

## **Oracle® Business Intelligence Applications**

Administrator's Guide

11g Release 1 (11.1.1.8.1)

**E51483-03**

December 2014

Provides topics of interest to system administrators, including customization, multi-language support, localizing deployments, the Endeca Discovery option, and using Oracle GoldenGate.

Oracle Business Intelligence Applications Administrator's Guide, 11g Release 1 (11.1.1.8.1)

E51483-03

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Primary Author: Dan Hilldale

Contributors: Oracle Business Intelligence development, product management, and quality assurance teams.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

---

---

# Contents

<b>Preface</b> .....	ix
Audience .....	ix
Documentation Accessibility .....	ix
Related Documentation .....	ix
Conventions .....	x
<b>What's New for Oracle Business Intelligence Applications Administration</b> .....	xi
New Features for Oracle BI Applications System Administrators .....	xi
<b>1 Customizing the Oracle Business Analytics Warehouse</b>	
Overview of Customization in Oracle Business Intelligence Applications .....	1-1
What is Customization in Oracle Business Intelligence Applications? .....	1-1
About the Customization Process .....	1-2
About the Impact of Patch Installation on Customizations .....	1-3
Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables .....	1-3
About Extending Mappings .....	1-4
Typical Steps to Extend Mappings in the Oracle Business Analytics Warehouse .....	1-4
Example of Extending the Oracle Business Analytics Warehouse .....	1-6
Overview to the Example .....	1-7
Example of Extracting Data from an Oracle EBS 11.5.10 Data Package	
Source into the ODI Staging Area .....	1-7
Example of Loading Data from the Staging Area into an Existing Target Table .....	1-12
Tips for Modifying the SQ_BCI_ Interface .....	1-14
Other Types of Customizations Requiring Special Handling .....	1-14
How to Modify Category 2 SCD Behavior .....	1-15
How to Add A Dimension to an Existing Fact .....	1-16
Create a Custom Dimension Datastore and Tasks .....	1-16
Customize Fact Datastores and Tasks .....	1-18
How to Add a DATE_WID column to a Fact .....	1-19
Category 2 Customizations: Adding Additional Tables .....	1-20
About Creating New Tables .....	1-20
About the Main Required Columns .....	1-21
About the DATASOURCE_NUM_ID Column .....	1-21
Additional Information About Customizing .....	1-21
About the Update Strategy .....	1-21

About Indices and Naming Conventions .....	1-21
Adding a New Dimension to the Oracle Business Analytics Warehouse .....	1-22
Adding a New Fact Table to the Oracle Business Analytics Warehouse .....	1-22
Adding a New Dimension Table for a New Fact Table in the Oracle Business Analytics Warehouse .....	1-24
Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table .....	1-25
How to Add New Data as a Whole Row Into a Standard Dimension Table .....	1-25
Configuring Extracts .....	1-25
Extracting Additional Data .....	1-25
Extracting New Data Using an Existing Source Table .....	1-26
Extracting Data from a New Source Table .....	1-26
Setting Up the Delimiter for a Source File .....	1-26
Configuring Loads .....	1-27
Filtering and Deleting Records .....	1-27
About Primary Extract and Delete Mappings Process .....	1-27
About Working with Primary Extract and Delete Mappings .....	1-29
Deleting the Configuration for Source-Archived Records .....	1-29
Configuring Slowly Changing Dimensions .....	1-29
About Identifying Historically Significant Attributes .....	1-29
About the Extract View .....	1-30
Type I and Type II Slowly Changing Dimensions .....	1-30
Type I Slowly Changing Dimension .....	1-30
Type II Slowly Changing Dimension .....	1-31
Effective Dates .....	1-31
Customizing Stored Lookups and Adding Indexes .....	1-32
About Stored Lookups .....	1-32
About Resolving Dimension Keys .....	1-32
Resolving the Dimension Key Using Lookup .....	1-32
About Domain Values .....	1-33
About the Domain Value Conversion Process .....	1-33
About the Importance of Domain Values .....	1-33
Hire Count .....	1-34
Re-hires Ratio .....	1-34
New Hire Count .....	1-34
Newly Separated Veterans - New Hires .....	1-34
Other Protected Veterans - New Hires .....	1-34
Special Disabled Veteran Head count - New Hires .....	1-34
Vietnam Era Veteran Head count - New Hires .....	1-34
About Extending the Domain Value Set .....	1-34
Configuring the Domain Value Set with CSV Worksheet Files .....	1-35
Configuring the Domain Value Set Using ODI Designer .....	1-36
How to add an index to an existing fact or dimension table .....	1-36
Trimming the RPD .....	1-37
Extracting Projects From Full RPD .....	1-37
About Resolving Dimension Keys .....	1-39
Resolving the Dimension Key Using Lookup .....	1-39
Merging Repositories .....	1-40

## 2 About Multi-Language Support

Introduction to Multi-Language Support .....	2-1
About Pseudo-Translations .....	2-2
About Oracle BI Applications Domains .....	2-2
About Dimension Translation Tables .....	2-4

## 3 Localizing Oracle Business Intelligence Deployments

Process of Maintaining Translation Tables for Oracle BI EE .....	3-1
Upgrading Oracle Business Intelligence Seed Data for Non-English Locales .....	3-2
Verify the Translation Table (W_LOCALIZED_STRING_G) and Corresponding Indexes .....	3-2
Importing Locale Seed Data Into The Translation Table (W_LOCALIZED_STRING_G) .....	3-3
Externalizing Customer Metadata Strings .....	3-4
Adding Custom Translations to the W_LOCALIZED_STRING_G Table .....	3-4
Adding String Translations for Analytics Metadata .....	3-5
About Translating Presentation Services Strings .....	3-5
Changing the Default Currency in Analytics Applications .....	3-6

## 4 Oracle Business Analytics Warehouse Naming Conventions

Naming Conventions for Oracle Business Analytics Warehouse Tables .....	4-1
Table Types for Oracle Business Analytics Warehouse .....	4-2
Aggregate Tables in Oracle Business Analytics Warehouse .....	4-4
Dimension Class Tables in Oracle Business Analytics Warehouse .....	4-4
Dimension Tables in Oracle Business Analytics Warehouse .....	4-4
Dimension Tables With Business Role-Based Flags .....	4-4
Fact Tables in Oracle Business Analytics Warehouse .....	4-5
Helper Tables in Oracle Business Analytics Warehouse .....	4-5
Hierarchy Tables in Oracle Business Analytics Warehouse .....	4-5
Mini-Dimension Tables in Oracle Business Analytics Warehouse .....	4-5
Staging Tables in Oracle Business Analytics Warehouse .....	4-6
Translation Tables in Oracle Business Analytics Warehouse .....	4-6
Internal Tables in Oracle Business Analytics Warehouse .....	4-6
Standard Column Prefixes in Oracle Business Analytics Warehouse .....	4-7
Standard Column Suffixes in Oracle Business Analytics Warehouse .....	4-7
System Columns in Oracle Business Analytics Warehouse Tables .....	4-8
Multi-Currency Support for System Columns .....	4-9
Oracle Business Analytics Warehouse Primary Data Values .....	4-10
About Multi-Language Support in the Oracle Business Analytics Warehouse .....	4-10
Oracle Business Analytics Warehouse Currency Preferences .....	4-10

## 5 Oracle Business Intelligence Endeca Discovery Option

Introduction .....	5-1
Setting Up Endeca Integration with Oracle BI Applications .....	5-1
Tasks for Setting Up Oracle Endeca with Oracle BI Applications .....	5-2
Setup Step: Install Oracle Endeca Server .....	5-2
Setup Step: Install Endeca 3.1 Studio .....	5-3

Setup Step: Set Up ODI Connectivity for Oracle BI and Endeca .....	5-3
Setup Step: Load Data .....	5-7
Setup Step: Create a New Endeca Application .....	5-8
ODI Package, Interface, and Procedure Design Overview .....	5-9
Deploying Sample Applications .....	5-10
Applying the Custom BI Applications Security Manager .....	5-13
Troubleshooting .....	5-17

## 6 Researching Data Lineage

Introduction .....	6-1
Setting Up Data Lineage .....	6-1
Tasks for Setting Up Data Lineage Dashboards .....	6-2
Setup Step: Configure ODI Topology and Load Plan .....	6-2
Setup Step: Configure the WebLogic Server Heap Size .....	6-8
Setup Step: Disable Fusion Step (Optional) .....	6-9
Setup Step: Create the Data Lineage Warehouse Tables .....	6-10
Tasks for Loading and Refreshing Data Lineage Dashboards .....	6-11
Extracting Oracle Business Intelligence Metadata Using Catalog Manager and Administration Tool .....	6-11
Executing and Monitoring the Data Lineage Load Plan .....	6-14
Performing Analysis with Data Lineage Dashboards .....	6-15
Overview of the Oracle BI Applications Data Lineage Dashboard .....	6-15
Analyzing Data Lineage for Oracle Business Intelligence Applications .....	6-16

## 7 Administering Oracle GoldenGate and Source Dependent Schemas

Introduction .....	7-1
Source Dependent Schema Architecture .....	7-1
Setting Up Oracle GoldenGate and Source Dependent Schemas .....	7-2
Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema .....	7-3
Setup Step: Configure Source and Target Database .....	7-3
Setup Step: Install Oracle GoldenGate on Source and Target Systems .....	7-5
Setup Step: Configure BI Applications Configuration Manager and Oracle Data Integrator to Support the Source Dependent Schema .....	7-8
Setup Step: Generate, Deploy, and Populate the Source Dependent Schema Tables on Target Database .....	7-9
Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines .....	7-12
Generate Oracle GoldenGate Parameter Files .....	7-13
Configure the Source System .....	7-15
Configure the Target System .....	7-17
Setup Step: Start Oracle GoldenGate on Source and Target Machines .....	7-18
Customization .....	7-18
Adding a Non-Custom Source Column to an Existing ETL Task .....	7-19
Adding a Custom Source Column to an Existing ETL Task .....	7-19
Adding a Non-Custom Source Table to an Existing ETL Task .....	7-19
Creating a Custom ETL Task .....	7-19
Patching .....	7-19

Troubleshooting .....	7-20
Create the SDS Tables .....	7-20
Using the DML Option to Perform an Initial Load .....	7-21
Create SDS Indexes and Analyze the SDS schema .....	7-22

## **Index**





---

---

# Preface

Oracle Business Intelligence Applications is comprehensive suite of prebuilt solutions that deliver pervasive intelligence across an organization, empowering users at all levels - from front line operational users to senior management - with the key information they need to maximize effectiveness. Intuitive and role-based, these solutions transform and integrate data from a range of enterprise sources and corporate data warehouses into actionable insight that enables more effective actions, decisions, and processes.

Oracle BI Applications is built on Oracle Business Intelligence Suite Enterprise Edition (Oracle BI EE), a comprehensive set of enterprise business intelligence tools and infrastructure, including a scalable and efficient query and analysis server, an ad-hoc query and analysis tool, interactive dashboards, proactive intelligence and alerts, and an enterprise reporting engine.

## Audience

This document is intended for system administrators and ETL team members who are responsible for managing Oracle BI Applications. It contains information about ETL customization, domains and localization, Oracle Business Analytics Warehouse naming conventions, and system administration tasks, including setting up and using Oracle GoldenGate and Source-Dependent Schemas to support ETL performance.

## 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 Documentation

See the Oracle Business Intelligence Applications documentation library at [http://docs.oracle.com/cd/E51479\\_01/index.htm](http://docs.oracle.com/cd/E51479_01/index.htm) for a list of related Oracle Business Intelligence Applications documents.

## Conventions

The following text conventions are used in this document:

<b>Convention</b>	<b>Meaning</b>
<b>boldface</b>	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	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.

---

---

# What's New for Oracle Business Intelligence Applications Administration

This preface describes features in Oracle Business Intelligence Applications that relate to administration.

## New Features for Oracle BI Applications System Administrators

### **Endeca Integration**

Oracle BI Applications can optionally leverage Oracle Endeca Information Discovery. Oracle Endeca Information Discovery is an enterprise discovery platform for the analysis of information from both structured and unstructured sources, providing the ability to visualize data in many different ways, including tag clouds.

Oracle BI Applications integration of Oracle Endeca Information Discovery leverages its ability to analyze unstructured data. Oracle Endeca Information Discovery includes ODI maps that can be run to load Endeca applications, covers more than 100 Oracle BI Application subject areas, and grants some Oracle BI Applications access to unstructured text fields from source systems, which enables users to examine their structured BI data in new ways.



---

---

# Customizing the Oracle Business Analytics Warehouse

This chapter describes concepts and techniques for customizing the ETL functionality in Oracle Business Intelligence Applications. In addition, it describes the process used to trim the Oracle BI Applications metadata repository.

This chapter contains the following topics:

- [Section 1.1, "Overview of Customization in Oracle Business Intelligence Applications"](#)
- [Section 1.2, "Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables"](#)
- [Section 1.3, "Category 2 Customizations: Adding Additional Tables"](#)
- [Section 1.4, "Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table"](#)
- [Section 1.5, "Customizing Stored Lookups and Adding Indexes"](#)
- [Section 1.6, "Trimming the RPD"](#)

## 1.1 Overview of Customization in Oracle Business Intelligence Applications

This section provides an overview of customization in Oracle Business Intelligence Applications, and contains the following topics:

- [Section 1.1.1, "What is Customization in Oracle Business Intelligence Applications?"](#)
- [Section 1.1.3, "About the Impact of Patch Installation on Customizations"](#)

### 1.1.1 What is Customization in Oracle Business Intelligence Applications?

In Oracle Business Intelligence Applications, customization is defined as changing the preconfigured behavior to enable you to analyze new information in your business intelligence dashboards. For example, you might want to add a column to a dashboard by extracting data from the field `HZ_CUST_ACCOUNTS.ATTRIBUTE1` and storing it in the Oracle Business Analytics Warehouse in the `X_ACCOUNT_LOG` field.

The type of data source that you have determines the type of customization that you can do. Data sources can be one of the following types:

- Packaged applications (for example, Oracle Fusion or EBS), which use prepackaged adapters.

- Non-packaged data sources, which use the Universal adapter.

Customizations are grouped into the following categories:

- **Category 1.** In a Category 1 customization, you add additional columns from source systems that have pre-packaged adapters and load the data into existing Oracle Business Analytics Warehouse tables. For more information about performing Category 1 customizations, see [Section 1.2, "Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables"](#).
- **Category 2.** In a Category 2 customization, you use pre-packaged adapters to add new fact or dimension tables to the Oracle Business Analytics Warehouse. Category 2 customizations normally require that you build new SDE and SIL mappings. For more information about performing Category 2 customizations, see [Section 1.3, "Category 2 Customizations: Adding Additional Tables"](#).
- **Category 3.** In a Category 3 customization, you use the Universal adapter to load data from sources that do not have pre-packaged adapters. For more information about performing Category 3 customizations, see [Section 1.4, "Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table"](#).

The figure below summarizes the category of customization that you can perform for each type of data source and type of modification.

**Figure 1–1 Supported customizations based on data source**

		Data Warehouse Modifications		
		Add Additional Column to Existing Fact or Dimension Table	Add Additional Rows to Existing Fact or Dimension Tables	Add New Fact or Dimension Tables
Data Sources	Packaged Application (Uses Prepackaged Adaptor)	Category 1	Configure Filter	Category 2
	Non-Packaged Data (Uses Universal Adaptor)	Category 1	Category 3	Category 2

For detailed information about tables and naming conventions, see *Oracle Business Analytics Warehouse Data Model Reference*.

When you customize ETL Packages and Interfaces, you usually work in the \Oracle BI Applications \Mappings folder in the Projects view in ODI Studio's Designer Navigator.

**Note:** The customization methodology is to make a copy of the ETL task and version both the original and copy while a datastore is simply versioned. These versions allow you to revert functionality if required as well as identify changes that have been introduced through customization, patches or upgrades.

### 1.1.2 About the Customization Process

This chapter explains how to customize your ETL functionality, after you have performed a Business Analysis and Technical Analysis. This chapter does not cover the other typical tasks that you need to perform, as follows:

- Business Analysis - before you start customization, you typically analyze your current BI dashboards to determine the changes you need to support your business or organization.
- Technical Analysis - when you have identified your business requirements, you need to determine the technical changes you need to make, by identifying source tables, staging tables, target tables, and ODI Packages and Interfaces that you need to modify.
- RPD Modification - having made the customizations in the ETL functionality, you need to modify your RPD to expose the new data in your dashboards. For more information about RPD modification, refer to the Oracle Business Intelligence Enterprise Edition documentation library.

### 1.1.3 About the Impact of Patch Installation on Customizations

This section explains what you must do to re-apply a customization to an object that has been patched. For example, if you install an Oracle Business Intelligence Applications patch that modifies the Supply Chain and Order Management application, you might need to manually re-apply customizations that you have made to the Supply Chain and Order Management application.

As part of customizing an ETL task (including interfaces and package under a specific task folder), you copy the task folder to be customized, version the original and version the copy. Any patches are applied to the current version of the original task. Leverage ODI's version compare utility to identify the changes introduced by the patch. The copy is also versioned so that any changes introduced can be isolated. Compare any changes with those introduced by the patch and verify there is no conflict, then manually apply the same changes introduced by the patch to the customized ETL tasks. For information about modifying and versioning ETL customizations, refer to [Section 1.2.2, "Typical Steps to Extend Mappings in the Oracle Business Analytics Warehouse"](#).

A patch only installs changed repository objects, not the whole Work Repository. Therefore, you only need to re-apply customizations to mappings that have been changed by the patch. For example, if a patch only modifies the Supply Chain and Order Management application, you only need to manually re-apply customizations that you have made to the Supply Chain and Order Management application. Customizations in other applications are not affected by the patch.

#### Note

All customization steps have you create a 'Custom' adaptor folder where customized ETL tasks are stored. This is not required but is considered a best practice to make identifying customized content easier.

## 1.2 Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables

Category 1 customizations add additional columns from source systems that have pre-packaged adapters and load the data into existing Oracle Business Analytics Warehouse tables.

This section contains the following topics:

- [Section 1.2.1, "About Extending Mappings"](#)
- [Section 1.2.2, "Typical Steps to Extend Mappings in the Oracle Business Analytics Warehouse"](#)

- [Section 1.2.3, "Other Types of Customizations Requiring Special Handling"](#)

## 1.2.1 About Extending Mappings

Category 1 customizations involve extracting additional columns from source systems for which pre-packaged adapters are included (for example, Oracle eBusiness Suite) and loading the data into existing Oracle Business Analytics Warehouse tables. For Category 1 customizations, data can also come from non-packaged sources, but this section assumes that the sources have already been mapped with a Universal adapter and only need to be extended to capture additional columns. (The initial mapping of a Universal adapter is considered a Category 3 customization. For information, see [Section 1.4, "Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table"](#).)

In order to see additional columns in the Oracle Business Analytics Warehouse, the columns must first be passed through the ETL process. The existing mappings and tables are extensible. Oracle Business Intelligence Applications provides a methodology to extend preconfigured mappings to include these additional columns and load the data into existing tables.

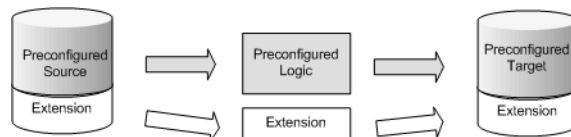
Oracle Business Intelligence Applications recognizes two types of customization: extension and modification. The supported extension logic allows you to add to existing objects. For example, you can extract additional columns from a source, pass them through existing mappings, and populate new columns added to an existing table. Generally, Oracle Business Intelligence Applications does not allow you to modify existing logic or columns. You should not change existing calculations to use different columns, and you should not remap existing columns to be loaded from different sources.

For example, if you want to calculate revenue differently from the existing logic, you should create a new column (for example, X\_REVENUE) and populate it with a custom mapping expression. You can then remap the Oracle Business Intelligence repository to point to the new X\_REVENUE column.

Most datastores have a single placeholder column named X\_CUSTOM. Each ETL task has mapping expressions to populate this column. These serve as templates for customizing ODI datastores and interfaces. When creating new custom columns, follow the naming convention of including the X\_ prefix to help distinguish custom columns.

In the figure below, the preconfigured logic is shaded in gray. You should not modify anything contained within these objects. You should add customizations to existing objects rather than creating new packages and interfaces, which allows them to run parallel to the existing logic.

**Figure 1–2 Preconfigured logic and customizations**



## 1.2.2 Typical Steps to Extend Mappings in the Oracle Business Analytics Warehouse

The most common reason for extending the Oracle Business Analytics Warehouse is to extract existing columns from a source system and map them to an existing Oracle Business Analytics Warehouse table (either fact or dimension). This type of change



typically requires you to extend the interfaces within a SIL package. If the data is coming from a packaged source, then you will also need to extend the interfaces within an appropriate SDE adapter package. If the data is coming from a non-packaged source, then you must use a Universal adapter package. If an appropriate package does not already exist, you will need to create a Universal adapter package with interfaces.

To extend an ODI package in the Oracle Business Analytics Warehouse:

1. Create new SDE and SIL Adaptor folders (do not copy existing Adaptor folder as this will copy all subfolders). Rename folders to include 'Custom' or some other useful identifier in the name, and set Release Tag to match that of the existing Adaptor folder. Do this for both the SDE and SIL folders.
  - a. Right-click the Mappings folder and select New Sub-Folder.
  - b. Set Name as CUSTOM\_<Original Folder Name>. For example, CUSTOM\_SDE\_ORA11510\_Adaptor, CUSTOM\_SILOS represent custom SDE and SIL folders.
  - c. Click the Connect Navigator button in the Designer tab.
  - d. Select Edit Release Tags.
  - e. Select the release tag that corresponds to your source. For example, EBS\_11\_5\_10.
  - f. Select the custom SDE folder you created and add it to the release tag.
  - g. Click Next.
  - h. Click Finish.
  - i. Repeat the above steps for the CUSTOM\_SILOS folder, associating it with the BIA\_11 Release Tag.
2. Enable versioning for the preconfigured Task Folder to be customized. The version comment should indicate this is the base version of the task. Subsequent patches applied to this task in the future would require increasing the version in the comment so that it can be compared to the original task to identify any changes.
  - a. Right-click the Task folder and select Version > Create Version.
  - b. Accept the default version number, 1.0.0.0.
  - c. Add a description indicating that this is the original version of this task.
3. Duplicate the Task folder to be customized by copying it. Cut and paste the copied task folder to the Custom adaptor, and rename it to remove the 'Copy of...' prefix.
4. Using the same method as in step 2, enable versioning of copied Task folder. The version comment should indicate this is the original version. This versioning enables comparison of the customized task to a copy of the original version to determine all changes that have been introduced.
5. Create another version of the copied task. The version comment should indicate this is the customized version. Use the same steps as above.
6. Version the Model that the datastore to be customized exists in, for example, Oracle BI Applications. Submodels and datastores cannot be versioned. The version comment should indicate this is the base or original version.
7. Create a new version of the model, with a version comment indicating that this is where customizations are introduced. The models can now be compared to show differences. If the model ever needs to be patched, the model should be versioned

again so that the patched version can be compared to the custom and original version.

