' then %PERSON_FIRST_NAME%|| ' '
    ||%PERSON_LAST_NAME% else (select dbms_random.string('U', 10) from
    dual) end)
    ```

    Columns within %% are present in the same table. The expression masks `PERSON_FULL_NAME` with the first and last name; otherwise, the mask is a random string.

  – **Substitution Mask**

    ```
    select MASK_ZIPCODE  from  data_mask.DATA_MASK_ADDR  where ADDR_SEQ
    = ora_hash( %ZIPCODE% , 1000, 1234)
    ```

    Select 1000 rows in the substitution table `data_mask.DATA_MASK_ADDR`. Mask `%ZIPCODE%` with the `MASK_ZIPCODE` column in the substitution table. The row selected is dependent on `ora_hash` and is deterministic in this case. Selection is random if `dbms_random` procedures are used.

- **Substitute**

  This format uses a hash-based substitution for the original value and always yields the same mask value for any given input value. Specify the substitution masking table and column. This format has the following properties:

  – The masked data is not reversible. That is, this format is not vulnerable to external security breaches, because the original value is replaced, so it is not possible to retrieve the original value from the mask value.

  – Masking multiple times with a hash substitute across different databases yields the same mask value. This characteristic is valid across multiple databases or multiple runs assuming that the same substitution values are used in the two runs. That is, the actual rows and values in the substitution table do not change. For example, suppose the two values Joe and Tom were masked to Henry and Peter. When you repeat the same mask on another database using the same substitution table, if there were Bob and Tom, they might be replaced with Louise and Peter. Notice that even though the two runs have different data, Tom is always replaced with Peter.

  - This format does not guarantee uniqueness.

- **Substring**

  Substring is similar to the database `substr` function. The start position can be either a positive or a negative integer. For example, if the original string is `abcd`, a substring with a start position of 2 and length of 3 generates a masked string of bcd. A substring with start position of -2 and length of 3 generates a masked string of cd. This format is combinable.

- **Table Column**

  A table column enables you to select values from the chosen column as the replacement value or part thereof. The data type and uniqueness must be compatible. Otherwise, a failure occurs when the job runs. This format is combinable.

- **Truncate**

  Truncates all rows in a table. If one of the columns in a table is marked as truncated, the entire table is truncated, so no other data masking formats can be specified for any of the other columns. If a table is being truncated, it cannot be referred to by a foreign key constraint or a dependent column.

- **User Defined Function**

  The data type and uniqueness of the output values must be compatible with the original output values. Otherwise, a failure occurs when the job runs.

  In the following statement:

  ```
  Function udf_func (rowid varchar2, column_name varchar2, original_value
  varchar2) returns varchar2;
  ```

  - `rowid` is the min (rowid) of the rows that contain the value `original_value` 3rd argument.
  - `column_name` is the name of the column being masked.
  - `original_value` is the value being masked.

Deterministic Masking Using the Substitute Format

You may occasionally need to consistently mask multiple, distinct databases. For instance, if you run HR, payroll, and benefits that have an employee ID concept on three separate databases, the concept may be consistent for all of these databases, in that an employee's ID can be selected to retrieve the employee's HR, payroll, or benefits information. Based on this premise, if you were to mask the employee's ID because it actually contains his/her social security number, you would have to mask this consistently across all three databases.

Deterministic masking provides a solution for this problem. You can use the Substitute format to mask employee ID column(s) in all three databases. The Substitute format uses a table of values from which to substitute the original value with a mask value.

As long as this table of values does not change, the mask is deterministic or consistent across the three databases.

Masking with an Application Data Model and Workloads

Before creating a masking definition, note the following prerequisites and advisory information:

- Ensure that you have the following minimum privileges for data masking:
  - `EM_ALL_OPERATOR` for Enterprise Manager Cloud Control users
  - `SELECT_CATALOG_ROLE` for database users
  - `SELECT ANY DICTIONARY` privilege for database users
  - `EXECUTE` privileges for the DBMS_CRYPTO package
- Ensure the format you select does not violate check constraints and does not break any applications that use the data.
- For triggers and PL/SQL packages, data masking recompiles the object.
- Exercise caution when masking partitioned tables, especially if you are masking the partition key. In this circumstance, the row may move to another partition.
- Data Masking does not support clustered tables, masking information in object tables, XML tables, and virtual columns. Relational tables are supported for masking.
- If objects are layered on top of a table such as views, materialized views, and PL/SQL packages, they are recompiled to be valid.

If you plan to mask a test system intended for evaluating performance, Oracle recommends the following best practices:

- Try to preserve the production statistics and SQL profiles after masking by adding a pre-masking script to export the SQL profiles and statistics to a temporary table, then restoring after masking completes.
- Run a SQL Performance Analyzer evaluation to understand the masking impact on performance. Any performance changes other than what appears in the evaluation report are usually related to application-specific changes on the masked database.

To create a data masking definition:

1. From the **Enterprise** menu, select **Quality Management**, then select **Data Masking Definitions**. The selection is also available from the **Data Masking and Subsetting** submenu of the **Security** menu on the all databases home page or on a particular database’s home page.

   The Data Masking Definitions page appears, where you can create and schedule new masking definitions and manage existing masking definitions.

2. Click **Create** to go to the Create Masking Definition page.

   A masking definition includes information regarding table columns and the format for each column. You can choose which columns to mask, leaving the remaining columns intact.

3. Provide a required **Name**, **Application Data Model**, and **Reference Database**.

When you click the search icon and select an Application Data Model (ADM) name from the list, the system automatically populates the Reference Database field.

- Optional: Check Ensure Workload Masking Compatibility if you want to mask Capture files and SQL Tuning Sets.

  When you enable this check box, the masking definition is evaluated to determine if the SQL Expression format or conditional masking is being used. If either is in use when you click OK, the option becomes unchecked and an error message appears asking you to remove these items before selecting this option.

  Note: Before proceeding to the next step, one or more sensitive columns must already be defined in the Application Data Model. See "Managing Sensitive Column Types" on page 2-6 for more information.

4. Click **Add** to go to the Add Columns page where you can choose which sensitive columns in the ADM you want to mask. See "Adding Columns for Masking" on page 3-20 for information on adding columns.

   The results appear in the Columns table. Primary key and foreign key columns appear below the sensitive columns.

5. Use filtering criteria to refine sensitive column results. For example, perhaps you want to isolate all columns that have order in the name (column name=order%). You first may have to expose the filter section (**Show Filters**).

6. Use the disable feature to exclude certain columns from masking consideration. All columns are enabled by default. You can disable selected or all columns. You can also search for a subset of columns (column name=order%) to disable. The Status column on the right changes to reflect a column's disabled state. Note that a column's disabled state persists on export of a data masking definition.

7. Optional. Click the edit icon in the Format column to review and edit the masking format.

8. Expand **Show Advanced Options** and decide whether the selected default data masking options are satisfactory.

   For more information, see "Selecting Data Masking Advanced Options" on page 3-21.

9. Click **OK** to save your definition and return to the Data Masking Definitions page.

   At this point, super administrators can see each other's masking definitions.

10. Select the definition and click **Generate Script**. The schedule job dialog opens. You may have to log in to the database first. For information on script generation, see "Scheduling a Script Generation Job" on page 3-24.

    Complete the schedule job dialog by providing the required information, then click **Submit**.

11. A message appears denoting that the job was submitted successfully and you return to the Data Masking Definitions page, where the status is "Generating Script." Click **View Job Details** to open the job summary page.

    When the job completes, click **Log Report** to check whether sufficient disk space is available for the operation, and to determine the impact on other destination objects, such as users, after masking. If any tables included in the masking definition have columns of data type LONG, a warning message may appear. For more information, see "Using Data Masking with LONG Columns" on page 3-33.

12. When the status on the Data Masking Definitions page is "Script Generated," select the script and choose from the following actions:

    - **Clone Database**–to clone and mask the database using the Clone Database wizard (this requires a Database Lifecycle Management Pack license). For more information, see "Cloning the Production Database" on page 3-28.
    - **Save Script**–to save the entire PL/SQL script to your desktop.
    - **Save Mask Bundle**–to download a zip file containing the SQL files generated as part of the Mask in Export script generation option. You can then extract and execute the script to create a masked dump of the database.
    - **View Script**–to view the PL/SQL script, which you can edit and save. You can also view errors and warnings, if any, in the impact report.

    Click **Go** to execute the selected action.

13. If you are already working with a test database and want to directly mask the data in this database, click **Schedule Job**. For information on masking a database, see "Scheduling a Data Masking Job" on page 3-25.

    – Provide the requisite information and desired options. You can specify the database at execution time to any database. The system assumes that the database you select is a clone of the source database. By default, the source database from the ADM is selected.