8. Apply customizations to the datastore and task. Customizations should be additive as much as possible rather than overwriting existing content. For example, if you don't like the way a particular column is calculated, add a new custom column and map it in the way you prefer. In the RPD, have the logical column point to this new custom column rather than the original column.
9. Prior to generating scenarios, ensure the 'Scenario Naming Convention' User Parameter has a value of %FOLDER\_NAME(2)%\_%OBJECT\_NAME%
10. Delete the scenario associated with the original task and generate a new scenario for the custom task using the same scenario name as the original. ODI enforces unique scenario names, so you should use the same scenario name for the custom task so that the load plan executes this ETL task rather than the original ETL task.
  - a. Delete the out-of-the-box scenario. Navigate to the original Task folder - Packages - <Package Name> - Scenarios - <Scenario Name>. Make a note of the Scenario Name, then right-click the scenario and select Delete.
  - b. Generate a scenario for the custom task. Navigate to the custom Task folder - Packages - <Package Name>, then right-click the package and select Generate Scenario. Use the original scenario name as noted.
  - c. Check the Generate scenario as if all underlying objects are materialized checkbox.
  - d. Click OK.
  - e. Select Use All from the Startup Parameters drop-down.
  - f. Click OK.

When you execute the load plan, it now executes the custom task rather than the original task. In the future if you make changes to any of the interfaces or the package, you can either regenerate the existing scenario or generate a new scenario. Unless you need separate scenarios, it is recommended that you regenerate the existing scenario. To do this, right-click the scenario and select Regenerate.

11. Generate the Load Plan.
12. Execute the Load Plan.

### 1.2.3 Other Types of Customizations Requiring Special Handling

This section contains the following topics:

- [Section 1.2.3.1, "How to Modify Category 2 SCD Behavior"](#)
- [Section 1.2.3.2, "How to Add A Dimension to an Existing Fact"](#)
- [Section 1.2.3.3, "How to Add a DATE\\_WID column to a Fact"](#)

#### 1.2.3.1 How to Modify Category 2 SCD Behavior

The BI Applications ETL process supports Type I and Type II slowly changing dimension behavior. Some dimensions are enabled only for Type I behavior while other dimensions are enabled to also support Type II behavior. Of those dimensions that support Type II behavior, different dimension attributes have different Slowly Changing behavior including some attributes being treated as Type I.

To enable or disable Type II behavior associated with a dimension:

**Note:** Modifying the Type-II tracking logic is the only change that you should make to shipped logic.

To modify a Category 2 SCD Trigger:

1. In ODI Designer, modify the dimension datastore.
  - a. In the Models view, expand the 'Oracle BI Applications' folder, Oracle BI Applications (Model), and Dimension (Submodel).
  - b. Double-click the Dimension table.
  - c. In the Definition tab, change the OLAP type value to either Dimension (only supports Type I changes) or Slowly Changing Dimension (supports Type II changes).
2. Modify the SIL Dimension Task.
  - a. Navigate to the SIL task that populates this dimension.
  - b. Double-click the 'Main' interface.
  - c. In the Flow subtab, select the 'Target (ORACLE\_BI\_APPLICATIONS)' window.
  - d. If the Property Window is not visible, open it by clicking the Menu Options View – Property Inspector.
  - e. Change the IKM Selector value to 'IKM BIAPPS Oracle Slowly Changing Dimension' if enabling Type II behavior or 'IKM BIAPPS Oracle Incremental Update' if removing Type II behavior.
  - f. Regenerate the scenario.

The following describes how to modify which columns are treated as Type I or Type II in a dimension that is configured to support Type II behavior. If a dimension is configured to support only Type I behavior, the following changes will have no effect as all columns are treated as Type I.

To enable or disable Type II behavior associated with a dimension:

1. In ODI Designer, modify the dimension datastore. In the Models view, expand the 'Oracle BI Applications' folder, Oracle BI Applications (Model), Dimension (Submodel), and Columns.
2. Double-click the column whose SCD behavior you want to change.
3. In the Description subtab's 'Slowly Changing Dimensions Behavior' drop-down list, select the column behavior. To implement Type I behavior, select Overwrite on Change. To implement Type II behavior, select Add Row on Change.

If enabling Type II behavior for a custom dimension, be sure to set columns as follows:

- ROW\_WID - Surrogate Key
- INTEGRATION\_ID, DATASOURCE\_NUM\_ID - Natural Key
- CURRENT\_FLG - Current Record Flag
- EFFECTIVE\_FROM\_DT - Starting Timestamp
- EFFECTIVE\_TO\_DT - Ending Timestamp

### 1.2.3.2 How to Add A Dimension to an Existing Fact

This section explains how to add a dimension to an existing fact, adding a dimension and dimension staging datastores as well as associated SDE and SIL processes, which

also requires extending the fact and fact staging tables to reflect the association with the new dimension. This section includes the following topics:

- [Section 1.2.3.2.1, "Create a Custom Dimension Datastore and Tasks"](#)
- [Section 1.2.3.2.2, "Customize Fact Datastores and Tasks"](#)

#### 1.2.3.2.1 Create a Custom Dimension Datastore and Tasks

Create the custom dimension datastores and tasks. Create a WC\_<dimension name>\_D datastore under the 'Oracle BI Applications – Dimension' model. Create a WC\_<dimension name>\_DS datastore under the 'Oracle BI Applications – Dimension Stage' model. Use the WC\_SAMPLE\_DS and WC\_SAMPLE\_D datastores as templates. These datastores include all required system columns. Custom tables should follow the WC\_naming convention to help distinguish from shipped tables.

---

---

**Note:** The specific submodel that a table belongs to drives the table maintenance behavior. For example, tables in the 'Dimension Stage' submodel will always be truncated at each ETL run while tables in the 'Dimension' submodel are truncated only during a Full ETL run. Do not create a 'Custom' submodel to place your datastores as table maintenance will not be implemented properly for tables in such a submodel.

---

---

As described below, a dimension can be defined either in ODI, generating DDL to create the table in the database, or by defining the table in the database and importing the definition into ODI using the BI Applications RKM. If you use the RKM, the imported table is automatically placed in the 'Other' submodel and needs to be moved into the 'Dimension Staging' and 'Dimension' submodels as appropriate. Also, the OLAP type will need to be set for the dimension to 'Dimension' or 'Slowly Changing Dimension' as appropriate.

#### Manually Create the Dimension Tables in ODI

To create the dimension and tasks manually using ODI:

1. In Designer, navigate to Models > Oracle BI Applications (Folder) > Oracle BI Applications (Model) > Dimension Stage (Submodel).
2. Right-click the WC\_SAMPLE\_DS datastore and select Duplicate Selection.
3. Double-click the new datastore and rename it. Name and Resource Name should match the actual table name. Alias can be the same or a more user friendly value.
4. In the Columns subtab, add all columns.
5. Repeat the same steps to create the Dimension Table by copying the WC\_SAMPLE\_D datastore under the Dimensions submodel.
6. For the dimension table, set the OLAP type to either Dimension if this is a Type I dimension or to Slowly Changing Dimension if this is a Type II dimension.

#### Import Custom Dimension Tables into ODI

To import custom dimension tables into ODI:

1. In Designer, navigate to Models > Oracle BI Applications (Folder) and double-click the Oracle BI Applications Model.
2. In the Reverse Engineer subtab, indicate the tables to be imported under the LIST\_OF\_TABLES option. To import multiple tables, provide a comma-separated list.

3. Click the Reverse Engineer button to start a session that imports the table(s) into ODI
4. The Reverse Engineer process places all tables in the Other submodel. Drag and drop W\_%\_DS tables into the Dimension Stage submodel and the W\_%\_D table into the Dimension submodel.
5. Double-click the new dimension datastore and set the OLAP type to either Dimension if this is a Type I dimension or to Slowly Changing Dimension if this is a Type II dimension.

### Create an ODI Sequence for the Custom Dimension

Create an ODI sequence for the custom dimension. A database sequence is used to populate the ROW\_WID column of the dimension. The Generate DDL procedure is used to generate the DDL required to create the database trigger in the database. Use WC\_SAMPLE\_D\_SEQ as a template.

1. In Designer, navigate to Projects > BI Apps Project > Sequences.
2. Right-click the Sequence folder and select New Sequence.
3. Set name to <Dimension Name>\_SEQ.
4. Select the Native sequence radio button.
5. Set the Schema to DW\_BIAPPS11G.
6. Generally, the Native sequence name should match the ODI name unless this causes the name length to exceed 30 characters, in which case, you can shorten the name to meet this limit. This is the name of the database trigger created to populate the ROW\_WID column.
7. Generate the DDL to create the table in the database. **Note:** If you manually created the dimension in ODI, this will generate the DDL to create both the table and sequence. If you imported the dimension into ODI, this will generate the DDL to create the sequence only.

### Create SDE and SIL Tasks

Create SDE and SIL tasks in the Custom SDE and SIL adaptor folders. Use the SDE\_<Product Line Code>\_SampleDimension and SIL\_SampleDimension tasks as a template. These tasks include the logic required to populate the system columns. Finally, generate scenarios for these tasks.

### Add the Load Plan Step

Add Load Plan step to the '3 SDE Dims X\_CUSTOM\_DIM <Product Line Version Code>' Load Plan Component.

1. In Designer, navigate to Load Plans and Scenarios > BIAPPS Load Plan > Load Plan Dev Components > SDE - <Product Line Version Code> and double-click the '3 SDE Dims X\_CUSTOM\_DIM <Product Line Version Code>' Load Plan Component.
2. In the Steps subtab, select the 'X\_CUSTOM\_DIM' step.
3. Click the green '+' symbol near the top right and select Run Scenario Step.
4. Provide the Scenario Name, set the Version as -1, and set the Step Name to match the Task name. Set the Restart Type to 'Restart from failed step.'

#### 1.2.3.2.2 Customize Fact Datastores and Tasks

The Fact related datastores and tasks must be extended to reflect the new dimension. Both the *W\_<Fact Name>\_FS* and *W\_<Fact Name>\_F* datastores must be extended.

To customize fact datastores and tasks:

1. Extend the Fact Staging datastore by adding an ID column that follows the naming convention *X\_<name>\_ID* and datatype VARCHAR2(80).
  - a. The Oracle BI Applications Model should already be versioned.
  - b. Navigate to Models > Oracle BI Applications (Folder) > Oracle BI Applications (Model) > Fact Stage (Submodel) and double-click the Fact Staging Table.
  - c. In the Columns subtab, select the 'X\_CUSTOM' column.
  - d. Click the green '+' symbol to add a column below the X\_CUSTOM column.
2. Extend the Fact datastore by adding a WID column that follows the naming convention *X\_<name>\_WID* and datatype NUMBER(10). Follow the same steps as above to add a column to the fact datastore.
3. Add a foreign key constraint to the fact table that refers to the custom dimension table created previously. The foreign key constraint ensures the Custom SIL task is included in the generated load plan. The Custom SDE task is included in the generated load plan because it populates the staging table that is used as a source for the custom SIL task.
  - a. Drill into the Fact datastore.
  - b. Right-click the Constraints subfolder below the Fact datastore and select New Reference.
  - c. The naming convention is *FK\_<Fact Table>\_<Dimension Table>*. If there are multiple WID columns that need to reference the same dimension table, enumerate each with a numeric suffix, for example, *FK\_WC\_CUSTOM\_F\_WC\_CUSTOM\_D1*. Type must be 'User Reference'.
  - d. Select the Custom Dimension from the Table drop-down list.
  - e. In the Columns subtab, click the green '+' symbol to add a new column.
  - f. For the Foreign Table column, select the custom WID column in the fact table. For the Primary Table column, select the ROW\_WID column in the dimension table.
4. Add a non-unique bitmap index on the *X\_<name>\_WID* column.
  - a. Drill into the Fact datastore.
  - b. Right-click the Constraints subfolder below the Fact datastore and select New Key.
  - c. The naming convention is *<Fact Table>\_F<n>*. Enumerate each of these indexes with a numeric suffix, for example, *WC\_CUSTOM\_F1*.
  - d. Select the Not Unique Index radio button.
  - e. In the Columns subtab, add the WID column using the shuttle button.
  - f. In the Control subtab, check the Defined in the Database and Active check boxes.
  - g. In the Flexfields subtab, set the index type value to QUERY and the bitmap index value to Y.

5. Modify the Fact SDE task. Pass the value from the source table to the custom X\_*<name>*\_ID column in the staging table. In the mapping expression, include any necessary conversion functions to get the data into the VARCHAR2(80) format.
6. Modify the Fact SIL task. Add logic to retrieve the ROW\_WID value from the custom dimension. This is usually done in one of the following ways. There is no significant difference between these two methods:
  - a. Add the dimension as a source in the SQ temp interface. Join on the fact table's ID column and the dimension table's INTEGRATION\_ID column and the fact and dimension DATASOURCE\_NUM\_ID columns. If the dimension is a Type II dimension, include a range join on the fact's canonical date between the dimension's effective dates. Configure the join as a Left Outer Join. Pass the ROW\_WID column as an output.
  - b. Add the dimension as a lookup in the main interface. The Lookup join is on the fact table's ID column and the dimension table's INTEGRATION\_ID column and the fact and dimension DATASOURCE\_NUM\_ID columns. If the dimension is a Type II dimension, include a range join on the fact's canonical date between the dimension's effective dates. Configure the Lookup Type as 'SQL left-outer join in the from clause'.
7. In the mapping expression to populate the custom WID column in the main interface, embed the ROW\_WID column from the dimension table in a function that defaults NULL values to 0. For example, COALESCE(SQ\_W\_AP\_HOLDS\_FS.PURCHASE\_ORG\_WID, 0)

### 1.2.3.3 How to Add a DATE\_WID column to a Fact

This use case is similar to adding a regular Dimension to a fact but in this case, a Date dimension is used. There are several Date related dimension, each representing dates in a different manner (fiscal, enterprise, and so on) and different granularities (day, week, month, etc.).

Joins between a fact and Date dimension table are performed on a Date specific WID column. The Date WID column is a 'smart key' value that represents the date in YYYYMMDD format. There is no need to do a lookup to resolve the ROW\_WID of the Date dimension, rather you pass the Date column through the ETL process and convert it to this format.

Each fact table has exactly one 'canonical' Date specific WID column. This is the primary date used to drive various date-related calculations. There is no particular metadata to identify this column but lookups to effective dated tables will use this column in the ETL and various date-related expressions in the RPD will also use this column. All packaged fact tables have a single canonical date already identified. When creating custom fact tables, one Date WID column should be nominated as the canonical date and consistently used.

Follow the same steps as adding a dimension to a fact with the following changes. There is no need to create a custom SDE as we use the existing Date dimension.

#### Customize Fact Datastores and Tasks

The Fact related datastores and tasks must be extended to reflect the new dimensionality. Both the W\_*<Fact Name>*\_FS and W\_*<Fact Name>*\_F datastores must be extended.

1. Extend the Fact Staging datastore by adding a DT column that follows the naming convention X\_*<name>*\_DT. This column should have the format DATE(7).

2. Extend the Fact datastore by adding both custom DT and DT\_WID columns. These follow the naming convention X\_<name>\_DT and X\_<name>\_DT\_WID.
3. Add a foreign key constraint to the Date dimension or dimensions. If there are multiple WID columns that need to reference the same date dimension table, enumerate each with a numeric suffix.
4. Modify the Fact SDE task. Pass the value from the source table to the custom X\_<name>\_DT column in the staging table. Apply any conversions required to get the data into DATE format.
5. Modify the Fact SIL task. Pass the X\_<name>\_DT value from the staging table to the corresponding column in the fact table. In the mapping expression to populate the custom X\_<name>\_DT\_WID column in the main interface, embed the DT column in a function that calculates the DT\_WID value, defaulting to 0 when the supplied DT value is NULL. For example, `CALCULATE_DT_WID_DFLT(SQ_W_AP_HOLDS_FS.HOLD_DATE, 0)`

## 1.3 Category 2 Customizations: Adding Additional Tables

Category 2 customizations use pre-packaged adapters to add new fact or dimension tables to the Oracle Business Analytics Warehouse.

This section contains the following topics:

- [Section 1.3.1, "About Creating New Tables"](#)
- [Section 1.3.2, "About the DATASOURCE\\_NUM\\_ID Column"](#)
- [Section 1.3.3, "Additional Information About Customizing"](#)
- [Section 1.3.4, "Adding a New Fact Table to the Oracle Business Analytics Warehouse"](#)

### 1.3.1 About Creating New Tables

This section relates to building entirely new tables that will be loaded with data from a source table that is not already extracted from. For example, you might want to create a new Project dimension table. In this case, you create new dimension and staging tables as well as new extract and load ETL mappings.

When creating a new custom table, use the prefix WC\_ to help distinguish custom tables from tables provided by Oracle as well as to avoid naming conflicts in case Oracle later releases a table with a similar name. For example, for your Project dimension you might create a WC\_PROJECT\_DS and a WC\_PROJECT\_D table.

When you create a new dimension or fact table, use the required system columns that are part of each of the Oracle Business Analytics Warehouse tables to maintain consistency and enable you to reference existing table structures. When you create a new table, you need to define the table and indices in ODI Designer Models area first. The destination model for the Oracle Business Analytics Warehouse is 'Oracle BI Applications'.

#### 1.3.1.1 About the Main Required Columns

For custom staging tables, the following columns are required:

- **INTEGRATION\_ID.** Stores the primary key or the unique identifier of a record as in the source table.



- **DATASOURCE\_NUM\_ID.** Stores the data source from which the data is extracted.

For dimension and fact tables, the required columns are the INTEGRATION\_ID and DATASOURCE\_NUM\_ID columns as well as the following:

- **ROW\_WID.** A sequence number generated during the ETL process, which is used as a unique identifier for the Oracle Business Analytics Warehouse.
- **ETL\_PROC\_WID.** Stores the ID of the ETL process information.

### 1.3.2 About the DATASOURCE\_NUM\_ID Column

The tables in the Oracle Business Analytics Warehouse schema have DATASOURCE\_NUM\_ID as part of their unique user key. While the transactional application normally ensures that a primary key is unique, it is possible that a primary key is duplicated between transactional systems. To avoid problems when loading this data into the data warehouse, uniqueness is ensured by including the DATASOURCE\_NUM\_ID as part of the user key. This means that the rows can be loaded in the same data warehouse tables from different sources if this column is given a different value for each data source.

### 1.3.3 Additional Information About Customizing

This section contains additional miscellaneous information about customization in Oracle Business Intelligence Applications

#### 1.3.3.1 About the Update Strategy

For loading new fact and dimension tables, design a custom process on the source side to detect the new and modified records. The SDE process should be designed to load only the changed data (new and modified). If the data is loaded without the incremental process, the data that was previously loaded will be erroneously updated again. For example, the logic in the preconfigured SIL mappings looks up the destination tables based on the INTEGRATION\_ID and DATASOURCE\_NUM\_ID and returns the ROW\_WID if the combination exists, in which case it updates the record. If the lookup returns null, it inserts the record instead. In some cases, last update date(s) stored in target tables are also compared in addition to the columns specified above to determine insert or update. Look at the similar mappings in the preconfigured folder for more details.

#### 1.3.3.2 About Indices and Naming Conventions

Staging tables typically do not require any indices. Use care to determine if indices are required on staging tables. Create indices on all the columns that the ETL will use for dimensions and facts (for example, ROW\_WIDs of Dimensions and Facts, INTEGRATION\_ID and DATASOURCE\_NUM\_ID and flags). Carefully consider which columns or combination of columns filter conditions should exist, and define indices to improve query performance. Inspect the preconfigured objects for guidance. Name all the newly created tables as WC\_. This helps visually isolate the new tables from the preconfigured tables. Keep good documentation of the customizations done; this helps when upgrading your data warehouse. Once the indices are decided upon, they should be registered in the ODI Model (for more information, see [Section 1.5.2, "How to add an index to an existing fact or dimension table"](#)).

### 1.3.4 Adding a New Fact Table to the Oracle Business Analytics Warehouse

Custom tables should follow the WC\_ naming convention to help distinguish from preconfigured tables. Follow this procedure to add a new fact table to the Oracle Business Analytics Warehouse.

To add a new fact table:

1. Create the custom fact datastores and tasks. Create a WC\_<fact name>\_F datastore under the 'Oracle BI Applications – Fact' model. Create a WC\_<fact name>\_FS datastore under the 'Oracle BI Applications – Fact Stage' model. Use the WC\_SAMPLE\_FS and WC\_SAMPLE\_F datastores as templates. These datastores include all required system columns.

Note that the specific submodel that a table belongs to drives the table maintenance behavior. For example, tables in the 'Fact Stage' submodel will always be truncated during each ETL run while tables in the 'Fact' submodel are only truncated during a Full ETL run.

A fact can be defined in ODI either manually, by generating the DDL to create the table in the database or by defining the table in the database and importing the definition into ODI using the BI Apps RKM. If using the RKM, the imported table will automatically be placed in the 'Other' submodel and will need to be moved into the 'Fact Staging' and 'Fact' submodels as appropriate. The OLAP type also needs to be set for the fact table to 'Fact Table'.

To manually create a Fact Table:

- a. In Designer, navigate to Models > Oracle BI Applications (Folder) > Oracle BI Applications (Model) > Fact Stage (Submodel), right-click the WC\_SAMPLE\_FS datastore and select Duplicate Selection.
- b. Double-click the new datastore and rename it. Name and Resource Name should match the actual table name. Alias can be the same or a more user friendly value.
- c. In the Columns subtab, add all columns.
- d. Repeat the same steps to create the Fact Table by copying the WC\_SAMPLE\_F datastore under the 'Facts' submodel.
- e. For the fact table, set the OLAP type to 'Fact Table'
- f. Generate the DDL to create the table in the database.

To import Fact Tables into ODI:

- a. In Designer, navigate to Models > Oracle BI Applications (Folder) and double-click the Oracle BI Applications model.
- b. In the Reverse Engineer subtab, indicate the tables to be imported under the 'LIST\_OF\_TABLES' option. To import multiple tables, provide a comma separated list.
- c. Click Reverse Engineer. A session is started that imports the table or tables into ODI.
- d. The Reverse Engineer process places all tables in the 'Other' submodel. Drag and drop W\_%\_FS tables into the Fact Stage submodel and the W\_%\_F table into the Fact submodel.
- e. Double-click the new fact datastore and set the OLAP type to 'Fact Table'.
- f. Generate the DDL to create the table in the database.

2. Add a foreign key constraint to all dimension tables associated with this fact. The foreign key constraint ensures the Dimension SIL task is included in the generated load plan. The Dimension SDE task will be included in the generated load plan because it populates the staging table that is used as a source for the Dimension SIL task.
  - a. Drill into the Fact datastore.
  - b. Right-click the 'Constraints' subfolder below the Fact datastore and select New Reference. The naming convention is FK\_<Fact Table>\_<Dimension Table>. If there are multiple WID columns that need to reference the same dimension table, enumerate each with a numeric suffix. For example, FK\_WC\_CUSTOM\_F\_WC\_CUSTOM\_D1.
  - c. Set the Type to 'User Reference', select the dimension from the Table drop-down list and, in the Columns subtab, click the green '+' button on the top right to add a new column.
  - d. For the Foreign Table column, select the custom WID column in the fact table. For the Primary Table column, select the ROW\_WID column in the dimension table.
3. Create an SDE and SIL task in the Custom SDE and SIL adaptor folders. Use the SDE\_<Product Line Code>\_SampleFact and SIL\_SampleFact tasks as a template. These tasks include the logic required to populate the system columns.
4. Add Load Plan step to the '3 SDE Facts X\_CUSTOM\_FG <Product Line Version Code>' Load Plan Component.
  - a. In Designer, navigate to Load Plans and Scenarios > BIAPPS Load Plan > Load Plan Dev Components.
  - b. Navigate to SDE - <Product Line Version Code> and double-click the '3 SDE Facts X\_CUSTOM\_FG <Product Line Version Code>' Load Plan Component.
  - c. Select the 'X\_CUSTOM\_FG' step.
  - d. Click the green '+' symbol near the top right and select the 'Run Scenario Step' option.
  - e. Provide the Scenario Name, Version should be -1, Step Name should match the Task name. Set the Restart Type to 'Restart from failed step.'
5. Add a Load Plan step to '3 SIL Facts X\_CUSTOM\_FG' Load Plan Component.
  - a. In Designer, navigate to Load Plans and Scenarios > BIAPPS Load Plan > Load Plan Dev Components.
  - b. Navigate to SIL and double-click the '3 SIL Facts X\_CUSTOM\_FG' Load Plan Component.
  - c. Select the 'X\_CUSTOM\_FG' step.
  - d. Click the green '+' symbol near the top right and select the 'Run Scenario Step' option.
  - e. Provide the Scenario Name, Version should be -1, Step Name should match the Task name. Set the Restart Type to 'Restart from failed step.'

## 1.4 Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table

Category 3 customizations use the Universal adapter to load data from sources that do not have pre-packaged adapters.

This section contains the following topics:

- [Section 1.4.1, "How to Add New Data as a Whole Row Into a Standard Dimension Table"](#)
- [Section 1.4.2, "Configuring Extracts"](#)
- [Section 1.4.3, "Configuring Loads"](#)

### 1.4.1 How to Add New Data as a Whole Row Into a Standard Dimension Table

Follow this procedure to add new data as a whole row into a standard dimension table in the Oracle Business Analytics Warehouse.

To add new data as a whole row into the standard dimension table:

1. Identify and understand the existing structure of staging tables. Refer to *Oracle Business Analytics Warehouse Data Model Reference* for the table structures. Non-system columns can include the null value.
2. Create a custom SDE interface to load the data into the staging table in the custom folder for this purpose. The staging table needs to be populated with incremental data (rows that have been added or changed since the last Refresh ETL process), for performance reasons.
3. Populate the INTEGRATION\_ID column with the unique identifier for the record. The combination of INTEGRATION\_ID and DATASOURCE\_NUM\_ID is unique. Populate the INTEGRATION\_ID column with the unique identifier for the record. The combination of INTEGRATION\_ID and DATASOURCE\_NUM\_ID is unique.
4. After the data is populated in the staging table, use the standard SIL interfaces to populate the dimension target tables.

### 1.4.2 Configuring Extracts

Each application has prepackaged logic to extract particular data from a particular source. This section discusses how to capture all data relevant to your reports and ad hoc queries by addressing what type of records you want and do not want to load into the Oracle Business Analytics Warehouse, and contains the following topics:

- [Section 1.4.2.1, "Extracting Additional Data"](#)
- [Section 1.4.2.2, "Setting Up the Delimiter for a Source File"](#)

#### 1.4.2.1 Extracting Additional Data

You can configure extract mappings and Interfaces in the Oracle Business Analytics Warehouse to accommodate additional source data. For example, if your business divides customer information into separate tables based on region, then you would have to set up the extract interface to include data from these tables.

**1.4.2.1.1 Extracting New Data Using an Existing Source Table** Extract interfaces generally consist of source tables, expressions used in the target columns, and a staging table. If you want to extract new data using the existing interface, you have to modify the extract interface to include the new data by performing the following tasks:

To modify an existing interface to include new data:

1. Modify the existing interface to extract information from the source, and add it to an appropriate extension column.
2. Modify the Expressions in the target table to perform any necessary transformations.
3. Save the changes.
4. Regenerate the scenario.

You have to determine which type of extension column to map the data to in the staging table. After you modified the extract interface, you would also have to modify the corresponding load interfaces (SDE and SIL) to make sure that the extension columns that you added are connected all the way from the staging table to the target data warehouse table.

**1.4.2.1.2 Extracting Data from a New Source Table** Extract interfaces (which have the `SQ_*` naming convention) reside in source-specific folders within the repository. Extract interfaces are used to extract data from the source system. You can configure these extract interfaces to perform the following:

- Extract data from a new source table.
- Set incremental extraction logic.

#### 1.4.2.2 Setting Up the Delimiter for a Source File

When you load data from a Comma Separated Values (CSV) formatted source file, if the data contains a comma character (,), you must enclose the source data with a suitable enclosing character known as a delimiter that does not exist in the source data.

**Note:** Alternatively, you could configure your data extraction program to enclose the data with a suitable enclosing character automatically.

For example, you might have a CSV source data file with the following data:

```
Months, Status
January, February, March, Active
April, May, June, Active
```

If you loaded this data without modification, ODI would load 'January' as the Months value, and 'February' as the Status value. The remaining data for the first record (that is, March, Active) would not be loaded.

To enable ODI to load this data correctly, you might enclose the data in the Months field within the double-quotation mark enclosing character (" ") as follows:

```
Months, Status
"January, February, March", Active
"April, May, June", Active
```

After modification, ODI would load the data correctly. In this example, for the first record ODI would load 'January, February, March' as the Months value, and 'Active' as the Status value.

To set up the delimiter for a source file:

1. Open the CSV file containing the source data.
2. Enclose the data fields with the enclosing character that you have chosen (for example, ").

You must choose an enclosing character that is not present in the source data. Common enclosing characters include single quotation marks (') and double quotation marks (").

3. Save and close the CSV file.
4. In ODI Designer, display the Models view, and expand the Oracle BI Applications folder.

Identify the data stores that are associated with the modified CSV files. The CSV file that you modified might be associated with one or more data stores.

5. In ODI Designer, change the properties for each of these data stores to use the enclosing character, as follows:
  - a. Double-click the data source, to display the DataStore: <Name> dialog.
  - b. Display the Files tab.
  - c. Use the **Text Delimiter** field to specify the enclosing character that you used in step 2 to enclose the data.
  - d. Click OK to save the changes.

You can now load data from the modified CSV file.

### 1.4.3 Configuring Loads

This section explains how to customize the way that Oracle Business Intelligence Applications loads data into the Oracle Business Analytics Warehouse.

#### 1.4.3.1 About Primary Extract and Delete Mappings Process

Before you decide to enable primary extract and delete sessions, it is important to understand their function within the Oracle Business Analytics Warehouse. Primary extract and delete mappings allow your analytics system to determine which records are removed from the source system by comparing primary extract staging tables with the most current Oracle Business Analytics Warehouse table.

The primary extract mappings perform a full extract of the primary keys from the source system. Although many rows are generated from this extract, the data only extracts the Key ID and Source ID information from the source table. The primary extract mappings load these two columns into staging tables that are marked with a \*\_PE suffix.

The figure below provides an example of the beginning of the extract process. It shows the sequence of events over a two day period during which the information in the source table has changed. On day one, the data is extracted from a source table and loaded into the Oracle Business Analytics Warehouse table. On day two, Sales Order number three is deleted and a new sales order is received, creating a disparity between the Sales Order information in the two tables.

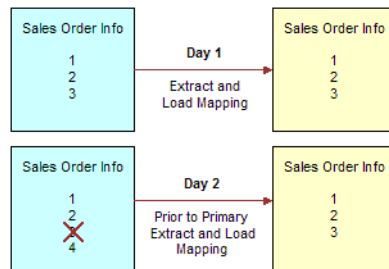
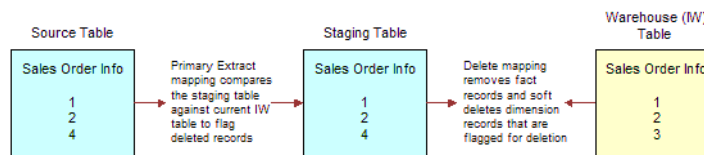
**Figure 1–3 Extract and load mappings**

Figure 1–4 shows the primary extract and delete process that occurs when day two's information is extracted and loaded into the Oracle Business Analytics Warehouse from the source. The initial extract brings record four into the Oracle Business Analytics Warehouse. Then, using a primary extract mapping, the system extracts the Key IDs and the Source IDs from the source table and loads them into a primary extract staging table.

The extract mapping compares the keys in the primary extract staging table with the keys in the most current the Oracle Business Analytics Warehouse table. It looks for records that exist in the Oracle Business Analytics Warehouse but do not exist in the staging table (in the preceding example, record three), and sets the delete flag to Y in the Source Adapter, causing the corresponding record to be marked as deleted.

The extract mapping also looks for any new records that have been added to the source, and which do not already exist in the Oracle Business Analytics Warehouse; in this case, record four. Based on the information in the staging table, Sales Order number three is physically deleted from Oracle Business Analytics Warehouse, as shown in Figure 1–4. When the extract and load mappings run, the new sales order is added to the warehouse.

**Figure 1–4 Primary Extract and Delete Mappings**

### 1.4.3.2 About Working with Primary Extract and Delete Mappings

The primary extract (\*\_Primary) and delete mappings (\*\_IdentifyDelete and \*\_Softdelete) serve a critical role in identifying which records have been physically deleted from the source system. However, there are some instances when you can disable or remove the primary extract and delete mappings, such as when you want to retain records in the Oracle Business Analytics Warehouse that were removed from the source systems' database and archived in a separate database.

Because delete mappings use Source IDs and Key IDs to identify purged data, if you are using multiple source systems, you must modify the SQL Query statement to verify that the proper Source ID is used in the delete mapping. In addition to the primary extract and delete mappings, the configuration of the delete flag in the load mapping also determines how record deletion is handled.

You can manage the extraction and deletion of data in the following ways:

- Deleting the configuration for source-archived records
- Deleting records from a particular source

- Enabling delete and primary-extract sessions
- Configuring the Record Deletion flag
- Configuring the Record Reject flag

**1.4.3.2.1 Deleting the Configuration for Source-Archived Records** Some sources archive records in separate databases and retain only the current information in the main database. If you have enabled the delete mappings, you must reconfigure the delete mappings in the Oracle Business Analytics Warehouse to retain the archived data.

To retain source-archived records in the Oracle Business Analytics Warehouse, make sure the `LAST_ARCHIVE_DATE` parameter value is set properly to reflect your archive date. The delete mappings will not mark the archived records as 'deleted'. For more information about extract and delete mappings, see [Section 1.4.3.2, "About Working with Primary Extract and Delete Mappings"](#).

## 1.5 Customizing Stored Lookups and Adding Indexes

This section contains miscellaneous information that applies to all three categories of customization in Oracle Business Intelligence Applications, and contains the following topics:

- [Section 1.5.1, "About Stored Lookups"](#)
- [Section 1.5.2, "How to add an index to an existing fact or dimension table"](#)

### 1.5.1 About Stored Lookups

This section explains codes lookup and dimension keys.

#### 1.5.1.1 About Resolving Dimension Keys

By default, dimension key resolution is performed by the Oracle Business Analytics Warehouse in the load mapping. The load interface uses prepackaged, reusable lookup transformations to provide pre-packaged dimension key resolution. This section describes how dimension keys are looked up and resolved.

There are two commonly used methods for resolving dimension keys. The first method, which is the primary method used, is to perform a lookup for the dimension key. The second method is to supply the dimension key directly into the fact load mapping.

**1.5.1.1.1 Resolving the Dimension Key Using Lookup** If the dimension key is not provided to the Load Interface through database joins, the load mapping performs the lookup in the dimension table. The load mapping does this using prepackaged Lookup Interfaces. To look up a dimension key, the Load Interface uses the `INTEGRATION_ID`, the `DATASOURCE_NUM_ID`, and the Lookup date, which are described in the table below.

**Table 1–1 Columns Used in the load mapping Dimension Key Lookup**

Port	Description
INTEGRATION ID	Uniquely identifies the dimension entity within its source system. Formed from the transaction in the Source Adapter of the fact table.
DATASOURCE_NUM_ID	Unique identifier of the source system instance.



**Table 1–1 (Cont.) Columns Used in the load mapping Dimension Key Lookup**

Port	Description
Lookup Date	The primary date of the transaction; for example, receipt date, sales date, and so on.

If Type II slowly changing dimensions are enabled, the load mapping uses the unique effective dates for each update of the dimension records. When a dimension key is looked up, it uses the fact's primary or 'canonical' date to resolve the appropriate dimension key. The effective date range gives the effective period for the dimension record. The same entity can have multiple records in the dimension table with different effective periods due to Type II slowly changing dimensions. This effective date range is used to exactly identify a record in its dimension, representing the information in a historically accurate manner.

There are four columns needed for the load interface lookup: INTEGRATION ID, DATASOURCE\_NUM\_ID, and Lookup Date (EFFECTIVE\_FROM\_DT and EFFECTIVE\_TO\_DATE). The lookup outputs the ROW\_WID (the dimension's primary key) to the corresponding fact table's WID column (the fact tables foreign key).

## 1.5.2 How to add an index to an existing fact or dimension table

Dimension and Fact Tables in the Oracle Business Analytics Warehouse use the following two types of index:

- ETL Index  
ETL Indexes are used for Unique/Binary Tree index.
- Query Index  
Query Indexes are used for Non-Unique/Bit Map Index.

To add an index to an existing fact or dimension table:

1. In ODI Designer, display the Models view, and expand the 'Oracle BI Applications' folder.
2. Expand the Fact or Dimension node as appropriate.
3. Expand the Table in which you want to create the index.
4. Right-click on the Constraints node, and select Insert Key to display the Key: New dialog.
5. Display the Description tab.
6. Select the **Alternate Key** radio button, and update the name of the Index in the **Name** field.
7. Display the Column tab.
8. Select the column on which you want to create the index.
9. Display the FlexFields tab.
10. Use the settings to specify the index type, as follows:
  - For 'Query' type indexes (the default), define the index as an 'Alternate Key' for unique indexes and as 'Not Unique Index' for non-unique indexes.
  - For 'ETL' type indexes, clear the check box for the INDEX\_TYPE parameter and set the value to 'ETL'. In addition, set the value of the IS\_BITMAP

parameter to 'N' and define the index as an 'Alternate Key' for unique indexes and as 'Not Unique Index' for non unique indexes.

11. Save the changes.

## 1.6 Trimming the RPD

Oracle BI Applications release 11.1.1.8.1 delivers a full RPD file with projects for all the BI Applications modules and for Operational Planning Applications. This full RPD is deployed to the BI Server. You can trim the RPD so that it includes only the projects that are relevant to your deployment. Although optional, trimming the RPD makes the BI Server startup process faster and also makes patching quicker.

The steps for trimming the RPD depend on the status of your deployment:

- If the RPD has not been customized for your deployment: Extract the projects for the products that your organization has purchased. You do not need to perform a merge. See section [Section 1.6.1, "Extracting Projects From Full RPD,"](#) for instructions.
- If the RPD has been customized for your deployment: Extract the applicable projects from the full (delivered) RPD for release 11.1.1.8.1, and, additionally, merge that RPD with your customized release 11.1.1.8.1 RPD. See [Section 1.6.1, "Extracting Projects From Full RPD,"](#) for instructions.

### 1.6.1 Extracting Projects From Full RPD

Follow this procedure to extract projects from the full RPD. The end result of this process is a trimmed RPD.

To extract from the RPD the projects for the products you have purchased:

1. Open a Command window on the computer where the BI Administration Tool is installed.
2. If you installed Oracle BI EE on Windows, then run `bi-init.cmd` to launch a Command prompt that is initialized to your Oracle instance. This utility is located in:

```
<MiddlewareHome>\instances\instance<n>\bifoundation\OracleBIApplication\coreapp
lication\setup
```

If you installed the BI Administration Tool using the BI Client installer, then run `bi_init.bat` to launch a Command prompt that is initialized your Oracle instance. This file is located in:

```
<Oracle BI Home>\oraclebi\orahome\bifoundation\server\bin
```

3. In a Command prompt window, run `ExtractProjects`, as described below:
  - If you installed Oracle BI EE on Windows, `ExtractProjects.exe` is located in `<Oracle Home for BI>\bifoundation\server\bin`.
  - If you installed BI Administration Tool using the BI Client installer, `ExtractProjects.exe` is located in `<Oracle BI Home>\oraclebi\orahome\bifoundation\server\bin`.

Run one of the following commands:

For extracting a single project:

```
ExtractProjects -B input_rpd -O output_rpd -I "project_name"
```

For extracting multiple projects:

```
ExtractProjects -B input_rpd -O output_rpd -I "project_name1" -I "project_
name2"-I "project_name3" (and so on)
```

where:

`input_rpd` is the name and path of the full (delivered) release 11.1.1.8.1 RPD and from which you want to extract the project or projects (for example, `OracleBIApps.rpd`).

`output_rpd` is the name and path of the RPD you want to create with the extracted projects (for example, `OracleBIAppsTrimmed.rpd`).

`project_name` is the name of the RPD project you want to extract.

You will be prompted to enter the encryption password for the RPD (`input_rpd`).

The list of projects in the 11.1.1.8.1 RPD includes the following:

- Financial Analytics Fusion Edition
- Human Resources Analytics Fusion Edition
- Marketing Analytics Fusion Edition
- Partner Analytics Fusion Edition
- Project Analytics Fusion Edition
- Sales Analytics Fusion Edition
- Supply Chain and Order Management Analytics Fusion Edition
- Student Information Analytics
- Service Analytics
- Price Analytics
- Manufacturing Analytics
- Operational Planning
- DataLineage\_Project

**Note:** The RPD contains projects in addition to those listed above. These projects are included for future content delivery and upgrade support. To determine the BI Applications available in this release, see "System Requirements and Supported Platforms," for Oracle BI Applications release 11.1.1.8.1 at <http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html>.

4. Save and rename the trimmed RPD. Make sure the name identifies this RPD as one that has been trimmed, for example, `OracleBIAppsTrimmed.rpd`.

### 1.6.1.1 About Resolving Dimension Keys

By default, dimension key resolution is performed by the Oracle Business Analytics Warehouse in the load mapping. The load interface uses prepackaged, reusable lookup transformations to provide pre-packaged dimension key resolution. This section describes how dimension keys are looked up and resolved.

There are two commonly used methods for resolving dimension keys. The first method, which is the primary method used, is to perform a lookup for the dimension

key. The second method is to supply the dimension key directly into the fact load mapping.

**1.6.1.1.1 Resolving the Dimension Key Using Lookup** If the dimension key is not provided to the Load Interface through database joins, the load mapping performs the lookup in the dimension table. The load mapping does this using prepackaged Lookup Interfaces. To look up a dimension key, the Load Interface uses the INTEGRATION\_ID, the DATASOURCE\_NUM\_ID, and the Lookup date, which are described in the table below.

**Table 1–2 Columns Used in the load mapping Dimension Key Lookup**

Port	Description
INTEGRATION ID	Uniquely identifies the dimension entity within its source system. Formed from the transaction in the Source Adapter of the fact table.
DATASOURCE_NUM_ID	Unique identifier of the source system instance.
Lookup Date	The primary date of the transaction; for example, receipt date, sales date, and so on.

If Type II slowly changing dimensions are enabled, the load mapping uses the unique effective dates for each update of the dimension records. When a dimension key is looked up, it uses the fact's primary or 'canonical' date to resolve the appropriate dimension key. The effective date range gives the effective period for the dimension record. The same entity can have multiple records in the dimension table with different effective periods due to Type II slowly changing dimensions. This effective date range is used to exactly identify a record in its dimension, representing the information in a historically accurate manner.