    – Click **Submit**.

    The Data Masking Definitions page appears. The job has been submitted to Enterprise Manager and the masking process appears. The Status column on this page indicates the current stage of the process.

Note that you can also perform data masking at the source (Mask in Export) as part of a data subsetting definition. See "Creating a Data Subsetting Definition" on page 4-1 for more information.

Editing a Data Masking Definition

Use the following procedure to edit or otherwise work with a data masking definition:

1. From the Data Masking Definitions page, click the radio button next to the definition you want to edit, then click **Edit**.
2. Add columns or edit format entries as you would when creating a new definition, which is explained starting in Step 4 above.
3. Click **OK** on the Edit Masking Definitions page when you have finished editing.
4. Perform these other actions with an existing masking definition by choosing the respective menu item from the **Actions** menu:

    - **Create Like**–to display the masking definition in the Create Masking Definition page where you can customize the definition.
    - **Grant Designer**–to grant Designer privileges on the masking definition to selected roles or administrators, which means the grantees can view and edit the definition. Privileges granted can also be revoked.

- **Grant Operator**–to grant Operator privileges on the masking definition to selected roles or administrators, which means the grantees can view and copy but not edit the definition. Privileges granted can also be revoked.
- **Export**–to export the definition as an XML file for use in other repositories.

Click **Go** to execute the selected action.

Adding Columns for Masking

Use this page to add one or more columns for masking and automatically add foreign key columns. Select the database columns you want to mask from the corresponding schema. After you select the columns, you specify the format used to mask the data within.

> **Note:** You need to add at least one column in the masking definition. Otherwise, you cannot generate a script that creates an impact report that provides information about the objects and resources examined and lists details of any warnings or errors detected.

1. Enter search criteria, then click **Search**.

   The sensitive columns you defined in the ADM appear in the table below.

2. Either select one or more columns for later formatting on the Create Masking Definition page, or formatting now if the data types of the columns you have selected are identical.

3. *Optional*: if you want to mask selected columns as a group, enable Mask selected columns as a group. The columns that you want to mask as a group must all be from the same table.

   Enable this check box if you want to mask more than one column together, rather than separately. When you select two or more columns and then later define the format on the Define Group Mask page, the columns appear together, and any choices you make for format type or masking table apply collectively to all of the columns.

   After you define the group and return to this page, the Column Group column in the table shows an identical number for each entry row in the table for all members of the group. For example, if you have defined your first group containing four columns, each of the four entries in this page will show a number 1 in the Column Group column. If you define another group, the entries in the page will show the number 2, and so forth. This helps you to distinguish which columns belong to which column groups.

4. Either click **Add** to add the column to the masking definition, return to the Create Masking Definition page and define the format of the column later, or click **Define Format and Add** to define the format for the column now.

   The Define Format and Add feature can save you significant time. When you select multiple columns to add that have the same data type, you do not need to define the format for each column as you would when you click Add. For instance, if you search for Social Security numbers (SSN) and the search yields 100 SSN columns, you could select them all, then click **Define Format and Add** to import the SSN format for all of them.

5. Do one of the following:

- If you clicked **Add** in the previous step:

  You will eventually need to define the format of the column in the Create Masking Definition page before you can continue. When you are ready to do so, click the icon in the page Format column for the column you want to format. Depending on whether you decided to mask selected columns as a group on the Add Columns page, either the Define Column mask or Define Group mask appears. Read further in this step for instructions for both cases.

- If you clicked **Define Format and Add** in the previous step and did not check **Mask selected columns as a group**:

  The Define Column Mask page appears, where you can define the format for the column before adding the column to the Create Masking Definition page, as explained below:

  - Provide a format entry for the required Default condition by either selecting a format entry from the list and clicking **Add**, or clicking **Import Format**, selecting a predefined format on the Import Format page, then clicking **Import**.

    The Import Format page displays the formats that are marked with the same sensitive type as the masked column.

  - Add another condition by clicking **Add Condition** to add a new condition row, then provide one or more format entries as described in the previous step.

  - When you have finished formatting the column, click **OK** to return to the Create Masking Definition page.

- If you clicked **Define Format and Add** in the previous step and checked **Mask selected columns as a group**:

  The Define Group Mask page appears, where you can add format entries for group columns that appear in the Create Masking Definition page, as explained below:

  - Select one of the available format types. For complete information on the format types, see the online help for the Defining the Group Masking Format topic.

  - Optionally add a column to the group.

  - When you have finished formatting the group, click **OK** to return to the Create Masking Definition page.

    The results appear in the Columns table. The sensitive columns you selected earlier now appear on this page. Primary key and foreign key columns appear below the sensitive columns.

Selecting Data Masking Advanced Options

The following options on the Data Masking Definitions page are all checked by default, so you need to uncheck the options that you do not want to enable:

- Data Masking Options
- Random Number Generation
- Pre- and Post-mask Scripts

Data Masking Options

The data masking options include:

- Disable redo log generation during masking

  Masking disables redo logging and flashback logging to purge any original unmasked data from logs. However, in certain circumstances when you only want to test masking, roll back changes, and retry with more mask columns, it is easier to uncheck this box and use a flashback database to retrieve the old unmasked data after it has been masked. You can use Enterprise Manager to flashback a database.

  > **Note:** Disabling this option compromises security. You must ensure this option is enabled in the final mask performed on the copy of the production database.

- Refresh statistics after masking

  If you have already enabled statistics collection and would like to use special options when collecting statistics, such as histograms or different sampling percentages, it is beneficial to turn off this option to disable default statistics collection and run your own statistics collection jobs.

- Drop temporary tables created during masking

  Masking creates temporary tables that map the original sensitive data values to mask values. In some cases, you may want to preserve this information to track how masking changed your data. Note that doing so compromises security. These tables must be dropped before the database is available for unprivileged users.

- Decrypt encrypted columns

  This option decrypts columns that were previously masked using the Encrypt format. To decrypt a previously encrypted column, the encryption seed must be the same as the value used to encrypt. For more information, see "Scheduling a Data Masking Job" on page 3-25.

  Decrypt only recovers the original value if the original format used for the encryption matches the original value. If the originally encrypted value did not conform to the specified regular expression, when decrypted, the encrypted value cannot reproduce the original value.

- Use parallel execution when possible

  Oracle Database can make parallel various SQL operations that can significantly improve their performance. Data Masking uses this feature when you select this option. You can enable Oracle Database to automatically determine the degree of parallelism, or you can specify a value. For more information about using parallel execution and the degree of parallelism, see the *Oracle Database Data Warehousing Guide*.

- Recompile invalid dependent objects after masking

  The masking process re-creates the table to be masked and as a consequence, all existing dependent objects (packages, procedures, functions, MViews, Views, Triggers) become invalid. You can specify that the masking process recompile these invalid objects after creating the table, by selecting the check box. Otherwise, invalid objects are not recompiled using `utl_comp` procedures at the end of masking.

If you choose this option, indicate whether to use serial or parallel execution. You can enable Oracle Database to automatically determine the degree, or you can specify a value. For more information about using parallel execution and the degree of parallelism, see the *Oracle Database Data Warehousing Guide*.

Random Number Generation

The random number generation options include:

- Favor Speed

  The `DBMS_RANDOM` package is used for random number generation.

- Favor Security

  The `DBMS_CRYPTO` package is used for random number generation. Additionally, if you use the Substitute format, a seed value is required when you schedule the masking job or database clone job.

Pre- and Post-mask Scripts

When masking a test system to evaluate performance, it is beneficial to preserve the object statistics after masking. You can accomplish this by adding a pre-masking script to export the statistics to a temporary table, then restoring them with a post-masking script after masking concludes.

Use the Pre Mask Script text box to specify any user-specified SQL script that must run before masking starts.

Use the Post Mask Script text box to specify any user-specified SQL script that must run after masking completes. Since masking modifies data, you can also perform tasks, such as rebalancing books or calling roll-up or aggregation modules, to ensure that related or aggregate information is consistent.

The following examples show pre- and post-masking scripts for preserving statistics.