There are four columns needed for the load interface lookup: INTEGRATION ID, DATASOURCE\_NUM\_ID, and Lookup Date (EFFECTIVE\_FROM\_DT and EFFECTIVE\_TO\_DATE). The lookup outputs the ROW\_WID (the dimension's primary key) to the corresponding fact table's WID column (the fact tables foreign key).

---

---

# About Multi-Language Support

This chapter provides information about multi-language support in Oracle BI Applications.

This chapter contains the following topics:

- [Section 2.1, "Introduction to Multi-Language Support"](#)
- [Section 2.2, "About Pseudo-Translations"](#)
- [Section 2.3, "About Oracle BI Applications Domains"](#)
- [Section 2.4, "About Dimension Translation Tables"](#)

## 2.1 Introduction to Multi-Language Support

Oracle BI Applications provides multi-language support for metadata level objects exposed in Oracle BI Enterprise Edition dashboards and reports, as well as for data, which enables users to see records translated in their preferred language.

### Configuring Base and Installed Data Warehouse Languages

After installing Oracle BI Applications, you use the Oracle BI Applications Configuration Manager (Configuration Manager) to configure which languages you want to support in the Oracle Business Analytics Warehouse. You must configure one "Base" language, and you can also configure any number of "Installed" languages. Typically, the Base language specified for the data warehouse should match the Base language of the source system. The Installed languages that you specify for the data warehouse do not have to match the languages that are installed in the source system. The data warehouse can have more, fewer, or completely different Installed languages compared to the source system. Note that for languages that match between the transactional system and the data warehouse, the corresponding record is extracted from the transactional system; languages that do not match will have a pseudo-translated record generated.

**Note:** You should only install the languages that you expect to use, because each installed language can significantly increase the number of records stored in the data warehouse and can affect overall database performance.

For information about how to configure data warehouse languages, see *Oracle Fusion Middleware Configuration Guide for Oracle Business Intelligence Applications*.

### Translation Tables

There are two types of translation tables: the Domains translation table and Dimension translation tables. There is a single Domain translation table which holds a translated value in each supported language for a domain. Dimension translation tables are extension tables associated with a given dimension. Depending on certain

characteristics of a translatable attribute, it will be found in either the domain or a dimension translation table.

The user's session language is captured in an Oracle BI Enterprise Edition session variable named `USER_LANGUAGE_CODE`. This is set when users log in from Answers, where they select their preferred language. If users decide to change their preferred language in the middle of a session by using the Administration option to change the current language, this session variable will detect this change. Records returned from a translation table are filtered to those records with a `LANGUAGE_CODE` value that matches this session variable.

## 2.2 About Pseudo-Translations

The ETL process extracts translation records from the source system that correspond to the languages installed in the data warehouse. If a record cannot be found in the source system that corresponds to a language that has been installed in the data warehouse, a pseudo-translated record will be generated. Without a pseudo-translated record, a user that logs in with the missing language as their preferred language will not see any records.

A pseudo-translated record is generated by copying the translation record that corresponds to the data warehouse Base language and flagging it with the missing record's language by populating the `LANGUAGE_CODE` column with the language value. `SRC_LANGUAGE_CODE` stores the language from which the pseudo-translated record was generated; this will always match the data warehouse Base language.

In the future, if a translation record is created in the source system, it will be extracted and the pseudo-translated record will be overwritten to reflect the actual translated value. [Table 2–1](#) provides an example in which "US" is the data warehouse Base language, and "IT" and "SP" are the Installed languages. The source system only had translated records for "US" and "IT" but did not have a translated record for "SP". The "US" and "IT" records are extracted and loaded into the data warehouse. Because there is no translation record in the source system for the "SP" language, a pseudo-translated record is generated by copying the "US" record and flagging `LANGUAGE_CODE` as if it were an "SP" record. The pseudo-translated record can be identified because `SRC_LANGUAGE_CODE` is different from `LANGUAGE_CODE`, matching the Base Language.

**Table 2–1 Example of Pseudo-Translated Record**

<code>INTEGRATION_ID</code>	<code>NAME</code>	<code>LANGUAGE_CODE</code>	<code>SRC_LANGUAGE_CODE</code>
ABC	Executive	US	US
ABC	Executive	IT	IT
ABC	Executive	SP	US

## 2.3 About Oracle BI Applications Domains

A domain refers to the possible, unique values of a table column in a relational database. In transactional systems, domains are often referred to as list of values (LOVs), which present attribute selections in the user's session language. The storage of the transaction is independent of the user's language; and, therefore, the field is stored using a language independent identifier. This identifier is typically a character code but can also be a numeric ID. The LOV or domain is then based on an ID-value

pair, referred to as a member, and the LOV presents the values in the user's session language. At run time, the IDs are resolved to the value for the user's session language.

In the Oracle Business Analytics Warehouse, the number of unique values in any particular domain is relatively small and can have a low cardinality relative to the dimension it is associated with. For example, the Person dimension may have the domain 'Gender' associated with. The dimension may have millions of records, but the domain will generally have two or three members (M, F and possibly U). In the Oracle Business Analytics Warehouse, the Gender Code is stored in the Person dimension which acts as a foreign key to the Domains Translation table which stores the translated values. When a query is run, the user-friendly text associated with the code value is returned in the user's session language.

Depending on certain properties associated with a domain, domains can be configured in the Configuration Manager. In addition to serving as a mechanism for supporting translations, domains can be used to conform disparate source data into a common set of data.

### Data Model

Oracle BI Applications domains are associated with dimensions as fields in the dimension table that follow the `_%CODE` naming convention. For example, the Person dimension `W_PARTY_PER_D` would store the Gender domain in the `GENDER_CODE` column.

Oracle BI Applications domains are stored in the domain translation table `W_DOMAIN_MEMBER_LKP_TL`. This table stores the translated values for each domain member code. Translated values are usually either a Name or a Description value which are stored in the `NAME` and `DESCR` columns of this table. The `DOMAIN_MEMBER_CODE` column acts as a key column when joining with the `_%CODE` column in the dimension table. As domains come from various systems, a `DATASOURCE_NUM_ID` column is used to identify which system the translated value comes from and is used as part of the join key with the dimension table. A `LANGUAGE_CODE` column is used to identify the language the translated values are associated with. Note that the `LANGUAGE_CODE` column follows the `_%CODE` naming convention. Language is considered a domain with a given set of unique values.

### ETL Process

The `W_DOMAIN_MEMBER_LKP_TL` table stores both domains that are extracted from the source system as well as internally defined domains that are seeded in the Configuration Manager. For each of the `_%CODE` columns that have translated values available in the source system, an ETL process extracts the domain members from the transactional system and loads them into `W_DOMAIN_MEMBER_LKP_TL`. Internally defined domains—usually domains specific to the Oracle Business Analytics Warehouse and known as conformed domains but can also include source domains—are stored in the Configuration Manager schema and are similarly extracted and loaded into the `W_DOMAIN_MEMBER_LKP_TL` table through ETL processes.

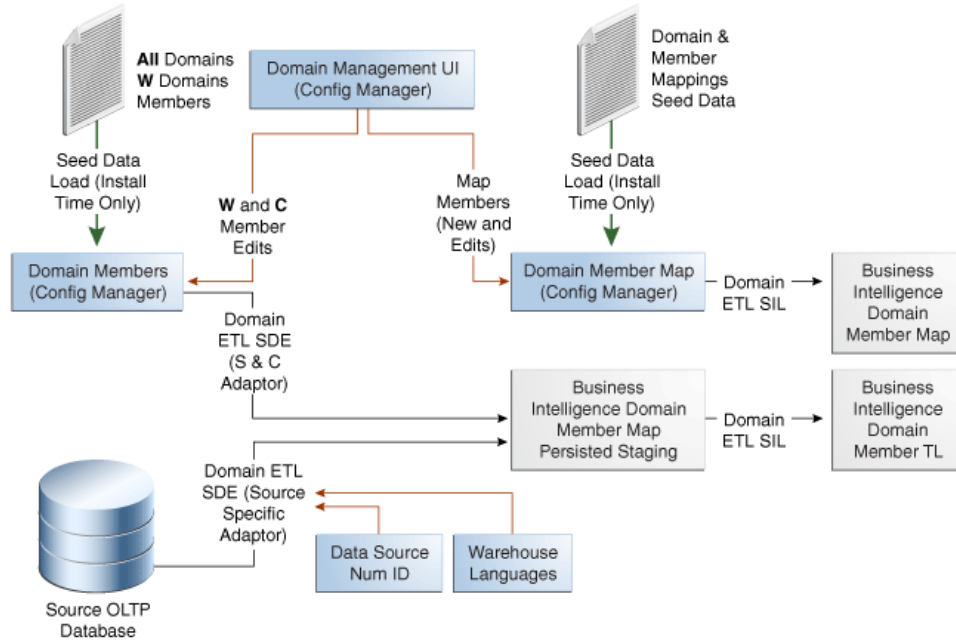
Only those translation records that match one of the languages that have been installed in the data warehouse are extracted from the transactional system. If translated records are not found in the transactional system matching an installed language, the ETL will generate a 'pseudo-translated' record for that language.

Some source applications store translations that can be extracted and loaded into the translation table. Some source applications do not maintain translations for an entity that corresponds to a dimension table. In these cases, whatever record is available is

extracted and used as the Base language record to generate pseudo-translations for all other installed languages.

Figure 2–1 shows an overview of the Oracle BI Applications domain ETL process.

**Figure 2–1 Overview of BI Applications Domain ETL Process**



**About Oracle BI Applications Domains and Oracle BI Enterprise Edition**

The exact mechanism used to retrieve the translated value in Oracle BI Enterprise Edition is the LOOKUP() function. When the LOOKUP() function is used, Oracle BI Enterprise Edition performs all aggregations before joining to the lookup table. The aggregated result set is then joined to the lookup table. Low-cardinality attributes tend to be involved in several aggregations, so it is useful to be joined after results are aggregated rather than before.

In a logical dimension, a Name or Description attribute will use the LOOKUP() function, passing the value in the %\_CODE column associated with that Name or Description to the Domain Lookup Table. The LOOKUP() function includes the Domain Name to be used when looking up values. The results from the Domain Lookup table are filtered to match the user's session language and returned as part of the query results.

Domains can be either source or conformed (internally defined warehouse domains). Source domains can come from a variety of transactional systems and so must include a Datasource\_Num\_Id value to resolve. Conformed domains are defined as part of the Oracle BI Applications and do not require a Datasource\_Num\_ID to resolve. As a result, there are two lookup tables implemented in the Oracle BI Repository that are aliases of W\_DOMAIN\_MEMBER\_LKP\_TL. When resolving a source domain, the source domain lookup requires Datasource\_Num\_Id to be passed as part of the LOOKUP() function while the conformed domain lookup does not.

**2.4 About Dimension Translation Tables**

As mentioned in Section 2.3, "About Oracle BI Applications Domains," domains are dimensional attributes that have a relatively small number of distinct members, have a



low cardinality relative to the number of records in the dimension, and are often used in aggregations. Dimensions have other attributes that require translation that may not fit one or more of these criteria. Dimensions may have translatable attributes that have a high cardinality relative to the dimension or may have a large number of members, and, thus, are not likely candidates for aggregation. If the domains ETL process was implemented in such cases, performance would be very poor. As a result, these particular attributes are implemented using dimension translation tables.

### Data Model

If a dimension has such high-cardinality attributes that cannot be treated as domains, the dimension will have an extension table that follows the `_TL` naming convention. If the `_TL` table has a one-to-one relationship with the dimension table (after filtering for languages), the `_TL` table name will match the dimension table name. For example, `W_JOB_D_TL` is the translation table associated with the `W_JOB_D` dimension table. If the `_TL` table does not have a one-to-one relationship with any dimension table, its name will reflect content.

The dimension and dimension translation table are joined on the translation table's `INTEGRATION_ID + DATASOURCE_NUM_ID`. If the translation and dimension tables have a one-to-one relationship (after filtering for language), the join to the dimension table is on its `INTEGRATION_ID + DATASOURCE_NUM_ID`. Otherwise, there will be a `%_ID` column in the dimension table that is used to join to the translation table.

### ETL Process

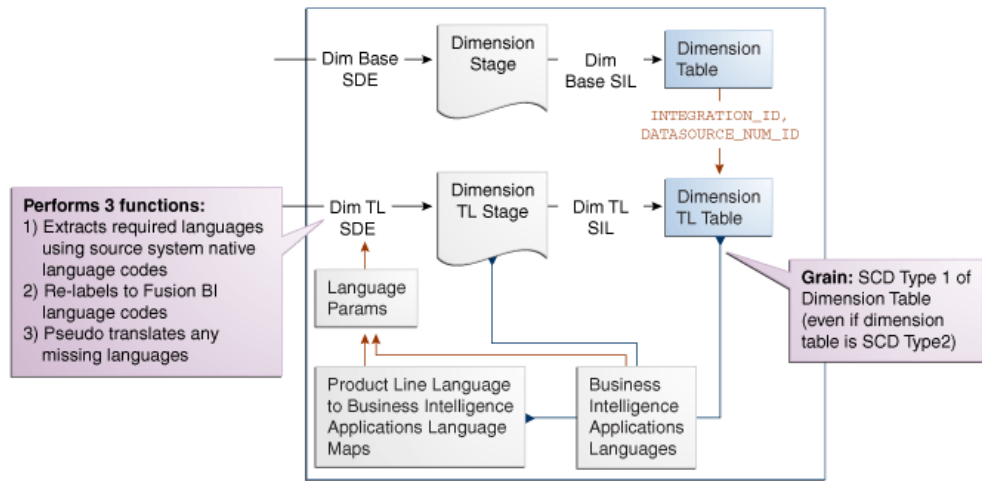
Similar to the Oracle BI Applications domain ETL process, when using dimension translation tables, ETL tasks extract the translated values from the transactional system. Rather than the domain staging table being loaded, the dimension's translation staging table is loaded. The ETL process then moves these records into the dimension translation table.

Only those translation records that match one of the languages that have been installed in the data warehouse are extracted from the transactional system. If translated records are not found in the transactional system matching a data warehouse Installed language, the ETL will generate a 'pseudo-translated' record for that language by copying the record that corresponds to the data warehouse Base language.

Some source applications store translations that can be extracted and loaded into the translation table. Some source applications do not maintain translations for an entity that corresponds to a dimension table. In these cases, whatever record is available is extracted and used as the Base language record, which is then used to generate pseudo-translations for all other Installed languages.

Oracle BI Applications does not support Type 2 SCD tracking of dimension translation attributes when the dimension and translation tables have a one-to-one relationship with each other. These tables are joined on `INTEGRATION_ID + DATASOURCE_NUM_ID`, and, therefore, can be joined to a single record in the translation table. Attributes in the dimension table can be Type 2-enabled, but the current and prior records will always have the same translated value. [Figure 2-2](#) describes the ETL domain process.

**Figure 2–2 Domain ETL Process**



**Oracle BI Enterprise Edition**

In Oracle BI Enterprise Edition, joins are created between the dimension and translation tables as normal. The translation table is brought in as another supporting table in the logical table source. If a user selects an attribute from the translation table, it will be included as a joined table in the SQL that Oracle BI Enterprise Edition generates. If the user does not select a translation attribute, the translation table will not be included in the generated SQL.

To ensure this behavior, the physical join between the dimension and translation tables is configured as one-to-many with the dimension table on the many side.

An important consideration is filtering on a user's language. If the language filter is included in the logical table source as a content filter, the translation table will always be joined whether a user selects a translation attribute or not. To avoid this behavior, opaque views are created in the physical layer that include a WHERE clause on the user's session language. Filtering on the user's language is still possible, but as the filter criteria is not implemented as a logical table source content filter, it is ensured that the translation table is only joined when necessary.

**Localizing New Domain Members and Oracle BI Repository Metadata**

If you added new domain members that require localization, see the section titled "How to Localize a New Domain Member," in *Oracle Fusion Middleware Configuration Guide for Oracle Business Intelligence Applications*.

Also, to add string localizations in the Oracle BI Repository metadata, see "How to Add String Localizations for Oracle BI Repository Metadata," in *Oracle Fusion Middleware Configuration Guide for Oracle Business Intelligence Applications*.

---

---

# Localizing Oracle Business Intelligence Deployments

This chapter describes concepts and techniques for localizing Oracle Business Intelligence Applications. Oracle Business Intelligence is designed to allow users to dynamically change their preferred language and locale preferences. This section explains how to configure Oracle BI Applications for deployment in one or more language environments other than English.

This chapter contains the following topics:

- [Section 3.1, "Process of Maintaining Translation Tables for Oracle BI EE"](#)
- [Section 3.2, "About Translating Presentation Services Strings"](#)
- [Section 3.3, "Changing the Default Currency in Analytics Applications"](#)

## 3.1 Process of Maintaining Translation Tables for Oracle BI EE

The Oracle Business Intelligence Presentation layer supports multiple translations for any column name. When working with Oracle BI Answers or rendering a dashboard, users see their local language strings in their reports. For example, English-speaking and French-speaking users would see their local language strings in their reports. There are two kinds of application strings requiring translation in Oracle Business Intelligence:

- **Metadata**  
Metadata strings are analytics-created objects in the Oracle Business Intelligence repository such as subject areas, metrics, and dimensions.
- **Presentation Services**  
Presentation Services objects are end-user created objects such as reports, dashboards, and pages. Translations for Presentation Services strings are stored in the XML caption files. For more information on accessing these strings and changing the translations, see *Oracle Business Intelligence Presentation Services Administration Guide*.

This process includes the following tasks:

- [Section 3.1.1, "Upgrading Oracle Business Intelligence Seed Data for Non-English Locales"](#)
- [Section 3.1.2, "Externalizing Customer Metadata Strings"](#)
- [Section 3.1.3, "Adding Custom Translations to the W\\_LOCALIZED\\_STRING\\_G Table"](#)

### 3.1.1 Upgrading Oracle Business Intelligence Seed Data for Non-English Locales

If Oracle Business Intelligence data in your deployment is to be viewed in a language other than English, you must do the following:

1. Verify creation of the Translation Table (W\_LOCALIZED\_STRING\_G) and corresponding indexes, as described in [Section 3.1.1.1, "Verify the Translation Table \(W\\_LOCALIZED\\_STRING\\_G\) and Corresponding Indexes"](#).
2. Import Locale seed data into the W\_LOCALIZED\_STRING\_G table, as described in [Section 3.1.1.2, "Importing Locale Seed Data Into The Translation Table \(W\\_LOCALIZED\\_STRING\\_G\)"](#).
3. Create an Initialization Block at the Session level to set the LOCALE variable.

For example, you might do the following:

- a. In Oracle BI EE Administration Tool, choose Manage, then Variables, to open the Variable Manager dialog.
- b. From the Action menu, choose New, then Session, then Initialization Block.
- c. In the Session Variable Initialization Block dialog, type a name for the block. For example, LOCAL\_INIT\_BLOCK.
- d. Click the Edit data source button.
- e. In the Default initialization string box, type the SQL initialization string. For example:
 

```
select 'VALUEOF(NQ_SESSION.WEBLANGUAGE)' from
VALUEOF(OLAPTBO).DUAL
```
- f. Click Browse next to the Connection Pool field to select an appropriate connection pool. For example, "Oracle EBS OLTP"."Oracle EBS OLTP InitBlocks Connection Pool".
- g. In the Variable Manager dialog, navigate to Session > Variables > Non-System.
- h. Double click the LOCAL variable to open the Session Variable dialog.
- i. In the Session Variable dialog, use the Initialization Block list to select the new initialization block, for example, LOCAL\_INIT\_BLOCK.

#### 3.1.1.1 Verify the Translation Table (W\_LOCALIZED\_STRING\_G) and Corresponding Indexes

To verify the Translation Table (W\_LOCALIZED\_STRING\_G) and corresponding indexes:

1. Verify that Oracle Business Analytics Warehouse contains the W\_LOCALIZED\_STRING\_G table.
2. Lookup the definitions of the following indexes in DAC and create them manually in Oracle Business Analytics Warehouse:
  - W\_LOCAL\_STRING\_G\_U1
  - W\_LOCAL\_STRING\_G\_P1
  - W\_LOCAL\_STRING\_G\_M1
  - W\_LOCAL\_STRING\_G\_M2

**Note:** It is better to add these indexes to `W_LOCALIZED_STRING_G` prior to importing the locale seed data in the next section, in order to safeguard against inadvertently duplicating the data in the table.

### 3.1.1.2 Importing Locale Seed Data Into The Translation Table (`W_LOCALIZED_STRING_G`)

If the primary language being used is not English, you might have to import additional locale seed data (depending on the number of languages you use). This process must be performed once for each language in which users might use in their Web client.

#### Notes

- This task should be performed only by a BI Administrator.
- To perform task, you need the `dataimp` utility, which can only be used on 32-bit operating systems.
- During the Oracle Business Intelligence Applications installation, a directory named `ORACLE_HOME\biapps\seeddata` is created, which contains a sub directory for each language. Within each language sub directory is a `.dat` file (the data to be imported) and an `.inp` file (the WHERE clause governing the import).

#### To import Locale seed data into the Translation Table (`W_LOCALIZED_STRING_G`)

1. Open a command window and navigate to `ORACLE_HOME\biapps\seeddata\bin` directory.
2. Run the import command in step 3 after replacing these connection parameters with the values appropriate to your database environment:
  - UserName
  - Password
  - ODBCDataSource
  - DatabaseOwner

3. Run the import command:

```
ORACLE_HOME\biapps\seeddata\bin\dataimp /u $UserName /p $Password /c
"$ODBCDataSource" /d $DatabaseOwner /f ORACLE_HOME\biapps\seeddata\l_
<XX>\analytics_seed_<XXX>.dat /w y /q 100 /h Log /x f /i ORACLE_
HOME\biapps\seeddata\l_<XX>\metadata_upgrade_<XXX>_<DBPlatform>.inp /l
metadata_upgrade_<XXX>.log
```

---

**Note:** Replace the `XX` with the Oracle Business Intelligence two-letter language code (`fr`, `it`) and the `XXX` with the Siebel Systems three-letter code (`FRA`, `ITA`).

---

4. When you have finished importing the Locale seed data into the Translation Table (`W_LOCALIZED_STRING_G`), configure the initialization block in the Oracle BI Repository using the Oracle BI Administration Tool to connect to the database where this table resides.

---

**Note:** Unicode connectivity can be used to access databases that do not support Unicode.

---

### 3.1.2 Externalizing Customer Metadata Strings

Metadata Strings are loaded by the Oracle BI Server from a database table. In the case of Oracle Business Intelligence applications, this table is `W_LOCALIZED_STRING_G` in the data warehouse. The initialization block 'Externalize Metadata Strings' loads the strings for the Server. It is recommended that you run a test to make sure that this initialization block runs successfully. An example of the translation table is shown in [Table 3–1](#).

**Table 3–1 Example of `W_LOCALIZED_STRING_G` Translation Table**

MSG_NUM	MSG_TEXT	LANG_ID
CN_Customer_Satisfaction	Customer Satisfaction	ENU
CN_Customer_Satisfaction	Kundenzufriedenheit	DEU
CN_Customer_Satisfaction	Satisfação do cliente	PTB

By default, the Oracle Business Intelligence repository is configured to run in English only. To deploy in any other language, you must externalize the metadata strings, as described in the following procedure.

#### To externalize metadata strings in the Oracle Business Intelligence repository

1. Stop the Oracle BI Server.
2. Using the Oracle BI Administration Tool in offline mode, open `OracleBIAnalyticsApps.rpd`.
3. Select the entire Presentation layer and right-click the mouse to display the menu.
  - From the pop-up menu, select Externalize Display Names. (A check mark appears next to this option the next time you right-click on the Presentation layer.)
  - Unselect the Presentation layer.

---

**Note:** When Externalize Display Names is checked, all metadata strings are read from the `W_LOCALIZED_STRING_G` table in the data warehouse.

---

4. In the Physical layer, select the Externalized Metadata Strings database icon. Expand the tree.
5. Double-click Internal System Connection Pool.  
In the Connection Pool dialog General tab, the field Data source name should point to the data warehouse.
6. Click OK and exit the Oracle BI Administration Tool.
7. Restart the Oracle BI Server.

### 3.1.3 Adding Custom Translations to the `W_LOCALIZED_STRING_G` Table

When you add custom objects to the metadata and choose to externalize these objects (by right-clicking the object and checking the Externalize Display Name option), the Oracle BI Server looks for the translations (including those for the native language) in the `W_LOCALIZED_STRING_G` table.

If you do not externalize the display names, you do not need to perform the following procedures.

---



---

**Note:** The custom Presentation layer objects show up only in the native language of the metadata (the language in which you added these new objects).

---



---

### 3.1.3.1 Adding String Translations for Analytics Metadata

The following procedure describes how to add string translations for Oracle Business Intelligence metadata to the W\_LOCALIZED\_STRING\_G table. This task occurs in any database administration tool, and in the Oracle BI Administration Tool.

#### To add string translations for Analytics metadata

1. Open a database administration tool and connect to your data warehouse database.
2. Query for the table named W\_LOCALIZED\_STRING\_G and add a new record to the table, as defined below in steps 4 to 8.
3. Obtain the Message Key from the Oracle BI Administration Tool as follows:
  - In the Oracle BI Administration Tool, right-click on the new Presentation layer metadata object and select Properties from the menu.
  - The Message key is displayed in the dialog under Custom Display Name. The Message key is the part that starts with CN\_.

For example, double-click the Pipeline catalog directory in the Presentation layer. The Custom Display name is Valueof(NQ\_SESSION.CN\_Pipeline). CN\_Pipeline is the Message Key.
4. Enter your deployment language in the new record.
5. Enter the Message Type required (for example, Metadata, FINS\_Metadata).
6. Select the Message Level *AnalyticsNew*, then do the following:
  - In the Message Text column, add the translation of the object.
  - Check the flags (set to Yes) for the Translate and Active columns.
  - Set the Error Message # column to 0.
7. Enter the required Message Facility (for example, HMF, FIN).
8. Repeat Step 3 through Step 7 for each new metadata object string.
9. Exit the database administration tool, then restart the Oracle BI Server.

## 3.2 About Translating Presentation Services Strings

The translations for such Presentation Services objects as report and page names are stored in the xxxCaptions.xml files available in the ORACLE\_HOME\biapps\catalog\res\web\l\_<Language Abbreviation>\Captions directories. In multiple language deployment mode, if you add any additional Presentation Services objects, such as reports and new dashboard pages, you also need to add the appropriate translations. Add these translations using the Catalog Manager tool. For more information on using this utility, see *Oracle Business Intelligence Presentation Services Administration Guide*.

## 3.3 Changing the Default Currency in Analytics Applications

In Oracle Business Intelligence Applications, you might see a dollar sign used as the default symbol when amounts of money are displayed. In order to change this behavior, you must edit the `currencies.xml` file using the following procedure. The `currencies.xml` file is located in the following directories:

- Windows:

```
ORACLE_HOME\bifoundation\web\display\currencies.xml
```

- UNIX:

```
ORACLE_HOME/bifoundation/web/display/currencies.xml
```

### To change the default currency in Analytics Applications

1. In a text editor, open the `currencies.xml` file.
2. Look for the currency tag for the warehouse default (tag="int:wrhs"):

```
<Currency tag="int:wrhs" type="international" symbol="$" format="$#" digits="2"
displayMessage="kmsgCurrencySiebelWarehouse">
  <negative tag="minus" format="-$#" />
</Currency>
```

3. Replace the symbol, format, digits and negative information in the warehouse default with the information from the currency tag you want to use as the default.

For example, if you want the Japanese Yen to be the default, replace the contents of the warehouse default currency tag with the values from the Japanese currency tag (tag="loc:ja-JP"):

```
<Currency tag="loc:ja-JP" type="local" symbol="¥" locale="ja-JP" format="$#"
digits="0">
  <negative tag="minus" format="-$#" />
</Currency>
```

When you are finished, the default warehouse currency tag for Japanese should look like the following example:

```
<Currency tag="int:wrhs" type="international" symbol="¥" format="$#" digits="0"
displayMessage="kmsgCurrencySiebelWarehouse">
  <negative tag="minus" format="-$#" />
</Currency>
```

4. Save and close the `currencies.xml` file.



---

---

# Oracle Business Analytics Warehouse Naming Conventions

This chapter includes information on the types of tables and columns in the Oracle Business Analytics Warehouse, including the naming conventions used.

---

---

**Note:** This chapter contains naming conventions used for database tables and columns in the Oracle Business Analytics Warehouse. This information does not apply to objects in the Oracle Business Intelligence repository.

---

---

This chapter contains the following topics:

- [Section 4.1, "Naming Conventions for Oracle Business Analytics Warehouse Tables"](#)
- [Section 4.2, "Table Types for Oracle Business Analytics Warehouse"](#)
- [Section 4.3, "Internal Tables in Oracle Business Analytics Warehouse"](#)
- [Section 4.4, "Standard Column Prefixes in Oracle Business Analytics Warehouse"](#)
- [Section 4.5, "Standard Column Suffixes in Oracle Business Analytics Warehouse"](#)
- [Section 4.6, "System Columns in Oracle Business Analytics Warehouse Tables"](#)
- [Section 4.7, "Multi-Currency Support for System Columns"](#)
- [Section 4.8, "Oracle Business Analytics Warehouse Primary Data Values"](#)
- [Section 4.8, "Oracle Business Analytics Warehouse Primary Data Values"](#)
- [Section 4.9, "About Multi-Language Support in the Oracle Business Analytics Warehouse"](#)
- [Section 4.10, "Oracle Business Analytics Warehouse Currency Preferences"](#)

## 4.1 Naming Conventions for Oracle Business Analytics Warehouse Tables

Oracle Business Analytics Warehouse tables use a three-part naming convention: PREFIX\_NAME\_SUFFIX, as shown in [Table 4-1](#).

**Table 4–1 Naming Conventions for Oracle Business Analytics Data Warehouse Tables**

Part	Meaning	Table Type
PREFIX	Shows Oracle Business Analytics-specific data warehouse application tables.	W_ = Warehouse
NAME	Unique table name.	All tables.
SUFFIX	Indicates the table type.	_A = Aggregate _D = Dimension _DEL = Delete _DH = Dimension Hierarchy _DHL = Dimension Helper _DHLS = Staging for Dimension Helper _DHS = Staging for Dimension Hierarchy _DS = Staging for Dimension _F = Fact _FS = Staging for Fact _G, _GS = Internal _H = Helper _HS = Staging for Helper _MD = Mini Dimension _PE = Primary Extract _PS = Persisted Staging _RH = Row Flattened Hierarchy _TL = Translation Staging (supports multi-language support) _TMP = Pre-staging or post-staging temporary table _UD = Unbounded Dimension _WS = Staging for Usage Accelerator

## 4.2 Table Types for Oracle Business Analytics Warehouse

Table 4–2 lists the types of tables used in the Oracle Business Analytics Warehouse.

**Table 4–2 Table Types Used in the Oracle Business Analytics Warehouse**

Table Type	Description
Aggregate tables (_A)	Contain summed (aggregated) data.
Dimension tables (_D)	Star analysis dimensions.
Delete tables (_DEL)	<p>Tables that store IDs of the entities that were physically deleted from the source system and should be flagged as deleted from the data warehouse.</p> <p>Note that there are two types of delete tables: _DEL and _PE. For more information about the _PE table type, see the row for Primary extract tables (_PE) in this table.</p>
Dimension Hierarchy tables (_DH)	Tables that store the dimension's hierarchical structure.
Dimension Helper tables (_DHL)	Tables that store many-to-many relationships between two joining dimension tables.
Staging tables for Dimension Helper (_DHLS)	Staging tables for storing many-to-many relationships between two joining dimension tables.
Dimension Hierarchy Staging table (_DHS)	Staging tables for storing the hierarchy structures of dimensions that have not been through the final extract-transform-load (ETL) transformations.

**Table 4–2 (Cont.) Table Types Used in the Oracle Business Analytics Warehouse**

<b>Table Type</b>	<b>Description</b>
Dimension Staging tables (_DS)	Tables used to hold information about dimensions that have not been through the final ETL transformations.
Fact tables (_F)	Contain the metrics being analyzed by dimensions.
Fact Staging tables (_FS)	Staging tables used to hold the metrics being analyzed by dimensions that have not been through the final ETL transformations.
Internal tables (_G, _GS)	General tables used to support ETL processing.
Helper tables (_H)	Inserted between the fact and dimension tables to support a many-to-many relationship between fact and dimension records.
Helper Staging tables (_HS)	Tables used to hold information about helper tables that have not been through the final ETL transformations.
Mini dimension tables (_MD)	Include combinations of the most queried attributes of their parent dimensions. The database joins these small tables to the fact tables.
Primary extract tables (_PE)	Tables used to support the soft delete feature. The table includes all the primary key columns (integration ID column) from the source system. When a delete event happens, the full extract from the source compares the data previously extracted in the primary extract table to determine if a physical deletion was done in the Siebel application. The soft delete feature is disabled by default. Therefore, the primary extract tables are not populated until you enable the soft delete feature.  Note that there are two types of delete tables: _DEL and _PE. For more information about the _DEL table type, see the row for Delete table (_DEL) in this table.
Persisted Staging table (_PS)	Tables that source multiple data extracts from the same source table.  These tables perform some common transformations required by multiple target objects. They also simplify the source object to a form that is consumable by the warehouse needed for multiple target objects. These tables are never truncated during the life of the data warehouse. These are truncated only during full load, and therefore, persist the data throughout.
Row Flattened Hierarchy Table (_RH)	Tables that record a node in the hierarchy by a set of ancestor-child relationships (parent-child for all parent levels).
Translation Staging tables (_TL)	Tables store names and descriptions in the languages supported by Oracle BI Applications.
Pre-staging or post-staging Temporary table (_TMP)	Source-specific tables used as part of the ETL processes to conform the data to fit the universal staging tables (table types_DS and _FS). These tables contain intermediate results that are created as part of the conforming process.
Unbounded dimension (_UD)	Tables containing information that is not bounded in transactional database data but should be treated as bounded data in the Oracle Business Analytics Warehouse.

**Table 4–2 (Cont.) Table Types Used in the Oracle Business Analytics Warehouse**

Table Type	Description
Staging tables for Usage Accelerator (_WS)	Tables containing the necessary columns for the ETL transformations.

### 4.2.1 Aggregate Tables in Oracle Business Analytics Warehouse

One of the main uses of a data warehouse is to sum up fact data with respect to a given dimension, for example, by date or by sales region. Performing this summation on-demand is resource-intensive, and slows down response time. The Oracle Business Analytics Warehouse precalculates some of these sums and stores the information in *aggregate tables*. In the Oracle Business Analytics Warehouse, the aggregate tables have been suffixed with `_A`.

### 4.2.2 Dimension Class Tables in Oracle Business Analytics Warehouse

A class table is a single physical table that can store multiple logical entities that have similar business attributes. Various logical dimensions are separated by a separator column, such as, type or category. `W_XACT_TYPE_D` is an example of a dimension class table. Different transaction types, such as, sales order types, sales invoice types, purchase order types, and so on, can be housed in the same physical table.

You can add additional transaction types to an existing physical table and so reduce the effort of designing and maintaining new physical tables. However, while doing so, you should consider that attributes specific to a particular logical dimension cannot be defined in this physical table. Also, if a particular logical dimension has a large number of records, it might be a good design practice to define a separate physical table for that particular logical entity.

### 4.2.3 Dimension Tables in Oracle Business Analytics Warehouse

The unique numeric key (`ROW_WID`) for each dimension table is generated during the load process. This key is used to join each dimension table with its corresponding fact table or tables. It is also used to join the dimension with any associated hierarchy table or extension table. The `ROW_WID` columns in the Oracle Business Analytics Warehouse tables are numeric. In every dimension table, the `ROW_WID` value of zero is reserved for Unspecified. If one or more dimensions for a given record in a fact table is unspecified, the corresponding key fields in that record are set to zero.

### 4.2.4 Dimension Tables With Business Role-Based Flags

This design approach is used when the entity is logically the same but participates as different roles in the business process. As an example, an employee could participate in a Human Resources business process as an employee, in the sales process as a sales representative, in the receivables process as a collector, and in the purchase process as a buyer. However, the employee is still the same. For such logical entities, flags have been provided in the corresponding physical table (for example, `W_EMPLOYEE_D`) to describe the record's participation in business as different roles.

While configuring the presentation layer, the same physical table can be used as a specific logical entity by flag-based filters. For example, if a particular star schema requires Buyer as a dimension, the Employee table can be used with a filter where the Buyer flag is set to Y.

## 4.2.5 Fact Tables in Oracle Business Analytics Warehouse

Each fact table contains one or more numeric foreign key columns to link it to various dimension tables.

## 4.2.6 Helper Tables in Oracle Business Analytics Warehouse

Helper tables are used by the Oracle Business Analytics Warehouse to solve complex problems that cannot be resolved by simple dimensional schemas.

In a typical dimensional schema, fact records join to dimension records with a many-to-one relationship. To support a many-to-many relationship between fact and dimension records, a helper table is inserted between the fact and dimension tables.

The helper table can have multiple records for each fact and dimension key combination. This allows queries to retrieve facts for any given dimension value. It should be noted that any aggregation of fact records over a set of dimension values might contain overlaps (due to a many-to-many relationship) and can result in double counting.

At times there is a requirement to query facts related to the children of a given parent in the dimension by only specifying the parent value (example: manager's sales fact that includes sales facts of the manager's subordinates). In this situation, one helper table containing multiple records for each parent-child dimension key combination is inserted between the fact and the dimension. This allows queries to be run for all subordinates by specifying only the parent in the dimension.

## 4.2.7 Hierarchy Tables in Oracle Business Analytics Warehouse

Some dimension tables have hierarchies into which each record rolls. This hierarchy information is stored in a separate table, with one record for each record in the corresponding dimension table. This information allows users to drill up and down through the hierarchy in reports.

There are two types of hierarchies in the Oracle Business Analytics Warehouse: a structured hierarchy in which there are fixed levels, and a hierarchy with parent-child relationships. Structured hierarchies are simple to model, since each child has a fixed number of parents and a child cannot be a parent. The second hierarchy, with unstructured parent-child relationships is difficult to model because each child record can potentially be a parent and the number of levels of parent-child relationships is not fixed. Hierarchy tables have a suffix of `_DH`.

## 4.2.8 Mini-Dimension Tables in Oracle Business Analytics Warehouse

Mini-dimension tables include combinations of the most queried attributes of their parent dimensions. They improve query performance because the database does not need to join the fact tables to the big parent dimensions but can join these small tables to the fact tables instead.

[Table 4–3](#) lists the mini-dimension tables in the Oracle Business Analytics Warehouse.

**Table 4–3 Mini-Dimension Tables in Oracle Business Analytics Warehouse**

Table Name	Parent Dimension
W_RESPONSE_MD	Parent W_RESPONSE_D
W_AGREE_MD	Parent W_AGREE_D
W_ASSET_MD	Parent W_ASSET_D

**Table 4–3 (Cont.) Mini-Dimension Tables in Oracle Business Analytics Warehouse**

Table Name	Parent Dimension
W_OPTY_MD	Parent W_OPTY_D
W_ORDER_MD	Parent W_ORDER_D
W_QUOTE_MD	Parent W_QUOTE_D
W_SRVREQ_MD	Parent W_SRVREQ_D

### 4.2.9 Staging Tables in Oracle Business Analytics Warehouse

Staging tables are used primarily to stage incremental data from the transactional database. When the ETL process runs, staging tables are truncated before they are populated with change capture data. During the initial full ETL load, these staging tables hold the entire source data set for a defined period of history, but they hold only a much smaller volume during subsequent refresh ETL runs.

This staging data (list of values translations, computations, currency conversions) is transformed and loaded to the dimension and fact staging tables. These tables are typically tagged as <TableName>\_DS or <TableName>\_FS. The staging tables for the Usage Accelerator are tagged as WS\_<TableName>.

The staging table structure is independent of source data structures and resembles the structure of data warehouse tables. This resemblance allows staging tables to also be used as interface tables between the transactional database sources and data warehouse target tables.

### 4.2.10 Translation Tables in Oracle Business Analytics Warehouse

Translation tables provide multi-language support by storing names and descriptions in each language that Oracle Business Analytics Warehouse supports. There are two types of translation tables:

- Domain tables that provide multi-language support associated with the values stored in the %\_CODE columns.
- Tables that provide multi-language support for dimensions.

Domains and their associated translated values are stored in a single table named W\_DOMAIN\_MEMBER\_LKP\_TL. Each dimension requiring multi-language support that cannot be achieved with domains has an associated \_TL table. These tables have a one-to-many relationship with the dimension table. For each record in the dimension table, you will see multiple records in the associated translation table (one record for each supported language).

## 4.3 Internal Tables in Oracle Business Analytics Warehouse

Internal tables are used primarily by ETL mappings for data transformation and for controlling ETL runs. These tables are not queried by end users. These tables are described in [Table 4–4](#).

**Table 4–4 Oracle Business Analytics Warehouse Internal Tables**

Name	Purpose	Location
W_DUAL_G	Used to generate records for the Day dimension.	Data warehouse
W_COSTLST_G	Stores cost lists.	Data warehouse

**Table 4–4 (Cont.) Oracle Business Analytics Warehouse Internal Tables**

Name	Purpose	Location
W_DOMAIN_MEMBER_G	Staging table for populating incremental changes into W_DOMAIN_MEMBER_G and W_DOMAIN_MEMBER_G_TL.	Data warehouse
W_DOMAIN_MEMBER_G_TL	Stores translated values for each installed language corresponding to the domain member codes in W_DOMAIN_MEMBER_G_TL.	Data warehouse
W_DOMAIN_MEMBER_GS	Stores all the domain members and value for each installed language.	Data warehouse
W_DOMAIN_MEMBER_MAP_G	Used at ETL run time to resolve at target domain code base on the value of a source domain code.	Data warehouse
W_DOMAIN_MEMBER_MAP_NUM_G	Used at ETL run time to resolve a target domain code based on the comparison of a numeric value within the source numeric range.	Data warehouse
W_EXCH_RATE_G	Stores exchange rates.	Data warehouse
W_LANGUAGES_G	Stores the language translations supported in the data warehouse and is used during ETL to help generate missing translation records from the base language called pseudo-translation	Data warehouse
W_LOCALIZED_STRING_G		Data warehouse
W_LOV_EXCPT_G	Stores the list of values for the list of values types in which the ETL process finds exceptions.	Data warehouse
W_UOM_CONVERSION_G	Stores a list of From and To UOM codes and their conversion rates.	Data warehouse

## 4.4 Standard Column Prefixes in Oracle Business Analytics Warehouse

The Oracle Business Analytics Warehouse uses a standard prefix to indicate fields that must contain specific values, as shown in [Table 4–5](#).

**Table 4–5 Standard Column Prefix**

Prefix	Description	In Table Types
W_	Used to store Oracle BI Applications standard or standardized values. For example, W_%_CODE (Warehouse Conformed Domain) and W_TYPE, W_INSERT_DT (Date records inserted into Warehouse).	_A _D _F

## 4.5 Standard Column Suffixes in Oracle Business Analytics Warehouse

The Oracle Business Analytics Warehouse uses suffixes to indicate fields that must contain specific values, as shown in [Table 4–6](#).

**Table 4–6 Standard Column Suffixes**

Suffix	Description	In Table Types
_CODE	Code field. (Especially used for domain codes.)	_D, _DS, _FS, _G, _GS
_DT	Date field.	_D, _DS, _FS, _G, _DHL, _DHLS

**Table 4–6 (Cont.) Standard Column Suffixes**

Suffix	Description	In Table Types
_ID	Correspond to the _WID columns of the corresponding _F table.	_FS, _DS
_FLG	Indicator or Flag.	_D, _DHL, _DS, _FS, _F, _G, _DHLS
_WID	Identifier generated by Oracle Business Intelligence linking dimension and fact tables, except for ROW_WID.	_F, _A, _DHL
_NAME	A multi-language support column that holds the name associated with an attribute in all languages supported by the data warehouse.	_TL
_DESCR	A multi-language support column that holds the description associated with an attribute in all languages supported by the data warehouse.	_TL

## 4.6 System Columns in Oracle Business Analytics Warehouse Tables

Oracle Business Analytics Warehouse tables contain system fields. These system fields are populated automatically and should not be modified by the user. [Table 4–7](#) lists the system columns used in data warehouse dimension tables.