Example 3–1 Pre-masking Script for Preserving Statistics

```
variable sts_task  VARCHAR2(64);

/*Step :1 Create the staging table for statistics*/

exec dbms_stats.create_stat_table(ownname=>'SCOTT',stattab=>'STATS');

/* Step 2: Export the table statistics into the staging table. Cascade results in
all index and column statistics associated with the specified table being exported
as well. */

exec
dbms_stats.export_table_stats(ownname=>'SCOTT',tabname=>'EMP',
partname=>NULL,stattab=>'STATS',statid=>NULL,cascade=>TRUE,statown=>'SCOTT');
exec
dbms_stats.export_table_stats(ownname=>'SCOTT',tabname=>'DEPT',
partname=>NULL,stattab=>'STATS',statid=>NULL,cascade=>TRUE,statown=>'SCOTT');

/* Step 3: Create analysis task */
3. exec :sts_task := DBMS_SQLPA.create_analysis_task(sqlset_name=>
'scott_test_sts',task_name=>'SPA_TASK', sqlset_owner=>'SCOTT');

/*Step 4: Execute the analysis task before masking */
exec DBMS_SQLPA.execute_analysis_task(task_name => 'SPA_TASK',
execution_type=> 'explain plan', execution_name  => 'pre-mask_SPA_TASK');
```

Example 3–2 Post-masking Script for Preserving Statistics

```
*Step 1: Import the statistics from the staging table to the dictionary tables*/

exec
dbms_stats.import_table_stats(ownname=>'SCOTT',tabname=>'EMP',
partname=>NULL,stattab=>'STATS',statid=>NULL,cascade=>TRUE,statown=>'SCOTT');
exec
dbms_stats.import_table_stats(ownname=>'SCOTT',tabname=>'DEPT',
partname=>NULL,stattab=>'STATS',statid=>NULL,cascade=>TRUE,statown=>'SCOTT');

/* Step 2: Drop the staging table */

exec dbms_stats.drop_stat_table(ownname=>'SCOTT',stattab=>'STATS');

/*Step 3: Execute the analysis task before masking */
exec DBMS_SQLPA.execute_analysis_task(task_name=>'SPA_TASK',
execution_type=>'explain plan', execution_name=>'post-mask_SPA_TASK');

/*Step 4: Execute the comparison task */
exec DBMS_SQLPA.execute_analysis_task(task_name =>'SPA_TASK',
execution_type=>'compare', execution_name=>'compare-mask_SPA_TASK');
```

See Also: "Masking a Test System to Evaluate Performance" on page 29 for a procedure that explains how to specify the location of these scripts when scheduling a data masking job

Scheduling a Script Generation Job

To schedule a script generation job:

1. Select the masking definition to generate a script for, then click **Generate Script**.
2. Change the default job name to something meaningful, if desired, and provide an optional job description.
3. Select a reference database from the drop-down list.
4. Select a script generation option:
   - **Mask in Database**–to replace sensitive data in-place with masked data on a specified database (usually copied from production). Use this option only in nonproduction environments. This differs from the **Actions** menu option **Clone Database**, which clones the database and then masks the data.
   - **Mask in Export**–to export masked data from the specified source database (usually production) using Oracle Data Pump. This option is safe to run in a production environment as it does not modify customer data. Note, however, that this option creates temporary tables that get dropped when the masking operation completes.

   Note that you can choose both options; that is, a script to mask the database directly and a script to create a masked dump.
5. Specify a tablespace for temporary objects. Accept the default or select a tablespace where it might be easier to remove sensitive data.
6. Specify credentials to log in to the reference database.
7. Specify to start the job immediately or at a later specified date and time, then click **Submit.**

A message confirms that the job has been scheduled. Refresh the page to see the job results.

> **Note:** You also can generate a masking script remotely or through a script using the EMCLI verb `generate_masking_script`. See the *Oracle Enterprise Manager Command Line Interface* manual for details.

Scheduling a Data Masking Job

To set up the data masking job and schedule its execution:

1. Select the masking definition for which a script has been generated, then click **Schedule Job**.
2. Change the default job name if desired and enter an optional job description.
3. Select a database from the drop-down menu and indicate your preference:
   - **Mask in Database**–to replace sensitive data in-place with masked data on a specified database (usually copied from production). Use this option only in nonproduction environments. This differs from the **Actions** menu option **Clone Database**, which clones the database and then masks the data.
   - **Mask in Export**–to export masked data from the specified source database (usually production) using Oracle Data Pump. This option is safe to run in a production environment as it does not modify customer data. Note, however, that this option creates temporary tables that get dropped when the masking operation completes.

   Your selections affect the check box text that appears below the radio buttons as well as other regions on the page.
4. Proceed according to your selections in Step 3:
   - **Data Mask Options**–Provide the requisite information as follows:
     - After script generation completes, the data masking script is stored in the Enterprise Manager repository. By default, the data masking job retrieves the script from the repository and copies it to the `$ORACLE_HOME/dbs` directory on the database host, using a generated file name. The Script File Location and Name fields enable you to override the default location and generated file name.
     - Workloads–Select options to mask SQL tuning sets and capture files, as appropriate. Browse to the file location where you want to capture the files.
     - Detect SQL Plan Changes Due to Masking–Run the SQL Performance Analyzer to assess the impact of masking. Provide a task name and browse to the corresponding SQL tuning set.
   - **Data Export Options**–Provide the requisite information as follows:
     - Specify a directory where to save the mask dump. The drop-down list consists of directory objects that you can access. Alternatively, you can select a custom directory path. Click the check box if you want to speed the process by using an external directory. Recommended default: `DATA_FILE_DIR`.
     - Specify appropriate values if you want to override the defaults: enter a name for the export file; specify a maximum file size in megabytes; specify

the maximum number of threads of active execution operating on behalf of the export job. This enables you to consider trade-offs between resource consumption and elapsed time.

   - Select whether to enable dump file compression and encryption. Enter and confirm an encryption seed, if appropriate. Note that you must specify the same encryption seed value to decrypt a previously encrypted column.

5. Specify credentials to log in to the database host.

> **Note:** Provide the SSH credentials for On-Cloud Databases. For more information, refer the "Support for Data Masking and Subsetting in Oracle Cloud" section.

6. Specify credentials to log in to the reference database.
7. Specify to start the job immediately or at a later specified date and time, then click **Submit.**

   A message confirms that the job has been scheduled. Refresh the page to see the job results.

> **Note:** You also can submit a masking job remotely or through a script using the EMCLI verb `submit_masking_script`. See the *Oracle Enterprise Manager Command Line Interface* manual for details.

Estimating Space Requirements for Masking Operations

Here are some guidelines for estimating space requirements for masking operations. These estimates are based on a projected largest table size of 500GB. In making masking space estimates, assume a “worst-case scenario."

- For in-place masking:
   - 2 * 500GB for the mapping table (the mapping table stores both the original and the masked columns. Worse case is every column is to be masked).
   - 2 * 500GB to accommodate both the original and the masked tables (both exist in the database at some point during the masking process).
   - 2 * 500GB for temporary tablespace (needed for hash joins, sorts, and so forth).

   Total space required for the worst case: 3TB.

- For at-source masking:
   - 2 * 500GB for the mapping table (as for in-place masking).
   - 2 * 500GB for temporary tablespace (as for in-place masking).
   - Sufficient file system space to accommodate the dump file.

   Total space required for the worst case: 2TB plus the necessary file system space.

In either case, Oracle recommends that you set the temp and undo tablespaces to auto extend.

You can specify a tablespace for mapping tables during script generation. If you do not specify a tablespace, the tables are created in the tablespace of the executing user. Note that space estimations are provided during script generation, with resource warnings as appropriate. There are some situations, for example when using the shuffle format, that do not require a mapping table. In these cases, updates to the original table happen in-line.

On Adding Dependent Columns

Dependent columns are defined by adding them to the Application Data Model. The following prerequisites apply for the column to be defined as dependent:

- A valid dependent column should not already be included for masking.
- The column should not be a foreign key column or referenced by a foreign key column.
- The column data should conform to the data in the parent column.

If the column does not meet these criteria, an "Invalid Dependent Columns" message appears when you attempt to add the dependent column.

Masking Dependent Columns for Packaged Applications

The following procedure explains how to mask data across columns for packaged applications in which the relationships are not defined in the data dictionary.

To mask dependent columns for packaged applications:

1. Go to Application Data Modeling and create a new Application Data Model (ADM) using metadata collection for your packaged application suite.

   When metadata collection is complete, edit the newly created ADM.

2. Manually add a referential relationship:

   a. On the **Referential Relationships** tab, select **Add Referential Relationship** from the **Actions** menu.

      The Add Referential Relationship pop-up window appears.

   b. Select the requisite Parent Key and Dependent Key information.

   c. In the Columns Name list, select a dependent key column to associate with a parent key column.

   d. Click **OK** to add the referential relationship to the ADM.

      The new dependent column now appears in the referential relationships list.

3. Perform sensitive column discovery.

   When sensitive column discovery is complete, review the columns found by the discovery job and mark them sensitive or not sensitive as needed.

   When marked as sensitive, any discovery sensitive column also marks its parent and the other child columns of the parent as sensitive. Consequently, it is advisable to first create the ADM with all relationships. ADM by default, or after running drivers, may not contain denormalized relationships. You need to manually add these.