**Table 4–7 System Columns Used in Data Warehouse Tables**

System Column	Description
ROW_WID	Surrogate key to identify a record uniquely.
CREATED_BY_WID	Foreign key to the W_USER_D dimension that specifies the user who created the record in the source system.
CHANGED_BY_WID	Foreign key to the W_USER_D dimension that specifies the user who last modified the record in the source system.
CREATED_ON_DT	The date and time when the record was initially created in the source system.
CHANGED_ON_DT	The date and time when the record was last modified in the source system.
AUX1_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX2_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX3_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX4_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
DELETE_FLG	This flag indicates the deletion status of the record in the source system. A value of Y indicates the record is deleted from the source system and logically deleted from the data warehouse. A value of N indicates that the record is active.
W_INSERT_DT	Stores the date on which the record was inserted in the data warehouse table.



**Table 4–7 (Cont.) System Columns Used in Data Warehouse Tables**

<b>System Column</b>	<b>Description</b>
W_UPDATE_DT	Stores the date on which the record was last updated in the data warehouse table.
DATASOURCE_NUM_ID	Unique identifier of the source system from which data was extracted. In order to be able to trace the data back to its source, it is recommended that you define separate unique source IDs for each of your different source instances.
ETL_PROC_WID	System field. This column is the unique identifier for the specific ETL process used to create or update this data.
INTEGRATION_ID	Unique identifier of a dimension or fact entity in its source system. In case of composite keys, the value in this column can consist of concatenated parts.
TENANT_ID	Unique identifier for a tenant in a multi-tenant environment. This column is typically used in an Application Service Provider (ASP)/Software as a Service (SaaS) model.
X_CUSTOM	Column used as a generic field for customer extensions.
CURRENT_FLG	This is a flag for marking dimension records as "Y" in order to represent the current state of a dimension entity. This flag is typically critical for Type II slowly changing dimensions, as records in a Type II situation tend to be numerous.
EFFECTIVE_FROM_DT	This column stores the date from which the dimension record is effective. A value is either assigned by Oracle BI Applications or extracted from the source.
EFFECTIVE_TO_DT	This column stores the date up to which the dimension record is effective. A value is either assigned by Oracle BI Applications or extracted from the source.
SRC_EFF_FROM_DT	This column stores the date from which the source record (in the Source system) is effective. The value is extracted from the source (whenever available).
STC_EFF_TO_DT	This column stores the date up to which the source record (in the Source system) is effective. The value is extracted from the source (whenever available).

## 4.7 Multi-Currency Support for System Columns

Table 4–8 lists the currency codes and rates for related system columns.

**Table 4–8 Currency Codes and Rates for Related System Columns**

<b>System Column</b>	<b>Description</b>
DOC_CURR_CODE	Code for the currency in which the document was created in the source system.
LOC_CURR_CODE	Usually the reporting currency code for the financial company in which the document was created.
GRP_CURR_CODE	The primary group reporting currency code for the group of companies or organizations in which the document was created.
LOC_EXCHANGE_RATE	Currency conversion rate from the document currency code to the local currency code.
GLOBAL1_EXCHANGE_RATE	Currency conversion rate from the document currency code to the Global1 currency code.

**Table 4–8 (Cont.) Currency Codes and Rates for Related System Columns**

System Column	Description
GLOBAL2_EXCHANGE_RATE	Currency conversion rate from the document currency code to the GLOBAL2 currency code.
GLOBAL3_EXCHANGE_RATE	Currency conversion rate from document currency code to the GLOBAL3 currency code.
PROJ_CURR_CODE	Code used in Project Analytics that corresponds to the project currency in the OLTP system.

## 4.8 Oracle Business Analytics Warehouse Primary Data Values

It is possible for various dimensions to have one-to-many and many-to-many relationships with each other. These kinds of relationships can introduce problems in analyses. For example, an Opportunity can be associated with many Sales Representatives and a Sales Representative can be associated with many Opportunities. If your analysis includes both Opportunities and Sales Representatives, a count of Opportunities would not be accurate because the same Opportunity would be counted for each Sales Representative with which it is associated.

To avoid these kinds of problems, the Oracle Business Analytics Warehouse reflects the primary member in the "many" part of the relationship. In the example where an Opportunity can be associated with many Sales Representatives, only the Primary Sales Representative is associated with that Opportunity. In an analysis that includes both Opportunity and Sales Representative, only a single Opportunity will display and a count of Opportunities returns the correct result.

There are a few important exceptions to this rule. The Person star schema supports a many-to-many relationship between Contacts and Accounts. Therefore, when querying the Person star schema on both Accounts and Contacts, every combination of Account and Contact is returned. The Opportunity-Competitor star schema supports a many-to-many relationship between Opportunities and Competitor Accounts, and the Campaign-Opportunity star schema supports a many-to-many relationship between Campaigns and Opportunities. In other star schemas, however, querying returns only the primary account for a given contact.

## 4.9 About Multi-Language Support in the Oracle Business Analytics Warehouse

Oracle BI Applications provides multi-language support for metadata level objects exposed in Oracle BI Enterprise Edition dashboards and reports, as well as data, which enables users to see records translated in their preferred language. For more information about multi-language support, see [Chapter 2, "About Multi-Language Support."](#)

## 4.10 Oracle Business Analytics Warehouse Currency Preferences

For information about setting up currencies, refer to the following task in Functional Setup Manager: **Common Areas and Dimensions Configurations\ Configure Global Currencies.**

The Oracle Business Analytics Warehouse supports the following currency preferences.

- Contract currency.** The currency used to define the contract amount. This currency is used only in Project Analytics.

- **CRM currency.** The CRM corporate currency as defined in the Fusion CRM application. This currency is used only in CRM Analytics applications.
- **Document currency.** The currency in which the transaction was done and the related document created.
- **Global currency.** The Oracle Business Analytics Warehouse stores up to three group currencies. These need to be pre-configured so as to allow global reporting by the different currencies. The exchange rates are stored in the table W\_EXCH\_RATE\_G.
- **Local currency.** The accounting currency of the legal entity in which the transaction occurred.
- **Project currency.** The currency in which the project is managed. This may be different from the functional currency. This applies only to Project Analytics.



---

---

# Oracle Business Intelligence Endeca Discovery Option

This chapter explains how to set up and use Oracle Endeca with Oracle BI Applications.

This chapter contains the following sections:

- [Section 5.1, "Introduction"](#)
- [Section 5.2, "Setting Up Endeca Integration with Oracle BI Applications"](#)
- [Section 5.3, "Tasks for Setting Up Oracle Endeca with Oracle BI Applications"](#)
- [Section 5.4, "ODI Package, Interface, and Procedure Design Overview"](#)
- [Section 5.5, "Deploying Sample Applications"](#)
- [Section 5.6, "Applying the Custom BI Applications Security Manager"](#)
- [Section 5.7, "Troubleshooting"](#)

## 5.1 Introduction

Oracle BI Applications can optionally be used with Oracle Endeca Information Discovery, which can use Oracle BI Applications data for its data visualizations. Oracle Endeca Information Discovery is an enterprise discovery platform for the analysis of information from both structured and unstructured sources, providing the ability to search for and visualize data in many different ways, including tag clouds.

## 5.2 Setting Up Endeca Integration with Oracle BI Applications

This section provides instructions for setting up Oracle Endeca with Oracle BI Applications.

### List of Steps for Setting Up Oracle Endeca Integration with Oracle BI Applications

To install and set up Endeca integration with Oracle BI Applications, you must complete the following tasks, in order. Each high-level task breaks down into a list of detailed steps provided in the next section.

1. Install Oracle Endeca Server 7.6.0. For information about requirements and installation of the Endeca Server, refer to the *Oracle Endeca Server Installation Guide*. Install all available patches.
2. Install Endeca 3.1 Studio. For information about requirements and installation of the Studio, refer to the *Oracle Endeca Information Discovery Studio Installation Guide*. Install available patches.

3. In Oracle Data Integrator (ODI) Studio, set up the ODI topology to include Oracle Business Intelligence Enterprise Edition as a source, the Endeca Server as a target, and access to required, as described in [Section 5.3.3, "Setup Step: Set Up ODI Connectivity for Oracle BI and Endeca."](#)
4. Load data by running ODI scenarios individually or by running the ODI load plan, as described in [Section 5.3.4, "Setup Step: Load Data."](#)
5. Create a new Endeca Studio Application and deploy the Sample Application, as described in [Section 5.3.5, "Setup Step: Create a New Endeca Application."](#)

## 5.3 Tasks for Setting Up Oracle Endeca with Oracle BI Applications

This section provides detailed tasks for setting up Oracle Endeca with Oracle BI Applications, and contains the following sections:

- [Section 5.3.1, "Setup Step: Install Oracle Endeca Server"](#)
- [Section 5.3.2, "Setup Step: Install Endeca 3.1 Studio"](#)
- [Section 5.3.3, "Setup Step: Set Up ODI Connectivity for Oracle BI and Endeca"](#)
- [Section 5.3.4, "Setup Step: Load Data"](#)
- [Section 5.3.5, "Setup Step: Create a New Endeca Application"](#)

**Note:** You must perform the tasks in this section in the sequence described in [Section 5.2, "Setting Up Endeca Integration with Oracle BI Applications"](#).

### 5.3.1 Setup Step: Install Oracle Endeca Server

For information about requirements and installation of the Endeca Server, refer to the *Oracle Endeca Server Installation Guide*. Install all available patches. Optionally, you can take some steps to optimize the Endeca Server described below.

#### Recommended Data Size

For Endeca Server hardware configured for six or less threads, it is recommended that you limit the data size as follows:

1. Limit data to 300 attributes and 1,000,000 rows.
2. The less attributes included, the more rows can be loaded in the same amount of time, up to 300 million attributes per hour.
3. The more processing available, the higher the number of rows that can be loaded in a given time.

#### Creating Endeca Data Domain Profile to Optimize CPU Core Count

Endeca Server's Data Domain Profile has its `numComputeThreads` property set to 2, representing two CPU Cores. Create a new Data Domain Profile to set the `numComputeThreads` value appropriately for the CPU cores available on the server when creating new Data Domains. For information about setting this property and creating domains using the `endeca-cmd` utility, refer to the chapter titled, "Endeca Server Command Reference" in the *Oracle Endeca Server Administrator's Guide*.

After the new Data Domain Profile is created, set the new profile name to the ODI parameter `ENDECA_DATADOMAIN_PROFILE` when running the Endeca Load Plan or scenarios. For more information about running the Endeca Load Plan and scenario, refer to [Section 5.3.4, "Setup Step: Load Data"](#).

### Disabling, Enabling, or Deleting Data Domains

During data ingest, data domains reach their highest memory usage and do not release memory. Domains tend to page memory as needed to make room for other memory requests made to the operating system. If the memory usage nears the maximum amount available, disabling and then enabling each data domain can reduce their claim on memory.

The below endeca-cmd commands can be run on the server to disable an existing domain and then enable it again. The location of endeca-cmd command line utility is `$MWHOME\EndecaServer7.6.1\endeca-cmd`:

- To disable a data domain: `endeca-cmd disable-dd <data-domain>`
- To enable a data domain: `endeca-cmd enable-dd <data-domain>`

A data domain can be left disabled to preserve the existing data and not required to be online for a period of time. If a data domain is no longer needed, it can also be deleted to free resources. No backup is performed, so when a data domain is deleted it cannot be restored. A shell or batch file script can be used to disable, enable, and delete multiple data domains.

## 5.3.2 Setup Step: Install Endeca 3.1 Studio

Install Endeca 3.1 Studio. For information about requirements and installation of the Studio, refer to the *Oracle Endeca Information Discovery Studio Installation Guide*. Install available patches.

## 5.3.3 Setup Step: Set Up ODI Connectivity for Oracle BI and Endeca

In this step, you create source and target server definitions in Oracle Data Integrator (ODI) to support Endeca ETL in the BI Applications data warehouse, as well as a definition for required Endeca view XML files.

### Create Oracle BI Enterprise Edition Source

Endeca when integrated with Oracle BI Applications sources data from the Oracle Business Analytics Warehouse using an Endeca load plan, which denormalizes the OBAW data and loads it into the Endeca data domain, before using BI Server metadata such as data type, column name, attribute group name, and so on to load the Endeca schema. To support this integration, you create an ODI Data Server definition in the ODI repository metadata to support connections to the Oracle BI Server.

1. In ODI Designer's Topology tab, expand the Oracle BI Technology in the Physical Architecture.
2. In the Oracle BI Technology, create a new BI Applications data server.
3. In the Definition tab of the new server, enter a name and, in the Connection details, provide User and Password credentials for an Oracle BI EE Server administrative user.
4. In the JDBC tab, enter 'oracle.bi.jdbc.AnaJdbcDriver' as the JDBC Driver.
5. Enter 'jdbc:oraclebi://<IP address or the Oracle BI EE Server hostname>:9703/' as the JDBC URL.

---

**Note:** Ensure that there are no leading or trailing spaces in the JDBC URL.

---

6. In the BI Applications data server, create a new Physical Schema. In its Definition tab, specify Core for the Catalog (Catalog) and for Catalog (Work Catalog).
7. In ODI Designer's Topology tab, expand the Logical Architecture and assign the OBI\_BIAPPS11G logical schema to the physical schema you created.

#### Create Endeca Server Target

To support Endeca Server integration, you create a target ODI Data Server definition in the ODI Physical Architecture for the Endeca Server.

1. In ODI Designer's Topology tab, expand the Endeca Server Technology in the Physical Architecture.
2. In the Endeca Server Technology, create a new BI Applications data server.
3. In the Definition tab of the new server, enter a name. In the Connection details, leave the User and Password fields empty.
4. In the JDBC tab, leave the JDBC Driver empty.
5. Enter 'http://<IP address or the Endeca Server hostname>:7001/endeca-server' as the JDBC URL.

---

---

**Note:** Ensure that there are no leading or trailing spaces in the JDBC URL.

---

---

6. In the BI Applications data server, create a new Physical Schema. In its Definition tab, you can leave Catalog (Catalog) and Catalog (Work Catalog) empty.
7. In ODI Designer's Topology tab, expand the Logical Architecture and assign the OEID\_BIAPPS11G logical schema to the physical schema you created.

#### Create Data Server for Endeca Files

To support Endeca Server integration, you create a target ODI Data Server definition in the ODI Physical Architecture for Endeca files used by the Endeca web service to load view XML, including Sample Applications.

1. In ODI Designer's Topology tab, expand the File Technology in the Physical Architecture.
2. In the File Technology, create a new BI Applications data server.
3. In the Definition tab of the new server, enter a name. In the Connection details, leave the User and Password fields empty.
4. In the JDBC tab, enter 'com.sunopsis.jdbc.driver.file.FileDriver' as the JDBC Driver.
5. Enter 'jdbc:snps:dbfile?OPT=TRUE' as the JDBC URL.

---

---

**Note:** Ensure that there are no leading or trailing spaces in the JDBC URL.

---

---

6. In the BI Applications data server, create a new Physical Schema. Use the directory containing the view XML files (for example, C:\Temp) for Directory (Schema) and Directory (Work Schema). This directory should be relative to the server where the ODI Agent is installed and can access the view XML files.



---

**Note:** The Sample Apps View XMLs can be found in the BIAPPS installation under *<Middleware Home>\Oracle\_BI1\biapps\admin\provisioning\endeca\OracleBIApps\_Endeca.zip*.

---

7. In ODI Designer's Topology tab, expand the Logical Architecture and assign the DW\_OEID\_SRCFILES logical schema to the physical schema you created.

### Exporting and Importing Views Using Endeca Web Service Procedure

The Endeca Web Service Procedure supports exporting views. Beyond sample application view XML, you can export and import views for other subject areas. Endeca Views facilitate additional data manipulation in Endeca Server using Endeca Query Language (EQL), SQL-like views which are typically created in Studio but are stored in the server and need to be exported and then imported. Eight Subject Areas, or sample applications, have the Endeca Web Service Procedure included in the ODI package to load Endeca views.

To export views:

1. Create a new ODI package and add the Endeca Web Service Procedure. Give the package a name, for example Export Views.
2. Set the following Procedure options:
  - ENDECA\_WS\_RELATIVE\_PATH: /ws/sconfig/<Data Domain Name>
  - REQUEST\_FILE\_NAME: Export\_Views.xml
  - RESPONSE\_FILE\_NAME: Name of the response message file views export. The naming convention is <Data Domain Name>\_Views.xml.
3. Place or save the Export\_Views.xml file into the directory entered in the BIAPPS\_OEID\_FILE physical schema. This is also the directory where the procedure outputs the response file (RESPONSE\_FILE\_NAME).
4. Run the ODI Package. If successful, the views are exported into the response file.

To import views:

1. The exported views xml requires formatting before it can be imported or re-imported. Open the xml in a text editor and search for validatedSemanticEntity and replace it with semanticEntity.
2. Search for listEntitiesResponse and replace it with putEntities.
3. Use a new or existing ODI package to load the Endeca Data Domain. In the package, add the Endeca Web Service Procedure as the final step.
4. Set the following Procedure options:
  - ENDECA\_WS\_RELATIVE\_PATH: /ws/sconfig/<Data Domain Name>
  - REQUEST\_FILE\_NAME: Name of the response message file from the views export. The naming convention is <Data Domain Name>\_Views.xml.
5. Place or save the Export\_Views.xml file into the directory entered in the BIAPPS\_OEID\_FILE physical schema. This is also the directory where the procedure outputs the response file (RESPONSE\_FILE\_NAME).
6. Run the ODI Package. If successful, the views are imported into the target data domain.

**Verify Connection to the Data Warehouse**

Verify that the Oracle Business Analytics Warehouse data server has correct connection information.

1. In ODI Designer's Topology tab, expand the Technology in the Physical Architecture corresponding to the data warehouse.
2. Select the BIAPPS\_DW data server.
3. In the Definition tab, enter the connection user name and password for the OBAW database.
4. In the JDBC tab, enter the correct JDBC Driver information for the database.

**Configure the ODI Agent**

ODI Agents exist as either a standalone agent or a Java EE agent. The Endeca IKM jar files, which must be included with the ODI Agent, are available in the BI Applications installation in `<Middleware Home>\Oracle_BI1\biapps\admin\provisioning\endeca\OracleBIApps_Endeca.zip` (Import/Lib directory).

If your ODI agent is a standalone agent, copy the JAR files into the ODI drivers directory located in `<install path>\Oracle\Middleware\Oracle_ODI1\oracledi\agent\drivers` (or the equivalent path on UNIX). Restart the ODI Agent afterwards.

If your ODI agent is a Java application running on the WebLogic Server, do the following:

1. Create a jlib directory under `<ORACLE_HOME>biapps/odi/bia_odiagent/`.
2. Copy all the required Endeca IKM jars in the odi agent classpath, in `<ORACLE_HOME>biapps/odi/bia_odiagent/jlib`.
3. Restart the ODI Server.

For more information about the IKM and jar files, refer to the *Endeca SQL to Endeca Server Installation and Usage Guide*.

**Disable Applying Schemas in the Load Target Schema ODI Interface**

Endeca Server schema metadata can be modified using Endeca Studio rather than editing ODI flexfields after the schema is created during the initial Endeca ETL. To prevent overwriting a customized schema, you set the APPLY\_SCHEMA option in the Load Target Schema Interface to FALSE.

1. In ODI Designer, under Projects, open the Package of the subject area and select the Diagram tab.
2. Right-click the TLP\_OEID\_<Subject Area Name>Load\_Tgt\_Schema step and select Edit Linked Object. The Load Target Schema Interface opens in a new tab.
3. In the Flow tab, click the Target (BIAPPS\_OEID). Change the APPLY\_SCHEMA option in the Properties Inspector options area to 'false'.
4. Save and close the interface, then the package.
5. Generate the scenario. In Projects, right-click the Package and select Generate Scenario, and click OK to accept the defaults.
6. Note the version of the new scenario.

You can now run the scenario directly using the noted version number, selecting the Context, Logical Agent, and Log Level. Alternately, you can run the scenario from the

Endeca load plan. For each of the scenario steps, the load plan sets the version -1, which uses the latest generated scenario. Scenarios are run directly in ODI Designer.

### 5.3.4 Setup Step: Load Data

You can load data into the Endeca domain either by running ODI scenarios directly or by running the Endeca load plan. Source and target data stores delivered with the BI Applications installation include all Oracle BI Applications repository (RPD) logical columns. These may be customized to add or remove mappings to meet business requirements.

#### Setting and Declaring Variables

When running scenarios directly, you can set session variables at runtime. The following ODI variables can be overridden at runtime.

- ENDECA\_DATADOMAIN\_PROFILE: This variable is the name of the Data Domain Profile if one was created. For information about creating a Data Domain Profile to maximize performance, refer to "[Creating Endeca Data Domain Profile to Optimize CPU Core Count](#)".
- ENDECA\_OBIEE\_SQL\_PREFIX: This variable is used to override the default preferred currency, which is GLOBAL1.
- ETL\_PREDICATE\_EXTRACT: This variable is used to enter filters.

The following ODI variables require modifications to the ODI package.

- ENDECA\_DATADOMAIN
- ENDECA\_DATALOAD\_LOG\_FILE\_PATH

To change the ENDECA\_DATADOMAIN and/or ENDECA\_DATALOAD\_LOG\_FILE\_PATH values:

1. In the Designer under Projects, open the package of the subject area and select the Diagram tab.
2. Select the ENDECA\_DATADOMAIN or ENDECA\_DATALOAD\_LOG\_FILE\_PATH variable. In the Properties pane, in the General tab, enter the desired value.
3. Save and close the Package.
4. Save and close the interface, then the package.
5. Generate the scenario. In Projects, right-click the Package and select Generate Scenario, and click OK to accept the defaults. Select Startup Parameters "Use All" when prompted, then click OK.