   For more information about sensitive column discovery, see step 6 on page 2-5.

4. Go to Data Masking and create a new masking definition.

5. Select the newly created ADM and click **Add**, then **Search** to view this ADM's sensitive columns.

6. Select columns based on your search results, then import formats for the selected columns.

Enterprise Manager displays formats that conform to the privacy attributes.

7. Select the format and generate the script.
8. Execute the masking script.

   Enterprise Manager executes the generated script on the target database and masks all of your specified columns.

Cloning the Production Database

When you clone and mask the database, a copy of the masking script is saved in the Enterprise Manager repository and then retrieved and executed after the clone process completes. Therefore, it is important to regenerate the script after any schema changes or modifications to the production database.

To clone and optionally mask the masking definition’s target database:

1. From the Data Masking Definitions page, select the masking definition you want to clone, select **Clone Database** from the Actions list, then click **Go**.

   The Clone Database: Source Type page appears.

   The Clone Database wizard appears, where you can create a test system to run the mask.
2. Specify the type of source database backup to be used for the cloning operation, then click **Continue**.
3. Proceed through the wizard steps as you ordinarily would to clone a database. For assistance, refer to the online help for each step.
4. In the Database Configuration step of the wizard, add a masking definition, then select the Run SQL Performance Analyzer option as well as other options as desired or necessary.
5. Schedule and then run the clone job.

Importing and Exporting a Data Masking Definition

You can import and re-use a previously exported data masking definition saved as an XML file to the current Enterprise Manager repository. You also can import an Oracle-supplied data masking definition from the Software Library.

> **Note:** You also can export and import data masking definitions remotely or through a script using the EMCLI verbs `export_masking_definition` and `import_masking_definition`, respectively. See the *Oracle Enterprise Manager Command Line Interface* manual for details.

Importing a Previously Exported Masking Definition

Note the following advisory information:

- The XML file format must be compliant with the masking definition XML format.
- Verify that the name of the masking definition to be imported does not already exist in the repository, and the source database name identifies a valid Enterprise Manager target.
- Verify that the value in the XML file to be imported refers to a valid database target.

To import a data masking definition:

1. From the Data Masking Definitions page, click **Import**.

   The Import Masking Definition page appears.

2. Specify a name for the masking definition and select the ADM to associate with the template. The Reference Database is automatically provided.

3. Browse for the XML file, or specify the name of the XML file, then click **Continue**.

   The Data Masking Definitions Page reappears and displays the imported definition in the table list for subsequent viewing and masking.

Importing a Data Masking Template from the Software Library

The Self Update feature ensures that the latest Oracle-supplied data masking templates are available in the Software Library. You can also check for updates. Go to the Self Update page and check for Data Masking and Subsetting updates. If present, download and apply them so that they are available in the Software Library.

1. On the Data Masking Definitions page, click **Import from Software Library**.

   The Import Masking Definition page appears.

2. Select a masking template in the Software Library list.

3. Specify a name for the masking definition and select the ADM to associate with the template. The Reference Database is automatically provided.

4. Click **Continue**.

   The Data Masking Definitions Page reappears and displays the imported definition in the table list for subsequent viewing and masking.

You can also export a data masking definition from the Software Library.

1. On the Data Masking Definitions page, click **Export from Software Library**.

2. Select a masking template in the Software Library list.

3. Click **Export**.

   Save the template file for import into a different repository.

Masking a Test System to Evaluate Performance

After you have created a data masking definition, you may want to use it to analyze the performance impact from masking on a test system. The procedures in the following sections explain the process for this task for masking only, or cloning and masking.

Using Only Masking for Evaluation

To use only masking to evaluate performance:

1. From the Data Masking Definitions page, select the masking definition to be analyzed, then click **Schedule Job**.

   The Schedule Data Masking Job page appears.

2. At the top of the page, provide the requisite information.

   The script file location pertains to the masking script, which also contains the pre- and post-masking scripts you created in "Pre- and Post-mask Scripts" on page 3-23.

3. In the Encryption Seed section, provide a text string that you want to use for encryption.

   This section only appears for masking definitions that use the Substitute or Encrypt formats. The seed is an encryption key used by the encryption/hash-based substitution APIs, which makes masking deterministic rather than random.

4. In the Workloads section:

   a. Select the **Mask SQL Tuning Sets** option, if desired.

      If you use a SQL Tuning Set that has sensitive data to evaluate performance, it is beneficial to mask it for security, consistency of data with the database, and to generate correct evaluation results.

   b. Select the **Capture Files** option, if desired, then select a capture directory.

      When you select this option, the contents of the directory is masked. The capture file masking is executed consistently with the database.

5. In the Detect SQL Plan Changes Due to Masking section, leave the Run SQL Performance Analyzer option unchecked.

   You do not need to enable this option because the pre- and post-masking scripts you created, referenced in step 2, already execute the analyzer.

6. Provide credentials and scheduling information, then click **Submit**.

   The Data Masking Definitions page reappears, and a message appears stating that the Data Masking job has been submitted successfully.

   During masking of any database, the AWR bind variable data is purged to protect sensitive bind variables from leaking to a test system.

7. When the job completes successfully, click the link in the SQL Performance Analyzer Task column to view the executed analysis tasks and Trial Comparison Report, which shows any changes in plans, timing, and so forth.

Using Cloning and Masking for Evaluation

Using both cloning and masking to evaluate performance is very similar to the procedure described in the previous section, except that you specify the options from the Clone Database wizard, rather than from the Schedule Data Masking Job page.

To use both cloning and masking to evaluate performance:

1. Follow the steps described in "Cloning the Production Database" on page 3-28.

2. At step 4, the format of the Database Configuration step appears different from the Schedule Data Masking Job page discussed in "Using Only Masking for Evaluation", but select options as you would for the Schedule Data Masking Job page.

3. Continue with the wizard steps to complete and submit the cloning and masking job.

Upgrade Considerations

Upgrading data masking definitions from 10 or 11 Grid Control to 12*c* Cloud Control assumes that you have completed the following tasks:

- Upgraded Enterprise Manager to 12*c*

- Downloaded the latest database plug-in using Self Update and deployed the plug-in to OMS and Management Agent

Completing these tasks automatically upgrades the masking definitions and creates for each a shell Application Data Model (ADM) that becomes populated with the sensitive columns and their dependent column information from the legacy mask definition. The ADM, and hence data masking, then remains in an unverified state, because it is missing the dictionary relationships.

Proceed as follows to complete the masking definition upgrade:

1. From the **Enterprise** menu, select **Quality Management**, then select **Application Data Modeling**. The selection is also available from the **Security** menu on the Databases page (**Database Load Map**).
2. For each shell ADM (verification status is Needs Upgrade), do the following:
   a. Select the ADM in the table.
   b. From the **Actions** menu, select **Upgrade and Verify**.
   c. Schedule and submit the job.

      When the job completes, verification status should be Valid.
3. From the **Enterprise** menu, select **Quality Management**, then select **Data Masking Definitions**. The selection is also available from the **Data Masking and Subsetting** submenu of the **Security** menu on the all databases home page or on a particular database's home page.
4. For each upgraded masking definition, do the following:
   a. Open the masking definition for editing.
   b. In **Advanced Options**, select the "Recompile invalid dependent objects after masking" option, with Parallel and Default settings.
   c. Click **OK** to save your changes.
5. Next, schedule a script generation job for each upgraded masking definition.

You can now resume masking with the upgraded data masking definitions.

> **See Also:** "On Adding Dependent Columns" on page 27 for information on dependent columns

Consider these other points regarding upgrades:

- You can combine multiple upgraded ADMs by exporting an ADM and performing an Import Content into another ADM.
- An upgraded ADM uses the same semantics as for upgrading a legacy mask definition (discussed above), in that you would need to perform a validation.
- An 11.1 Grid Control E-Business Suite (EBS) masking definition based on an EBS masking template shipped from Oracle is treated as a custom application after the upgrade. You can always use the approach discussed in the first bulleted item above to move into a newly created EBS ADM with all of the metadata in place. However, this is not required.

Using the Shuffle Format

A shuffle format is available that does not preserve data distribution when the column values are not unique and also when it is conditionally masked. For example, consider the Original Table (Table 3–1) that shows two columns: EmpName and Salary. The Salary column has three distinct values: 10, 90, and 20.

Table 3–1 Original Table (Non-preservation)

EmpName	Salary
A	10
B	90
C	10
D	10
E	90
F	20

If you mask the Salary column with this format, each of the original values is replaced with one of the values from this set. Assume that the shuffle format replaces 10 with 20, 90 with 10, and 20 with 90 (Table 3–2).

Table 3–2 Mapping Table (Non-preservation)

EmpName	Salary
10	20
90	10
20	90

The result is a shuffled Salary column as shown in the Masked Table (Table 3–3), but the data distribution is changed. While the value 10 occurs three times in the Salary column of the Original Table, it occurs only twice in the Masked Table.

Table 3–3 Masked Table (Non-preservation)

EmpName	Salary
A	20
B	10
C	20
D	20
E	10
F	90

If the salary values had been unique, the format would have maintained data distribution.

Using Group Shuffle

Group shuffle enables you to perform a shuffle within discrete units, or groups, where there is a relationship among the members of the group. Consider the case of shuffling the salaries of employees. Table 3–4illustrates the group shuffle mechanism, where employees are categorized as managers (M) or workers (W), and salaries are shuffled within job category.

Table 3–4 Group Shuffle Using Job Category

Employee	Job Category	Salary	Shuffled Salary
Alice	M	90	88
Bill	M	88	90
Carol	W	72	70
Denise	W	57	45
Eddie	W	70	57
Frank	W	45	72

Using Conditional Masking

To demonstrate how conditional masking can handle duplicate values, add to Table 3–4 another job category, assistant (A), where the employee in this category, George, earns the same as Frank. Assume the following conditions:

- If job category is M, replace salary with a random number between 1 and 10.
- If job category is W, set salary to a fixed number (01).
- Default is to preserve the existing value.

Applying these conditions results in the masked values shown in Table 3–5:

Table 3–5 Using Job Category for Group Shuffle

Employee	Job Category	Salary	Conditional Result
Alice	M	90	5
Bill	M	88	7
Carol	W	72	01
Denise	W	57	01
Eddie	W	70	01
Frank	W	45	01
George	A	45	45

Conditional masking works when there are duplicate values provided there are no dependent columns or foreign keys. If either of these is present, a "bleeding condition" results in the first of two duplicate values becoming the value of the second. So, in the example, George’s salary is not preserved, but becomes 01.

Using Data Masking with LONG Columns

When data masking script generation completes, an impact report appears. If the masking definition has tables with columns of data type LONG, the following warning message is displayed in the impact report:

```
The table <table_name> has a LONG column. Data Masking uses "in-place" UPDATE to
mask tables with LONG columns. This will generate undo information and the
original data will be available in the undo tablespaces during the undo retention
period. You should purge undo information after masking the data. Any orphan rows
in this table will not be masked.
```

4

Data Subsetting

This chapter describes how to perform the following tasks:

- Creating a Data Subsetting Definition
- Importing and Exporting Subset Templates and Dumps
- Creating a Subset Version of a Target Database
- Synchronizing a Subsetting Definition with an Application Data Model
- Granting Privileges on a Subsetting Definition

The chapter also covers the Integrated Subset and Mask capability, where you perform data masking and subsetting in a single task flow. ("About Integrated Subset and Mask" on page 4-13), and outlines a number of scenarios to demonstrate the process ("Integrated Subset and Mask Scenarios" on page 4-14).

You must have the Oracle Data Masking and Subsetting Pack license to use data subsetting features.

> **Note:** Data subsetting is supported only in Oracle Database versions 10.1 and higher. The procedures in this chapter are applicable only to Oracle Enterprise Manager Cloud Control 12.1 and higher.

Creating a Data Subsetting Definition

The procedure described in this section enables you to create a subset database, after which you can perform other tasks, such as editing the properties of the data subsetting definition or exporting a data subsetting definition.

The interface also allows you to apply ns while creating the data subsetting definition. For more information, see "About Integrated Subset and Mask" on page 4-13.

Before proceeding, ensure that you have the following privileges:

- `EM_ALL_OPERATOR` for Enterprise Manager Cloud Control users
- `SELECT_CATALOG_ROLE` for database users
- `SELECT_ANY_DICTIONARY` privilege for database users
- Additionally, to perform an in-place delete operation, the DBA user must be granted the `EXECUTE_ANY_TYPE` privilege

To create a data subsetting definition:

1. From the Enterprise menu, select **Quality Management**, then **Data Subsetting Definitions**. The selection is also available from the **Data Masking and**

Subsetting submenu of the **Security** menu on the all databases home page or on a particular database's home page.

2. On the Data Subsetting Definitions page, select **Create** from the **Actions** menu, or just click the **Create** icon.

3. Define the data subsetting definition properties:

    a. Provide the requisite information in the General pop-up that appears, then click **Continue**.

    You can select any source database associated with the Application Data Model.

    If you are performing masking within the subsetting definition, you must select the same ADM and target used in creating the masking definition.

    b. Provide a job name, credentials, and specify a schedule in the Schedule Application Detail Collection pop-up that appears, then click **Submit**.

    If you want to use new credentials, choose the New Credentials option. Otherwise, choose the Preferred Credentials or Named Credentials option.

    The space estimate collection job runs, and then displays the Data Subsetting Definitions page. Your definition appears in the table, and the Most Recent Job Status column should indicate Scheduled, Running, or Succeeded, depending on the schedule option selected and time required to complete the job.

4. Select the definition within the table, then select **Edit** from the **Actions** menu.

    The Database Login page appears.

5. Select either Named Credentials or New Credentials if you have not already set preferred credentials, then click **Login**.

6. In the Applications subpage of the Edit page, move applications from the Available list to the Selected list as follows:

    - If you intend only to mask the data (no subsetting), select all applications.
    - If you intend only to subset the data (no masking), select specific applications as appropriate.
    - If you intend both to subset and mask the data, the applications selected must include those that the masking definitions require.

    The names of application suites, applications, and application modules are maintained in the Application Data Model.

7. Click the **Table Rules** tab.

    Note: If you are masking only, set the Default Table Rules option to include all rows and skip to Step 13.

    You can add rules here to define the data to include in the subset.

8. Select **Actions**, then **Create** to display the Table Rule pop-up, or just click the **Create** icon.

    a. Select the application for which you want to provide a rule.

    Associate the rule with all tables, a specific table, or a category of tables.

b. In the Rows to Include section, select the option that best suits your needs for a representative sample of production data. If you do not want to include all rows, you can include some rows by specifying a percentage portion of the rows. For finer granularity, you could specify a Where clause, such as where region_id=6.

   For more information on specifying Where clauses, see Step e.

c. In the Include Related Rows section, do one of the following:

   – Select **Ancestor and Descendant Tables**

     This rule impacts the parent and child columns, and ensures that referential integrity is maintained, and that child columns are also selected as part of the subset.

   – Select **Ancestor Tables Only**

     This rule only impacts the parent columns, and ensures that referential integrity is maintained.

   If you disable the Include Related Rows check box, referential integrity may not be maintained. However, you can subsequently provide additional rules to restore referential integrity. You can disable this check box whether or not you specify a Where clause.

d. If you want to specify a Where clause, go to the next step. Otherwise, skip to Step 9.

e. Provide a rule parameter, if desired, for the clause.

   For instance, if you specify a particular value for an employee ID as employee_id=:emp_id, you could enter query values for the default of 100:

   – Select the Rows Where button and enter employee_id=:emp_id.

   – Click **OK** to save the rule and return to the Table Rules tab.

     If this is a new rule, a warning appears stating that "Rule parameters corresponding to the bind variables ’emp_id’ should be created before generating subset."

   – Select the table rule, click the **Rule Parameters** tab, then click **Create**.

     The Rule Parameter Properties pop-up appears.

   – Enter emp_id for the Name and 100 for the Value.

   Note: The colon (:) preceding emp_id is only present in the Where clause, and not required when creating a new rule parameter.

   – Click **OK** to save the properties, which now appear in the **Rule Parameters** tab.

   – Skip to Step 10.

9. Click **OK** to save the rule and return to the **Table Rules** tab.

   The new rule is displayed in the list. The related tables are displayed in the table below. Related rows from the tables are included in the subset to provide referential integrity in the subset database.

10. In the Related Tables section of the **Table Rules** tab, you can manage the size of the subset by controlling the levels of ancestors and descendants within the subset.

Notice that each node in the table has a check box. By default, all nodes are included in the subset, as indicated by the check mark. Deselect the check box to exclude a node from the subset. The deselection option is disabled for parent rows (the join columns to the right identify parent and child rows). In addition, you can make these other refinements to subset content:

- Click **Allow Excluding Parent Tables**. This enables the check marks that were grayed out. You can now selectively exclude parent rows from the subset by deselecting the check box.
- Select a node within the table and click **Add Descendants** to include related rows. In the dialog that opens, make appropriate selections and click **OK**.

As you make these refinements, columns on the right reflect the effect on space estimates for the subset. The Related Tables section also denotes the processing order of the ancestor and descendant tables, including the detailed impact of including each table. When you are done with the refinements, go to the **Space Estimates** tab to see a finer granularity of the impact on the overall size of the subset.