#### Run Scenarios Directly

To run ODI scenarios directly:

1. In ODI Designer, open the scenario to run.
2. Click Run.
3. Select the Context, Logical Agent, and Log Level.
4. Use the default or modify the session variables as needed. For example, BIAPPS.ETL\_PREDICATE\_EXTRACT is a customizable filter variable.
5. Click OK.

### Run the Endeca Load Plan

A predefined load plan is provided to run the Endeca packages. The load plan contains steps to evaluate whether the IS\_ENDECA\_DEPLOYED variable is set to 'Y' or 'N' for each of the OEID\_ scenarios, which correlate with Endeca Fact Groups. The frequency with which you run the load plan for data refresh is dependent on your use. Depending on the cost of a typical load, nightly execution may be appropriate, but if you do not require real-time data, you may run it less frequently. In the case that you are performing only historical analysis, the first load may be the only required.

---

**Note:** Configuration Manager does not include an offering for setting up the ODI repository to load Endeca. The load plan is executed directly using the ODI Web Console or Client.

---

### Set IS\_ENDECA\_DEPLOYED Variable and Run ODI Load Plan

To select which scenarios are run, set the IS\_ENDECA\_DEPLOYED variable, then run the load plan.

1. In ODI Client, open the BI Apps DW to Endeca Load Plan under Load Plan and Scenarios > Predefined Load Plans/Endeca.
2. Select the Steps tab and expand the Parallel step.
3. Select the substep with an "OEID\_" prefix containing the scenario to be executed, for example the OEID\_COSTING\_INVENTORY\_VALUATION\_FG, which represents the Costing Inventory Valuation fact group.
4. In the Properties Inspector, under the Variables section, set the value for BIAPPS.IS\_ENDECA\_DEPLOYED to Y. Ensure that the Y is capitalized and that the Refresh checkbox is not selected after entering the value.
5. Repeat the above steps for each OEID\_ substep which contains a scenario to be executed.
6. Save the load plan.
7. Run and monitor the load to execute the substeps which were set to Y.

### Set Filters and Currency Setting

Within the predefined Endeca load plan, variables such as ENDECA\_OBIEE\_SQL\_PREFIX and ETL\_PREDICATE\_EXTRACT may be overridden and saved in the load plan. Filter and currency modifications can be done directly in the load plan under Scenario Variables by selecting the scenario step, and checking and providing a new value.

## 5.3.5 Setup Step: Create a New Endeca Application

### Creating a New Application

To create a new Endeca Application:

1. Log in to Endeca Studio as an Administrator.
2. Click the gear icon and, in the Control Panel, select Endeca Servers.
3. Click New Connection and enter the required parameters.
4. Validate the connection and, optionally, test it.
5. Click Back to Home to return to the home page.

6. Click New Application.
7. Enter an Application name and Application description.
8. Select a pre-built Endeca Server under Select a Data Source.
9. Select a managed data connection.
10. Click Done. A default template is created.

## 5.4 ODI Package, Interface, and Procedure Design Overview

This section describes unique design details about the Endeca ODI package, interface, and procedures. Source and target data stores delivered with the BI Applications installation include all Oracle BI Applications repository (RPD) logical columns. These may be customized to add or remove mappings to meet business requirements.

### New Flexfields

The flexfield properties in the ODI Data Store columns configure the Endeca Server column settings. For example, the Endeca properties define whether the value of a column is text searchable, value searchable, and so on. These properties all begin with the "Endeca Property" as a prefix.

### Packages

Each Subject Area (SA) has the following package design pattern:

- ENDECA\_DATADOMAIN variable: Name of SA Data Domain, for example, OEID\_Project\_Cost.
- ENDECA\_DATADOMAIN\_PROFILE variable: Data Domain profile. If blank, the default Data Domain profile is used.
- ENDECA\_DATALOAD\_LOG\_FILE\_PATH variable: Path/name for Log File, for example OEID\_Project\_Cost\_ENDECA\_DATALOAD.LOG.
- ENDECA\_OBIEE\_SQL\_PREFIX variable: Optional Oracle BI EE prefix, usually set for currency, for example, set variable PREFERRED\_CURRENCY='Global Currency 4'.
- ETL\_PREDICATE\_EXTRACT variable: Optional Oracle BI EE where clause (filter) condition. The default is 1=1.
- TLP\_OEID\_<SA>\_Load\_Tgt\_Schema interface, for example, TLP\_OEID\_Project\_Cost\_Load\_Tgt\_Schema
  - Interface does not load data (has filter set to 1=2).
  - Interface creates the Data Domain if it does not already exist.
  - Mapping contains all of the columns for the Subject Area and loads only the schema.

---

**Caution:** If any columns are modified in the data loading interfaces, this load target schema interface's columns needs to be updated and maintained.

---

- There is an IKM option to disable the "APPLY SCHEMA" option to preserve attribute configurations made in Studio.

- TLP\_OEID\_<SA>\_<Fact Table Name> interfaces, for example, TLP\_OEID\_Project\_Cost\_Fact\_Project\_Cost.
  - Interface or interfaces (some Subject Areas may have more than one) extracts data from Oracle BI EE and loads Endeca.
  - One interface exists per dataset collection name, for example, "Fact\_Project\_Cost"
  - Each interface loads a single denormalized fact star schema.
  - Each interface truncates the collection data before each load, so each load is a destructive load and not incremental.

Each interface uses the following:

- LKM: LKM BIAPPS OBIEE to SQL BMMFETCH
- IKM: IKM SQL to Endeca Server

#### **Data Stores**

- Source: Oracle BI Applications\Oracle BI Applications Business Model\Core
- Target: Oracle BI Applications\Oracle BI Applications Endeca\Core

#### **Endeca Web Service Procedure**

- The procedure requires the same jars used by the SQL to Endeca Server IKM.
- Only Subject Areas that are sample apps have the Endeca Web Service Procedure added to load views as the final step in the Package.
- The Endeca Web Service Procedure can optionally be included in any Subject Area, or even be run standalone in its own ODI Package.

## **5.5 Deploying Sample Applications**

This section describes how to deploy Endeca sample applications, which can be downloaded from the BI Applications 11.1.1.8.1 Media Pack from Oracle Software Delivery Cloud, and are used as a basis to create your own applications. There are eight sample applications, with Data Domains and Studio Applications (.lar files) delivered in the Media Pack under Supporting Files for Endeca Sample Applications for BI Applications 11.1.1.8.1, and View XML files deployed by the BI Applications installer. Notice that some sample applications contain tag clouds based on enriched text, so if the Data Domain is reloaded, rules need to be re-run. If the Data Domain is reset, rule need to be recreated.

#### **Importing Sample Data Domains**

The data domain files included with the installation should be placed in: `$DOMAIN_HOME\EndecaServer\offline`.

The `endeca-cmd` utility can import the data domains and enable them using the following commands:

```
CALL endeca-cmd import-dd my_dd1 --offline-name my_dd_offline1
CALL endeca-cmd enable-dd my_dd1
```

You can use a script to import and enable multiple data domains. The below is an example of a DOS .bat script:

```
CALL endeca-cmd import-dd OEID_Sales_Order_Lines \--offline-name OEID_Sales_Order_Lines
```

```

CALL endeca-cmd enable-dd OEID_Sales_Order_Lines
CALL endeca-cmd import-dd OEID_Project_Performance \--offline-name OEID_Project_
Performance
CALL endeca-cmd enable-dd OEID_Project_Performance
CALL endeca-cmd import-dd OEID_SIA_Admissions_and_Recruiting_Student_Response
\--offline-name OEID_SIA_Admissions_and_Recruiting_Student_Response
CALL endeca-cmd enable-dd OEID_SIA_Admissions_and_Recruiting_Student_Response
CALL endeca-cmd import-dd OEID_SIA_Admissions_and_Recruiting_Application_
Evaluation \--offline-name OEID_SIA_Admissions_and_Recruiting_Application_
Evaluation
CALL endeca-cmd enable-dd OEID_SIA_Admissions_and_Recruiting_Application_
Evaluation
CALL endeca-cmd import-dd OEID_Employee_Expenses_Overview \--offline-name OEID_
Employee_Expenses_Overview
CALL endeca-cmd enable-dd OEID_Employee_Expenses_Overview
CALL endeca-cmd import-dd OEID_Manufacturing_Work_Order_Performance
\--offline-name OEID_Manufacturing_Work_Order_Performance
CALL endeca-cmd enable-dd OEID_Manufacturing_Work_Order_Performance
CALL endeca-cmd import-dd OEID_Human_Resources_Recruitment \--offline-name OEID_
Human_Resources_Recruitment
CALL endeca-cmd enable-dd OEID_Human_Resources_Recruitment
CALL endeca-cmd import-dd OEID_Manufacturing_Actual_Production \--offline-name
OEID_Manufacturing_Actual_Production
CALL endeca-cmd enable-dd OEID_Manufacturing_Actual_Production

```

For information about using the endeca-cmd utility, refer to the chapter titled, "Endeca Server Command Reference" in the *Oracle Endeca Server Administrator's Guide*. For information about importing, exporting, enabling, or disabling data domains, refer to the chapter titled, "Managing Data Domains" in the *Oracle Endeca Server Administrator's Guide*.

### Importing Endeca Studio Application Sample Applications

To import Endeca Studio sample applications, you are required to have created a new application prior to import. To create a new application using the default template:

1. Log in to Endeca Studio as an Administrator.
2. Click the gear icon on the top right to navigate to the Control Panel, then Endeca Servers.
3. Click the New Connection button and input the necessary parameters. Below is an example of typical parameters:

```
Connection ID: OEID_Employee_Expenses_Overview
```

```
{
  "dataDomainName": "OEID_Employee_Expenses_Overview",
  "name": "OEID_Employee_Expenses_Overview",
  "port": "7001",
  "server": "hostname"
}
```

---

**Note:** If applying the custom OBIA Studio security manager for the sample apps, the Connection ID needs to match the Data Domain name to be in sync with the Connection ID specified in the delivered security columns configuration file. For more information about the security columns configuration file, refer to [Section 5.6, "Applying the Custom BI Applications Security Manager"](#).

---

4. Validate the connection, then click Save. Optionally, confirm connectivity using the Test Connection button.
5. Click the Back to Home button to return to the home page.
6. Click the New Application button.
7. Provide an Application name, for example, Employee Expenses, and an Application description.
8. Under Select a Data Source, choose Use a Pre-built Endeca Server.
9. Select a managed data connection. For example, OEID\_Employee\_Expenses\_Overview.
10. Click Done to create a default template.

### **Import Sample Application (.jar file)**

To import a sample application provided in the Media Pack:

1. Click the gear icon on the top right to navigate to the Control Panel, then select Applications.
2. On the line of the application, to import the .jar file, select Actions > Manage Pages.
3. Under All Pages, in the Export/Import tab, select the Import tab.
4. Click Browse or Choose File, and select the appropriate .jar file.
5. Under the section titled What would you like to import, leave the defaults. To delete any pages in the destination environment that do not exist in the .jar file, check the Delete Missing Pages checkbox. To ensure that the import includes all of the component configuration, check the User Preferences checkbox.
6. Click Import.

### **Text Enrichment (Term Extraction) / Enrichment Rules**

Some of the sample applications contain tag clouds based on enriched text. The tag clouds should work after importing sample data domains, because the text enrichment pipelines are stored in the data domains and not in Studio. If the sample application data domains are loaded without being imported, the enriched tag clouds display an error message due to missing enriched attributes. The text enrichment pipelines need to be recreated. To do this:

1. In Application Settings > Data Sets > Enrichments, click Add Enrichment, extract terms, and select the attribute or attributes containing free form text (unstructured content) to be enriched
2. Provide an output name, for example Attribute Name (terms). The rest of the settings can remain defaults.
3. Click the Run button to process the enrichment.
4. Edit the tag clouds settings to replace the invalid output attributes with the new ones.

Whenever data is replaced or reloaded into the Endeca Data Domain, the tag clouds and enrichment pipelines should still work and do not need to be recreated, but the enrichment pipeline needs to be run again by clicking on the run button as described above. This recreates the enriched data. Otherwise, the enriched tag clouds do not show any data. To rerun enrichment:



1. Navigate to Application Settings > Data Sets and select the datasets which have enrichment pipelines.
2. Select the Enrichment tab and run the Enrichment or Enrichments by clicking the Run button.
3. Click the Run button to process the enrichment.

### Refinement Rules

For applications that contain multiple data sets, refinement rules allow you to connect attributes from the different data sets. For example, a sales data set for automotive data includes the make and vehicle ID number (VIN) of cars that were sold. Another data set containing warranty claims also includes the make and VIN of cars for which warranty claims were filed. If you then create refinement rules for the make and VIN attributes, then when users refine one data set by a make or VIN, the other data set also is refined by that make or VIN.

For more information about using refinement rules, refer to the chapter titled, "Using Refinement Rules to Link Attributes from Different Data Sets," in the *Oracle Endeca Information Discovery Studio User's Guide*.

## 5.6 Applying the Custom BI Applications Security Manager

This section describes how to set up Endeca security for interoperability with Oracle BI Applications and Oracle WebLogic. It includes information about assigning required BI roles, creating the Endeca credential store and including the files required by the Oracle BI Applications Security Manager, defining new users in Endeca Studio, and other optional security configurations you can make to enhance Endeca Server performance or user experience.

### Assigning BIImpersonator Role to an OBIEE user

The Oracle BI Applications Security Manager requires an Oracle BI Enterprise Edition user with administrator and impersonator roles, which is used to obtain information necessary to apply security filters in an Endeca Application. To add the impersonator role to an existing Oracle BI EE administrator account:

1. Log in to Oracle Business Intelligence's Enterprise Manager with Administrator privileges.
2. Expand Business Intelligence in the left-hand pane.
3. Right-click coreapplication and select Security > Application Roles to navigate to the Application Roles page.
4. By default, the obi application stripe is selected and the default application roles are displayed. Search for the Role Names with a prefix of "BI".
5. In the Members list, click the BIImpersonator role, then click Edit.
6. Click Add and, in the Add Principal dialog box, search for Type of User and locate an administrative user.

### Creating and Setting Up the Credential Store in Endeca Studio

The Oracle BI Enterprise Edition account credentials with administrator and impersonator roles is saved locally in the Endeca Studio Domain's Credential Store. The BI Applications Security Manager obtains the password information from the Credential Store to connect to Oracle BI Enterprise Edition using JDBC. To set up the credential store, follow the steps below, which assume WebLogic has already been

installed and occur before creating an Endeca Studio domain. It is possible to extend an existing domain to include Enterprise Manager.

1. Verify that Oracle Application Development Framework was installed as part of the Endeca Server installation. ADF may already be available if Endeca Studio is installed on the same WebLogic Server.
2. When creating the Endeca Studio Domain for the first time, check "Oracle Enterprise Manager". JRF is automatically included. If a Studio Domain has already been created, the existing domain can be extended to include Enterprise Manager.
3. Start WebLogic and create a Credential Store to save the OBIEE credential information using Enterprise Manager.
  - a. Log in to Enterprise Manager and, in the WebLogic Domain, right-click `endeca_studio_domain > Security > Credentials`. Create a new map (for example, `oracle.bi.enterprise`) and key (for example, `repository.OBIA`), then save the OBIEE username and password information.

---

---

**Note:** The password assigned to the map and key must match the OBIEE account with administrator and impersonator roles assigned to it.

---

---

### Including the Oracle BI Applications Security Manager Related Files

The Oracle BI Enterprise Edition account credentials with administrator and impersonator roles are saved locally in the Endeca Studio Domain's Credential Store. The BI Applications Security Manager obtains the password information from the Credential Store to connect to Oracle BI Enterprise Edition using JDBC. To set up the credential store, follow the steps below, which assume WebLogic has already been installed and occur before creating an Endeca Studio domain. It is possible to extend an existing domain to include Enterprise Manager.

For the WebLogic version of Endeca Studio 3.1, the custom BI Applications Security Manager .jar files need to be added to the .ear installation file. Decompress the .ear file using a utility, then copy the .jar files into the `\APP-INF\lib\` directory. To copy the files:

1. Save a copy of the Endeca Studio installation .ear file with a different file name, for example, `OBIA-endeca-portal-weblogic-3.1.13849.ear`.
2. Add `OBIAMDEXSecurityManager.jar` and `bijdbc.jar` to Endeca Studio's .ear file under `\APP-INF\lib\` using a compression utility.
3. Deploy the .ear into WebLogic following the installation instructions. For information about deploying the .ear, refer to the section titled, "Deploying Studio to the WebLogic domain" in the *Oracle Endeca Information Discovery Studio Installation Guide*. If there is an existing Endeca Studio deployment, be sure to undeploy it first. Note the name used for the deployment, as this value will be used when modifying the `system-jazn-data.xml`. A typical name for the deployment is `OBIA-endeca-portal-weblogic-3`.
4. Add directories on the WebLogic server named `XML` and `User_Input` under `$MW_HOME\user_projects\domains\endeca_studio_domain`.
5. Add or create the `config.properties` file in the `User_Input` directory. Set the parameters as follows. A sample file is available in `<Middleware Home>\Oracle_BI1\biapps\admin\provisioning\endeca\OracleBIApps_Endeca.zip (OBIAMDEXSecurityManager/User_Input/`.

```

OBIEE_HOST=<OBIEE hostname>
OBIEE_USERID=<OBIEE username with Admin and Impersonator Roles assigned>
OBIEE_JDBC_PORT=<port number, usually 9703>
OBIEE_USERID_MAP=<Credential Store Map Name>
OBIEE_USERID_KEY=<Credential Store Key Name>

```

6. Add or create the securitycolumns.csv file in the User\_Input directory. A sample file is available in <Middleware Home>\Oracle\_BI1\biapps\admin\provisioning\endeca\OracleBIApps\_Endeca.zip (OBIAMDEXSecurityManager/User\_Input/. This .csv file has three columns, Endeca Server Connection ID, Collection Name, and Security Columns, used to relate the security columns.
7. Start and log in to Endeca Studio. From the Control Panel, select Framework Settings and set df.mdexSecurityManager from "com.endeca.portal.data.security.DefaultMDEXSecurityManager" to "com.endeca.portal.extensions.OBIAMDEXSecurityManager".
8. Click Update Settings.
9. Shut down the WebLogic server.
10. Add the following entry to the system-jazn-data.xml file found in \$DOMAIN\_HOME\config\fmwconfig, under the <system-policy> and <jazn-policy> tags. This includes permissions for Studio to access the Credential Store. Examples of

```

<grant>
<grantee>
<codesource>
<url>file:${oracle.deployed.app.dir}/<appName>${oracle.deployed.app.ext}</url>
</codesource>
</grantee>
<permissions>
<permission>
<class>oracle.security.jps.service.credstore.CredentialAccessPermission</class>
<name>context=SYSTEM,mapName=<mapName>,keyName=<keyName></name>
<actions>*</actions>
</permission>
</permissions>
</grant>

```

Examples of values include:

- appName=OBIA-endeca-portal-weblogic-3
  - mapName=oracle.bi.enterprise
  - keyName=repository.OBIA
11. Restart the WebLogic Endeca Studio Server with the new custom Security Manager applied and in use by Endeca Studio.

### Optional: Increasing WebLogic Server Heap Space to Improve Endeca Studio Performance

Update the setDomainEnv script file, which is named setDomainEnv.cmd in Windows environments and setDomainEnv.sh in Linux. The file is located in the bin subdirectory of the domain directory, <MiddlewareHomeDirectory>/user\_projects/domains/endeca\_studio\_domain/bin/.

Search for the following in the setDomainEnv.cmd file:

```

if NOT "%USER_MEM_ARGS%"==" " (
set MEM_ARGS=%USER_MEM_ARGS%

```

)

Before the above "if" statement, add the following, which sets a higher -Xmx or maximum memory heap size):

```
set MEM_ARGS=-Xms128m -Xmx1280m %MEM_DEV_ARGS% %MEM_MAX_PERM_SIZE%
```

### Optional: Enabling Verbose Debugger Logging

To enable logging of debugging messages in the log file located in \$MW\_HOME\user\_projects\domains\endeca\_studio\_domain\eid-studio.log, select Server Administration in the Control Panel, then select the Log Levels tab. Select the Add Category tab and enter the following:

- com.endeca.portal.extensions.OBIAMDEXSecurityManager (DEBUG)
- com.endeca.portal.extensions.OBIAMDEXSecurityManager.BIHandlers (DEBUG)

Log out, then log back in to enable the changes. This step has to be repeated if the Endeca Studio server is restarted.

### Overriding Screen Name Validator in Endeca Studio

By default, Endeca Studio does not allow screen names to contain underscores. The screen name validator must be changed from the DefaultScreenNameValidator to the LiberalScreenNameValidator.

1. Shut down Endeca Studio.
2. Open the portal-ext.properties file under <%WLS\_HOME%\user\_projects\domains\endeca\_studio\_domain\eid\studio. Back up the existing portal-ext.properties file.
3. Add the following at the bottom of the file and save it:
 

```
users.screen.name.validator=com.liferay.portal.security.auth.LiberalScreenNameValidator.
```
4. Start Endeca Studio.

### Defining New Users in Endeca Studio (Must Match OBIEE Userid to Obtain Role Security Info) and Adding Users to Studio Applications

The Custom Oracle BI Applications Security Manager applies filters in Endeca based on the user's application role information set in Oracle Business Intelligence Enterprise Edition and the security columns defined in securitycolumns.csv.

For an Oracle BI EE user, a new user account must be created in Endeca Studio where the screenname matches their BI EE user ID. For information about how to create a new user in Endeca Studio, refer to the chapter titled, "Creating and Editing Users in Studio" in the *Oracle Endeca Information Discovery Studio Studio Administration and Customization Guide*.

Optionally, you can also change the default behavior, which is to have Endeca users log in using their email addresses, so that logins are consistent with Oracle BI Enterprise Edition. For information about changing this default behavior, refer to the section titled, "Configuring the type of user name for Studio" in the *Oracle Endeca Information Discovery Studio Administration and Customization Guide*. For more information about users, roles, and application permissions in Endeca Studio, refer to the "Configuring and Removing Applications" chapter in the *Oracle Endeca Information Discovery Studio Studio Administration and Customization Guide*.