11. In the Default Table Rows section of the **Table Rules** tab, choose whether you want to include or exclude the tables not affected by the defined rules in the subset.

    When you select the Include All Rows option, all of the rows for the table are selected as part of the subset.

    This is a global rule and applies to the entire subset. You can only select the Include All Rows option when all of the rules have a scope of None. A scope of None is established when you uncheck the Include Related Rows option in the Table Rule pop-up.

    Note: For a subsetting definition that has column mask rules (see Step 12), be sure to use table rules to include the corresponding tables. You can use the Default Table Rules option to include all tables not affected by table rules, if required.

12. *Optional*: Click the **Column Mask Rules** tab to integrate masking with the subsetting definition.

    a. Click **Create** and enter search criteria to filter on columns within the schema. These would typically be vertical columns such as CLOB AND BLOB columns.

    Note: If you are using column mask rules instead of masking definitions (see Step 13), you can select no more than 10 columns in a given table. This restriction applies to the export method but not to the in-place delete method.

    Click **OK**.

    b. Select a row or rows in the column search results and click **Manage Masking Formats**.

    c. In the pop-up dialog, select a masking format and value to apply to the columns. For multiselection, the same format must be appropriate for all columns. If you select multiple columns, ensure that the column mask rule format you choose is applicable to the selected columns. Use the columns (flags) **not null** and **unique** to enforce compliance.

Click **OK** to apply the masking format to the columns.

13. *Optional*: Click the **Data Masking** tab to integrate masking definitions with the subsetting operation or to perform mask in export only.

    a. Click **Add**.

    b. In the pop-up dialog, enter search criteria to retrieve appropriate definitions. Be sure to select the desired radio button (All or Any). All formats except compound masking are supported for integrated masking.

    > **Note:** No single table within a masking definition can have more than 10 masked columns if you are using the export method. The restriction does not apply to the in-place delete method.

    Click **OK**.

    The search results appear in the data masking table.

14. Click the **Space Estimates** tab.

    - Note the value in the Estimated Subset Size MB column. The space estimates depend on optimizer statistics, and the actual distribution of data can only be calculated if histogram statistics are present.
    - Whenever you add new rules, recheck the space estimates for updated values.
    - Data in the Space Estimates subpage is sorted with the largest applications appearing at the top.

    > **Note:** Space estimates do not reflect the effect of data masking, if used.

    If you provide a Where clause and subsequent rule parameter properties, the Space Estimates subpage is updated with the value contained in the **Rule Parameters** tab.

15. *Optional*: click the **Pre/Post Subset Scripts** tab.

    - You can specify a pre-subset script to run on the subset database before you select subset data.
    - You can specify a post-subset script to run on the subset database after you assemble the subset data.
    - Either script type runs on the source database.

    Pre- and post-scripting provides the ability to do SQL-based customization. Perhaps you want to create an index prior to subsetting to reduce overhead. Post-subsetting, you can delete the index.

16. Click **Return**.

    The definition is complete and displayed in the Data Subsetting Definitions table.

You can now proceed with subset generation. Alternatively, you may want to save the script for future use. Saving the script gives you the opportunity to review and edit it. As it is standard SQL, you can schedule and run it outside of Enterprise Manager, by calling the script from `dbms_schedule`, for example. In either case, you must decide whether to export data to a dump file or delete data from a target database.

> **Tip:** If you have a very large database of 4 terabytes, for instance, and you want to export a small percentage of the rows, such as 10%, it is more advantageous to use the export method. Using the in-place delete method would require 3.6 terabytes of data, which would not perform as quickly as the export method.
>
> The in-place delete method is recommended when the amount of data being deleted is a small percentage of the overall data size.

- Generating a Subset
- Saving a Subset Script

Generating a Subset

To prepare and submit a job to generate a subset:

1. Select the definition within the table, open the **Actions** menu, then select **Generate Subset**.

   The Subset Mode pop-up appears.

2. Select a target database that is either the same target database you used to create the subset model, or similar to this database regarding the table schema and objects.

3. Decide if you want to create a subset by writing subset data to export files, or by deleting data from a target database.

   Choosing to delete data creates an in-place subset by removing/deleting unwanted data from a cloned copy of the production database, not from the production database itself. Only data satisfying the rules are retained. Choosing this option presupposes that the target database is a cloned copy of the production database.

   Select either Named Credentials or New Credentials if you have not already set preferred credentials.

   If you have defined any parameters from the **Rule Parameters** tab, they appear in the table at the bottom. You can change a parameter value by clicking on the associated field in the Value column.

4. Click **Continue** to access the Parameters pop-up. The contents of the pop-up depend on whether you chose the export or delete option in the previous step.

   For **Writing Subset Data to Export Files**, provide the requisite information, then click **Continue** to schedule the job.

   – Specify a subset directory where to save the export dump. The drop-down list consists of directory objects for which you have access. Alternatively, you can select a custom directory path. Click the check box if you want to speed the process by using an external directory. Recommended default: `DATA_PUMP_DIR`.

   – Specify appropriate values if you want to override the defaults: enter a name for the export file; specify a maximum file size in megabytes; specify the maximum number of threads of active execution operating on behalf of the export job. This enables you to consider trade-offs between resource consumption and elapsed time.

  - Select whether to enable dump file compression and encryption. Enter and confirm an encryption password, if appropriate. Log file generation is selected by default.

  For **Deleting Data From a Target Database**, provide the requisite information, then click **Continue** to schedule the job.

  - Specify a subset directory where to save the subset scripts. The drop-down list consists of directory objects to which you have access. Alternatively, you can select a custom directory path. Recommended default: `DATA_FILE_DIR`.
  - You must enable the check box indicating that the selected target is not a production database in order to proceed.

5. Click **Continue** to schedule the job from the Generate Subset Schedule pop-up, then click **Submit**. For the delete option, you must specify and confirm an encryption seed.

   The Data Subsetting Definitions page reappears, and the Most Recent Job Status column shows that the subset job is running, and subsequently that it has succeeded.

After performing this procedure, you can now create a subset database with the generated export files at any time.

> **Note:** You also can generate the subset remotely or through a script using the EMCLI verb `generate_subset`. See the *Oracle Enterprise Manager Command Line Interface* manual for details.

Saving a Subset Script

To prepare and submit a job to save a subset script:

1. Select the definition within the table, open the **Actions** menu, then select **Save Subset Script**. The Subset Mode pop-up appears.
2. Select a target database that is either the same target database you used to create the subset model, or similar to this database regarding the table schema and objects.
3. Decide if you want to create a subset by writing subset data to export files, or by deleting data from a target database.

   Choosing to delete data creates an in-place subset by removing/deleting unwanted data from a cloned copy of the production database, rather than a production database. Only data satisfying the rules are retained. Choosing this option presupposes that the target database is a cloned copy of the production database.

   Select either Named Credentials or New Credentials if you have not already set preferred credentials.

   If you have defined any parameters from the **Rule Parameters** tab, they appear in the table at the bottom. You can change a parameter value by clicking on the associated field in the Value column.
4. Click **Continue** to access the Parameters pop-up. The contents of the pop-up depend on whether you chose the export or delete option in the previous step.

   For **Writing Subset Data to Export Files**, provide the requisite information, then click **Continue** to schedule the job.

- Specify a subset directory where to save the export dump. The drop-down list consists of directory objects for which you have access. Alternatively, you can select a custom directory path. Click the check box if you want to speed the process by using an external directory. Recommended default: `DATA_PUMP_DIR`.
- Specify appropriate values if you want to override the defaults: enter a name for the export file; specify a maximum file size in megabytes; specify the maximum number of threads of active execution operating on behalf of the export job. This enables you to consider trade-offs between resource consumption and elapsed time.
- Select whether to enable dump file compression and encryption. Enter and confirm an encryption password, if appropriate. Log file generation is selected by default.

For **Deleting Data From a Target Database**, provide the requisite information, then click **Continue** to schedule the job.

- Specify a subset directory where to save the subset scripts. The drop-down list consists of directory objects to which you have access. Alternatively, you can select a custom directory path. Recommended default: `DATA_FILE_DIR`.
- You must enable the check box indicating that the selected target is not a production database in order to proceed.

5. Click **Continue**. A progress indicator tracks script generation. When complete, the Files table lists the results of script generation.
6. Click **Download**. In the File Download pop-up that appears, click **Save**.
7. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The file containing the scripts (`SubsetBundle.zip`) now appears at the specified location on your desktop.

To run the saved script at a later time:

1. Port the ZIP file to the target database and extract it to a directory on which you have the requisite privileges.
2. Change directory to where you extracted the files.
3. Execute the following script from the SQL command line:

   ```
   subset_exec.sql
   ```

   Note that if you want to change generated parameter settings, you can do so by editing the following file in a text editor prior to executing the script:

   ```
   subset_exec_params.lst
   ```

Importing and Exporting Subset Templates and Dumps

A subset template is an XML file that contains the details of the subset, consisting of the application, subset rules, rule parameters, and pre-scripts or post-scripts. When you create a subsetting definition and specify that you want to write subset data to export files, the export files become a template that you can subsequently import for reuse. You would import the template to perform subset operations on a different database.

Typically, the workflow is that you would first import a previously exported ADM template, which is another XML file, while creating an ADM. You would then import the related subset template while creating a data subsetting definition. You could alternatively select an existing ADM (skipping the import ADM flow) while importing the subset template.

- Importing a Subsetting Definition
- Exporting a Subsetting Definition

> **Note:** You also can export and import data subsetting definitions, or import a subset dump remotely or through a script using the EMCLI verbs `export_subset_definition`, `import_subset_definition`, and `import_subset_dump`, respectively. See the *Oracle Enterprise Manager Command Line Interface* manual for details.

Importing a Subsetting Definition

There are three methods of import:

- Import a subsetting definition XML file from the desktop
- Import a subset dump
- Import a subsetting definition XML file from the Software Library

To import a subsetting definition XML file from your desktop:

1. From the **Actions** menu, select **Import**, then select **File from Desktop**.
2. In the pop-up that appears:
   - Specify a name for the subsetting definition
   - The ADM on which the subset is based
   - A source database
   - The location on your desktop from which you want to import the subsetting definition
   - Click **Continue**
3. In the pop-up that appears:
   - Enter a descriptive job name (or accept the default)
   - Provide credentials
   - Schedule the job
   - Click **Submit**

   After the job runs successfully, the imported subset appears in the list of subsets in the table on the Data Subsetting Definitions page.

To import a subset dump:

1. From the **Actions** menu, select **Import**, then select **Subset Dump**.
2. In the pop-up that appears:
   - Select a target database
   - Provide database and host credentials, then click **Login**.

Note: Provide the SSH credentials for On-Cloud Databases. For more information, refer the "Support for Data Masking and Subsetting in Oracle Cloud" section.

- Specify the location of the dump file, which can be in a selected directory on the target database or at a custom path on the target. Note that if the original export action used an external location for the dump files, the location must be specified as well.
- Click **Continue**.

3. In the pop-up that appears:
   - Select whether to import both metadata and data, or data only. If data only, indicate if you want to truncate, that is, overlay existing data or append to existing data.
   - Perform tablespace remapping as necessary
   - Perform schema remapping as necessary
   - Select log file options
   - Click **Continue**
4. In the pop-up that appears:
   - Enter a descriptive job name (or accept the default)
   - Schedule the job
   - Click **Submit**

   The job reads the dump files and loads the data into the selected target database.

To import a subsetting definition XML file from the Software Library:

1. From the **Actions** menu, select **Import**, then select **File from Software Library**.
2. In the Import Data Subset Definition from Software Library pop-up that appears:
   - Selected the desired subsetting definition XML file on the left
   - Provide the ADM properties on the right.
   - Click **Continue**
3. In the pop-up that appears:
   - Enter a descriptive job name (or accept the default)
   - Provide credentials
   - Schedule the job
   - Click **Submit**

   After the job runs successfully, the imported subset appears in the list of subsets in the table on the Data Subsetting Definitions page.

Exporting a Subsetting Definition

There are two methods of export:

- Export a selected subsetting definition to the desktop
- Export a subsetting definition from the Software Library

To export a subsetting definition as an XML file to your desktop:

1. From the Data Subsetting Definitions page, select the subsetting definition you want to export.
2. From the **Actions** menu, select **Export**, then select **Selected Subset Definition**.
3. In the File Download pop-up that appears, click **Save**.
4. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The system converts the subsetting definition into an XML file that now appears at the specified location on your desktop.

To export a subsetting definition from the Software Library:

1. From the **Actions** menu, select **Export**, then select **File from Software Library**.
2. In the Export Subset Definition from Software Library pop-up that appears, select the desired subsetting definition and click **Export**.
3. In the File Download pop-up that appears, click **Save**.
4. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The system converts the subsetting definition into an XML file that now appears at the specified location on your desktop.

After the job runs successfully, the subset template appears in the list of subsets in the table on the Data Subsetting Definitions page.

Creating a Subset Version of a Target Database

After a subset is defined, analyzed, and validated, you can execute the subset operation to create a subset version of the source data.

The procedure assumes the following prerequisites:

- A subsetting definition already exists that contains the rules needed to subset the database.
- You have the requisite privileges to extract the data from the source and create the subset version in a target database. Depending on the subset technique, different levels of file or database privileges may be created. The required privileges include:
  - `EM_ALL_OPERATOR` for Enterprise Manager Cloud Control users
  - `SELECT_CATALOG_ROLE` for database users
  - `SELECT ANY DICTIONARY` privilege for database users
  - DBA privileges on a database for target database users
- Additionally, to perform an in-place delete operation, the DBA user must be granted the `EXECUTE_ANY_TYPE` privilege

To create a subset version of a target database:

1. Create a subset operation by selecting a subsetting definition and associating it with a source database.

   Enterprise Manager validates the subsetting definition against the source database and flags schema differences. Note that this association may be different from the original association that an application developer may have created.
2. Edit the definition to remap the defined schema to a test schema.

You are prompted to connect to a database, whereupon the database is associated with the subsetting definition. This also enables you to remap the vendor-provided schema names to actual schema names in the database.

3. Select one of the various subset creation techniques:

   - Data Pump dump file followed by a Data Pump import
   - In-place delete, in which rows in the specified database not matching the rule conditions are deleted
   - In-transit subset creation or refresh

   Enterprise Manager generates the appropriate response file (that is, SQL script, Data Pump script, or OS script), checks the target system for appropriate privileges to be able proceed with the operation, and estimates the size of the target.

4. After reviewing the analysis, submit the subset process.

   Enterprise Manager executes the subset process and summarizes the results of the execution.

Synchronizing a Subsetting Definition with an Application Data Model

Changes to an ADM, adding referential relationships or deleting tables, for example, can render a subsetting definition stale. The Subsetting Definitions page clearly indicates this condition with a lock icon next to the subset name and an invalid status. Also, most menu items on the **Actions** menu are disabled. To revert the status to valid and unlock the subsetting definition, you have to synchronize the definition with its associated ADM.

1. On the Subsetting Definitions page, select the locked subsetting definition.
2. From the **Actions** menu, select **Synchronize**.
3. Complete the job submission dialog, then click **Submit**.

   When the job completes, the subsetting definition is unlocked and available for use.

Granting Privileges on a Subsetting Definition

You can grant privileges on a subsetting definition that you create so that others can have access. To do so, you must be an Enterprise Manager Administrator with at least Designer privileges on the subsetting definition.

1. From the **Enterprise** menu, select **Quality Management**, then select **Data Subset Definitions**.
2. Select the subsetting definition to which you want to grant privileges.
3. From the **Actions** menu, select **Grant**, then select as follows:

   - **Operator**–to grant Operator privileges on the subsetting definition to selected roles or administrators, which means the grantees can view and copy but not edit the definition.
   - **Designer**–to grant Designer privileges on the subsetting definition to selected roles or administrators, which means the grantees can view and edit the definition.

4. In the dialog that opens, select the type (administrator or role, or both). Search by name, if desired. Make your selections and click **Select**.

   Those selected now have privileges on the subsetting definition.

5. Use the **Revoke** action if you want to deny privileges previously granted.

See "Access Rights to Data Security Objects" on page 2-11 for more information on privileges within the test data management area.

About Integrated Subset and Mask

You can reduce the size of the database simultaneous with masking sensitive data. This serves the dual purpose of obscuring exported production data while greatly reducing hardware costs related to storing large masked production databases for testing.

> **Note:** The Mask in Export feature (also known as At Source masking) works with Oracle Database 11.1 and higher.

The benefits of integrating data masking with subsetting include the following:

- Prepare the test system in a single flow
- Avoid the necessity of maintaining large-size masked databases for test purposes
- Exported data in the form of a dump file can be imported into multiple databases without exposing sensitive data
- Subsetting is enhanced by ability to discard columns containing chunks of large data

You can select one or more ns during subset creation. The masking definitions must be based on the same ADM as the current subsetting definition. At the same time, you can significantly reduce the subset size by defining column rules to set CLOB and BLOB columns to null (or another supported format such as Fixed String, Fixed Number).

You generate a subset in two ways:

- Export Dump–if masking definitions are part of the subset model, mapping tables are created during generation, and the resulting dump contains masked values
- In-Place Delete–subsetting is performed on a cloned copy of the production database; if data masking is part of the subset model, pregenerated masking scripts are executed on the target sequentially

Advantages of integrated subset and mask include the following:

- Sensitive data never leaves the production environment and thus is not exposed (Export Dump option).
- There is no need to temporarily store data in a staging area.
- Exported data can subsequently be imported into multiple environments.
- You can define table rules to export only a subset of data, and can further trim the volume by using column rules to eliminate large vertical columns.
- You can mask the same data in different ways and import into different test databases.

- You can use the provisioning framework to create multiple copies of trimmed down, referentially intact databases containing no sensitive data (in-place delete), or import a dump file into multiple databases (export dump).

The section "Creating a Data Subsetting Definition" on page 4-1 includes instructions for combining data subsetting and data masking within the process of creating a data subsetting definition. See Chapter 3, "Data Masking,"for information on data masking and creating a n.

Integrated Subset and Mask Scenarios

The scenarios described below assume that an Application Data Model (ADM) exists for a production (or test) database in which sensitive column details are captured. The steps outlined are at a high level. See "Masking with an Application Data Model and Workloads" on page 3-17 for details on creating a masking definition; see "Creating a Data Subsetting Definition" on page 4-1 for details on creating and editing a subsetting definition.

Consider the following scenarios:

- Mask and Export Production Data
- Mask and Delete Operation on a Test Database
- Mask Sensitive Data and Export a Subset of a Production Database
- Perform Subset, Mask, and Delete Operations on a Test Database
- Apply Column Mask Rules
- Export a Subsetting Definition That Uses Integrated Subset and Mask
- Import a Subsetting Definition That Uses Integrated Subset and Mask
- Import a Subset Dump
- Save Subset Script Bundle

Mask and Export Production Data

As the Security Administrator, you want to create copies of the production database by exporting the data with masked values; that is, the export dump will have only masked values and no sensitive data.

1. Create a masking definition. Implies the following:
   a. Select an appropriate ADM.
   b. Search and select sensitive columns (includes dependent columns and recommended masking formats).
   c. Review suggested formats and edit as necessary.
   d. Save the results.
2. Create a subsetting definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
3. Edit the subsetting definition.

   On the **Data Masking** tab, search for and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.

4. Generate the subset using the Export option.

Summarizing the outcome:

- Generates and executes a script to create a mapping table and a mapping function. Also creates a table to map the column(s) to the respective mapping function.
- Copies subsetting and masking scripts to the target database.
- Generates an export dump of production data, replacing sensitive data with masked values using the mapping function.

Mask and Delete Operation on a Test Database

As the Security Administrator, you want to create a usable test database by masking sensitive information. The resulting database will have only masked values and no sensitive data.

1. Create a masking definition on a cloned database. Implies the following:
   a. Select an appropriate ADM.
   b. Search and select sensitive columns (includes dependent columns and recommended masking formats).
   c. Review suggested formats and edit as necessary.
   d. Save.
2. Create a subsetting definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
3. Edit the subsetting definition.

   On the **Data Masking** tab, search and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
4. Generate the subset using the In-Place Delete option.

Summarizing the outcome:

- Copies subsetting and masking scripts to the target database.
- Performs data subsetting based on subset rules, if specified.
- Sequentially executes the pregenerated data masking scripts on the target database.
- Creates a masked copy of the production database for use in testing.

Mask Sensitive Data and Export a Subset of a Production Database

As the Security Administrator, you want to create copies of the production database by exporting a subset of production data with masked values.

1. Create a masking definition. Implies the following:
   a. Select an appropriate ADM.
   b. Search and select sensitive columns (includes dependent columns and recommended masking formats).
   c. Review suggested formats and edit as necessary.
   d. Save.

2. Create a subsetting definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
3. Edit the subsetting definition.
   a. Define table rules, resulting in space estimates.
   b. On the **Data Masking** tab, search and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
4. Generate the subset using the Export option.

Summarizing the outcome:

- Generates and executes a script to create a mapping table and a mapping function. Also creates a table to map the column(s) to the respective mapping function.
- Copies subsetting and masking scripts to the target database.
- Generates an export dump of production data, replacing sensitive data with masked values using the mapping function.

Perform Subset, Mask, and Delete Operations on a Test Database

As the Security Administrator, you want to create a usable test database by masking sensitive information. On import, the database will have only masked values and no sensitive data.

1. Create a masking definition. Implies the following:
   a. Select an appropriate ADM.
   b. Search and select sensitive columns (includes dependent columns and recommended masking formats).
   c. Review suggested formats and edit as necessary.
   d. Save.
2. Create a subsetting definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
3. Edit the subsetting definition.
   a. Define table rules, resulting in space estimates.
   b. On the **Data Masking** tab, search and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
4. Generate the subset using the In-Place Delete option.

Summarizing the outcome:

- Copies subsetting and masking scripts to the target database.
- Performs data subsetting based on subset rules, if specified.
- Following subset completion, sequentially executes the pregenerated data masking scripts on the target database.

- Applies masking definitions and subsetting rules, resulting in a masked database of reduced size.

Apply Column Mask Rules

As the Security Administrator, you want to create a targeted subset by selecting large-sized columns and setting them to null or a fixed value. Table rules can also be used to further reduce database size. Impact of size reduction is immediately visible and applied to the final subset.

1. Create a subsetting definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
2. Edit the subsetting definition.
   a. Click the **Table Rules** tab and select from existing options, if desired.
   b. Click the **Column Rules** tab, then click **Create**.
   c. Specify filtering criteria to search for large-sized columns and select the desired columns in the results table.
   d. Click **Manage Masking Formats** and select a format from the drop-down list. Enter a value if appropriate to the selection.
   e. Click **OK** and review the updated space estimates.
3. Generate the subset, using either the Export or In-Place Delete option.

Summarizing the outcome:

- Generates an export dump/subset of production data.
- Column rules are applied on the target database.
- If table rules were also applied, the resulting subset reflects the combined effect of table and column rules.

Export a Subsetting Definition That Uses Integrated Subset and Mask

As the Security Administrator, you want to export a subsetting definition for reuse.

1. Create a subsetting definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
2. Edit the subsetting definition.
   a. Create rules to compute space estimates.
   b. On the **Data Masking** tab, search and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
3. Select the subsetting definition on the Subset home page and export it.

The subsetting definition is saved on the client machine as an XML file that potentially contains the following:

- Information on selected applications
- Rules and rule parameters

- Selected masking definitions
- Columns to set to null
- Pre- and post-scripts

Had column rules been used, they would replace the masking definitions in the list.

Import a Subsetting Definition That Uses Integrated Subset and Mask

As the Security Administrator, you want to import a subsetting definition XML file to create replicas of the subsetting definition previously exported.

1. Import a subsetting definition.
2. Select an exported XML template that contains exported masking definitions. System validation:
   - Checks for overlapping columns that use multiple masking definitions.
   - Ensures that the masking definition specified is part of the same ADM as the current subset model.
3. Submit the job to create the subset model.

Summarizing the outcome:

- Creates a subsetting definition model
- Applies specified rules and calculates space estimates
- Remembers masking definitions that were part of the XML

Import a Subset Dump

As the Security Administrator, you want to import a subset dump, which might contain either or both of the following:

- A masked version of a production database
- A subset version of a production database

Note that this example assumes a previous export dump.

1. On the subset home page, select **Import Subset Dump** from the **Actions** menu.
2. Provide credentials, a dump name, and select the dump location.
3. Provide the import type, tablespace options, and log file location details.
4. Schedule the job and submit.

The job reads the dump files and loads the data into the selected target database.

Save Subset Script Bundle

As the Security Administrator, you want to save a subset script bundle so that it can be executed on a target database external to Enterprise Manager.

This example presupposes the existence of a subset model that has required table rules and masking definitions.

1. On the subset home page, from the **Actions** menu, select **Generate**, then select **Subset**.
2. Complete the mode page as follows:

   a. Indicate the method of subset creation.
   b. Specify which credentials to use.
   c. Provide rule parameters as appropriate.
   d. Click **Continue**.

3. Complete the parameters page as follows:
   a. Select the location where to save the subset export.
   b. If the subset is to be stored externally, click the check box and select the location.
   c. Specify an export file name. Note that you can use the % wildcard.
   d. Specify the maximum file size and number of threads.
   e. Indicate whether to generate a log file and specify a log file name.
   f. Click **Continue**.
4. Note the progress of the script file generation. When complete, click **Download**.
5. Specify where to save the `SubsetBundle.zip` file.

Index