### **Adding Users and Managing Studio Applications Permissions in Endeca Studio**

Users must be added to a Studio Application to enable them to view it. For information about how add users to Studio Applications in Endeca Studio, refer to the section titled, "Adding and removing application members" in the *Oracle Endeca Information Discovery Studio Studio Administration and Customization Guide*.

By default, users are able to create new applications. To prevent this ability for a user, remove the Power User role. For information about removing this role, refer to the section titled, "Preventing a user from creating applications" in the *Oracle Endeca Information Discovery Studio Studio Administration and Customization Guide*.

The application type determines whether an application is visible to users on the Discovery Applications page, and can be set to either Public or Private. To change this value, refer to the section titled, "Configuring the application type" in the *Oracle Endeca Information Discovery Studio Studio Administration and Customization Guide*. You can also control the visibility of pages within an application. To manage page visibility, refer to the section titled, "Configuring the visibility type for a page" in the *Oracle Endeca Information Discovery Studio Studio Administration and Customization Guide*.

## **5.7 Troubleshooting**

This section provides troubleshooting notes and guidelines.

### **Troubleshooting ETL**

To troubleshoot ETL, view logging information in the ODI Operator tab. Key areas of interest include row counts, logical SQL for the business model mapping, and the timing of steps in the ETL. To troubleshoot logical SQL, obtain the SQL from the Operator, then run it in IssueSQL in Log Level 2 and obtain the SQL query from the Oracle BI Server's log, nquery.log.

Note that repository metadata changes may impact ETL. If columns of joins are modified in the RPD, this may impact data extracted by ODI. Also, ROW\_WID = 0 rows may cause less or no data to be returned if a fact table is joined to a dimension table missing a ROW\_WID = 0 row.



---

---

## Researching Data Lineage

This chapter explains how to set up and use data lineage dashboards.

This chapter contains the following sections:

- [Section 6.1, "Introduction"](#)
- [Section 6.2, "Setting Up Data Lineage"](#)
- [Section 6.3, "Tasks for Setting Up Data Lineage Dashboards"](#)
- [Section 6.4, "Tasks for Loading and Refreshing Data Lineage Dashboards"](#)
- [Section 6.5, "Performing Analysis with Data Lineage Dashboards"](#)

### 6.1 Introduction

Data lineage dashboards provide reporting on out-of-the-box Business Intelligence Application module metrics and data, allowing data analysis from the transactional source application through the Business Intelligence repository and the underlying ETL mappings. Using data lineage dashboards, business analysts and those customizing ETL and repository objects can gain insight into where the data in their applications' reports and dashboards is being sourced from, how it is modeled in the Oracle BI Applications repository, and how it is loaded during ETL.

Data lineage dashboards are useful in performing business analysis and in understanding source-to-target ETL mappings and data transformation logic when planning or implementing Oracle Business Analytics Warehouse customizations.

### 6.2 Setting Up Data Lineage

#### Overview

When you set up data lineage, prebuilt data lineage warehouse tables in the OBAW are populated by an ETL package which loads lineage metadata from five sources:

- Oracle BI Applications Configuration Manager
- Oracle Data Integrator (ODI) Work Repository
- Oracle BI Presentation Catalog
- Oracle BI metadata repository file (RPD)
- Oracle Fusion OLTP tables

One-time preliminary setup steps include preparing metadata from and access to these sources for load of data lineage information into the prebuilt data lineage warehouse tables. Metadata and data refresh can then be performed on an ongoing basis.

### List of Steps for Setting Up Data Lineage

To install and set up data lineage dashboards, you must complete the following tasks, in order. Each high-level task breaks down into a list of detailed steps.

1. Configure the environment to support data lineage, as described in [Section 6.3.1, "Setup Step: Configure ODI Topology and Load Plan."](#)
2. Configure the WebLogic Server heap size, as described in [Section 6.3.2, "Setup Step: Configure the WebLogic Server Heap Size."](#)
3. Optionally, disable the Fusion step if you are using other BI Applications sources, as described in [Section 6.3.3, "Setup Step: Disable Fusion Step \(Optional\)."](#)
4. Create data lineage warehouse tables, as described in [Section 6.3.4, "Setup Step: Create the Data Lineage Warehouse Tables."](#)
5. Extract metadata using Windows OBIEE clients, as described in [Section 6.4.1, "Extracting Oracle Business Intelligence Metadata Using Catalog Manager and Administration Tool."](#)
6. Execute the data lineage load plan in ODI to load the data lineage warehouse tables, as described in [Section 6.4.2, "Executing and Monitoring the Data Lineage Load Plan."](#) This step loads the data lineage warehouse tables.

## 6.3 Tasks for Setting Up Data Lineage Dashboards

This section provides detailed tasks for setting up data lineage dashboards. It includes the following sections:

- [Section 6.3.1, "Setup Step: Configure ODI Topology and Load Plan"](#)
- [Section 6.3.2, "Setup Step: Configure the WebLogic Server Heap Size"](#)
- [Section 6.3.3, "Setup Step: Disable Fusion Step \(Optional\)"](#)
- [Section 6.3.4, "Setup Step: Create the Data Lineage Warehouse Tables"](#)

**Note:** You must perform the tasks in this section in the sequence described in [Section 6.2, "Setting Up Data Lineage"](#).

### 6.3.1 Setup Step: Configure ODI Topology and Load Plan

The ODI Work Repository comes preconfigured with required topology and environment settings to support data lineage data extraction from the dashboards' five sources. During initial setup, you configure prebuilt data lineage sources in the ODI topology so that ODI, Configuration Manager, and Oracle Business Intelligence Enterprise (OBIEE) data can be sourced during data lineage ETL. You then configure a prebuilt data lineage home directory variable which provides access to extracted metadata files and configure an adaptor list variable.

The data lineage ETL uses the following connections to extract metadata from its sources and load it into the data warehouse:

- BIAPPS\_ODIREP: Used to extract ODI Metadata from ODI schema where SNP\_\*\*\* tables are located.
- BIAPPS\_BIACOMP: Used to extract configuration metadata from the Configuration Manager schema where C\_\*\*\* tables are located.
- BIAPPS\_DW: Used to load data lineage tables located in the OBAW.

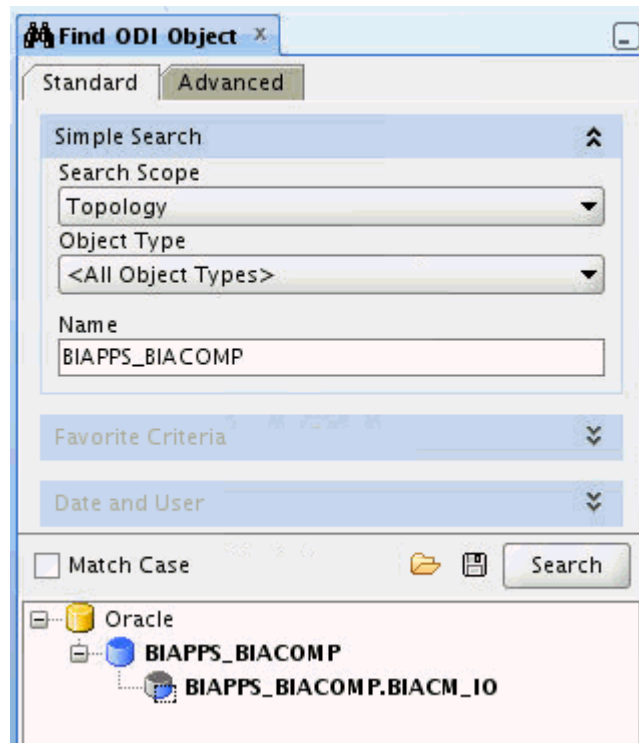


- **BIAPPS\_DW\_FILE:** Used to extract Oracle Business Intelligence Enterprise (OBIEE) metadata from files. This connection should point to the BIA\_11 location (<source file home>/biapps/etl/data\_files/src\_files/BIA\_11)

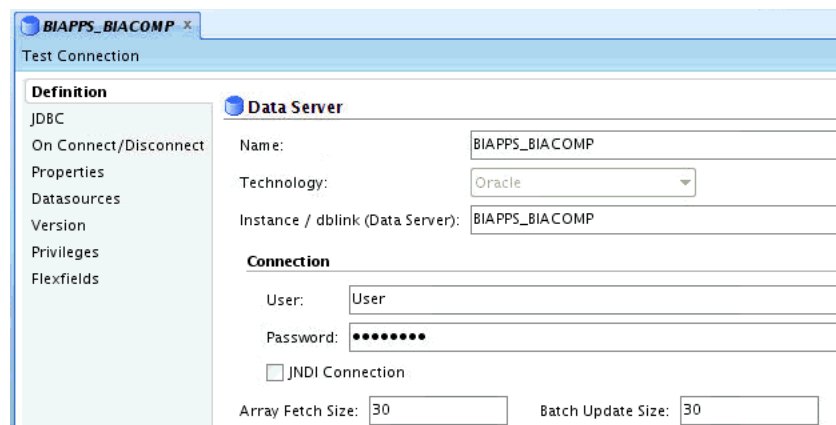
### Configure Data Lineage Source and Target Connections

To configure the data lineage source and target connections:

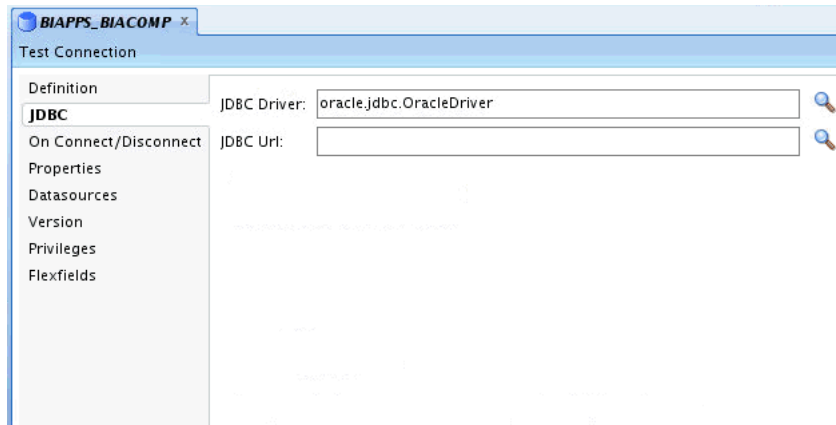
1. In ODI Studio, navigate to the Physical Architecture view of the Topology Navigator's tree view, and search for the prebuilt BIAPPS\_BIACOMP Data Server. The BIAPPS\_BIACOMP connection to the Configuration Manager database may already have been configured during BI Applications installation and configuration.



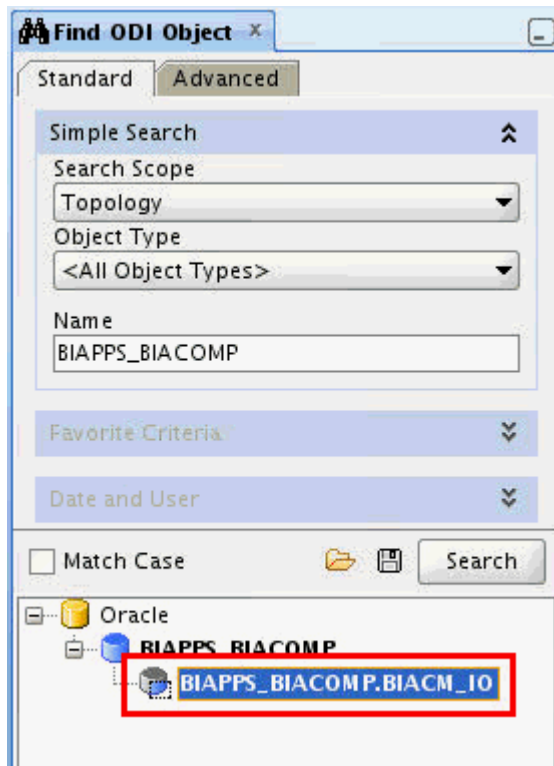
2. In the Definition tab of the editor, verify or enter the correct connection details for the server, including the instance name and a correctly privileged ODI user and password.



3. In the JDBC tab of the editor, verify the JDBC Driver and verify or specify a valid JDBC URL for the connection.



4. Click Test Connection to verify that the connection details you have entered are correct.
5. In the Search pane, double-click the physical schema to open it in the editor.



6. Configure or confirm the Schema and Work Schema.

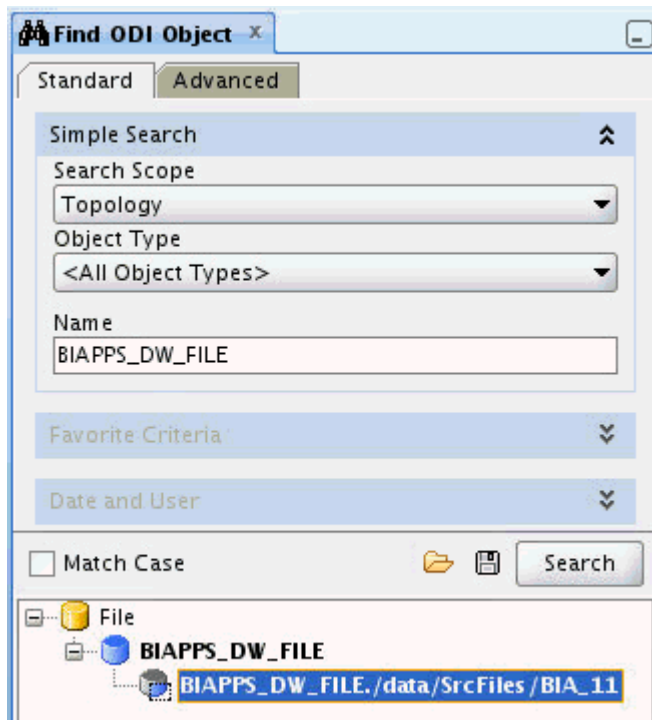
7. Repeat the above steps to configure the other connections:

- Configure the BIAPPS\_ODIREP connection with ODI Repository database and schema details.
- Configure the BIAPPS\_DW connection with OBAW warehouse database and schema details. This connection may already have been configured while installing and configuring BIApps Product.

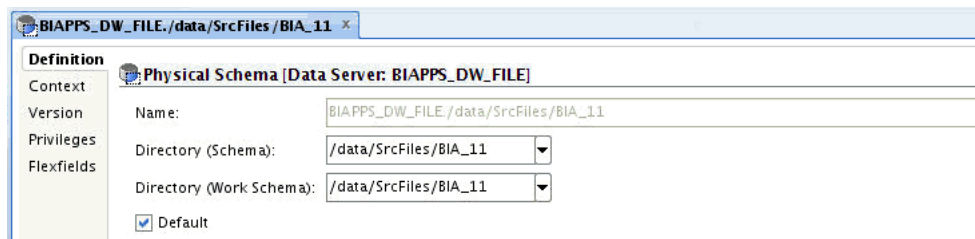
### Configure Data Lineage File Connection

To configure the data lineage file connection:

1. Search for the prebuilt BIAPPS\_DW\_FILE and double-click it to open it in the editor.

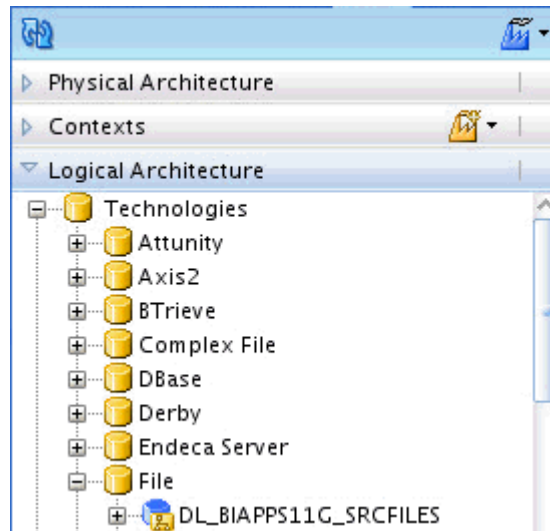


2. Configure Schema and Work Schema to point to the BIA\_11 location, *<source file home>/biapps/etl/data\_files/src\_files/BIA\_11*.

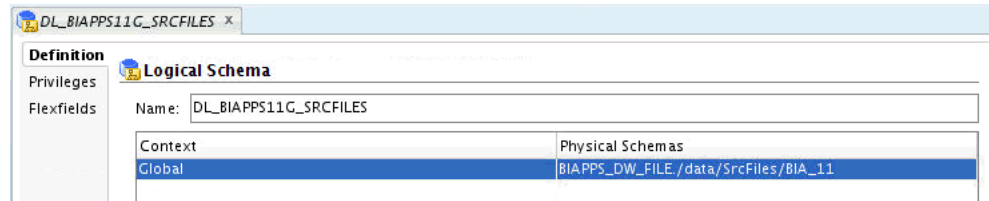


Open Logical File Connection DL\_BIAPPS11G\_SRCFILES. It can be located in Topology - Logical Architecture under Technologies -> File.

3. Navigate to the Logical Architecture view of the Topology Navigator's tree view, and locate the Logical File Connection DL\_BIAPPS11G\_SRCFILES under Technologies > File.



- Double-click the connection and, in the editor, associate DL\_BIAPPS11G\_SRCFILES to the BIAPPS\_DW\_FILE - Physical Schema.



### Configure LoadPlan Variables

The following Loadplan variables must be configured in the Data Lineage Extract and Load Loadplan.

- "DL\_HOME: Location of data lineage source files. Configure to *<source file home>/biapps/etl/data\_files/src\_files/BIA\_11*, the same as the BIAPPS\_DW\_FILE configured in a previous step.
- ADAPTOR\_LIST - List of required Source Adaptor Names.

Based on your source system, one or more values can be selected from the following list:

'SDE\_ORA11510\_Adaptor','SDE\_ORAR1212\_Adaptor','SDE\_PSFT\_90\_Adaptor','SDE\_OP\_V1\_Adaptor','SDE\_ORAR1211\_Adaptor','SDE\_PSFT\_91\_Adaptor','SDE\_SBL\_822\_Adaptor','SDE\_SBL\_811\_Adaptor','SDE\_ORAR12\_Adaptor','SDE\_ORAR1213\_Adaptor','SDE\_JDEE1\_91\_Adaptor','SDE\_JDEE1\_90\_Adaptor','SDE\_Universal\_Adaptor','SDE\_FUSION\_V1\_Adaptor'

The Adaptor Name has to be enclosed in single quotes. If you want to enter multiple adaptors, use commas as separators between multiple adaptor names.

- ADAPTOR\_LIST1 - This is the same as ADAPTOR\_LIST. If you want to configure multiple adaptors and ADAPTOR\_LIST value has reached limits, use ADAPTOR\_LIST1. Otherwise, copy the ADAPTOR\_LIST value to ADAPTOR\_LIST1.

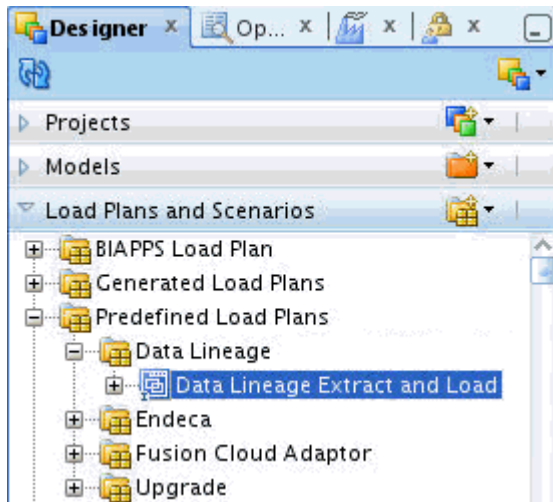
---

**Note:** Do not leave ADAPTOR\_LIST1 value empty. If the variable is empty, it will cause ETL failure.

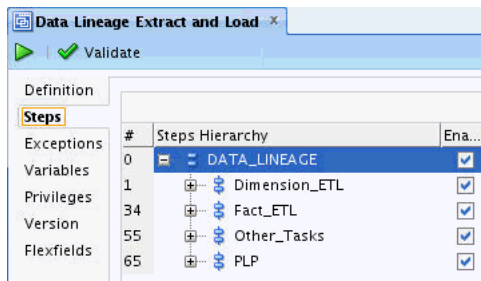
---

To configure the LoadPlan variable:

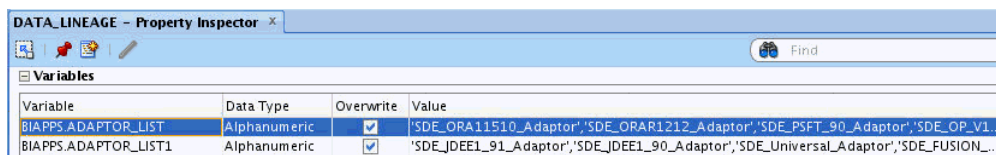
1. In ODI Designer's tree view, navigate to the Load Plans and Scenarios view and open the Data Lineage Extract and Load load plan under Predefined Load Plans > Data Lineage.



2. In the Steps tab, click the root step, DATA\_LINEAGE.



3. The Property Inspector window lists all variables. Enable the Overwrite option and type a value for DL\_HOME, ADAPTOR\_LIST and ADAPTOR\_LIST1. Enclose Adaptor Names in single quotes and, if you want to enter multiple adaptors, use commas as separators between multiple adaptor names.



### 6.3.2 Setup Step: Configure the WebLogic Server Heap Size

To set the heap size in Windows:

1. Open the <DOMAIN\_HOME>/bin/setDomainEnv.cmd file for editing.
2. Locate the if NOT "%USER\_MEM\_ARGS%"==" ( line, and add the following code before it:

```
if "%SERVER_NAME%"=="odi_server1" (
    set USER_MEM_ARGS=-Xms512m -Xmx3072m -XX:PermSize=256m -XX:MaxPermSize=512m
)
```

To set the heap size in Linux:

1. Open the `<DOMAIN_HOME>/bin/setDomainEnv.sh` file for editing.
2. Locate the `if [ "${USER_MEM_ARGS}" != " " ] ; then` line, and add the following code before it:

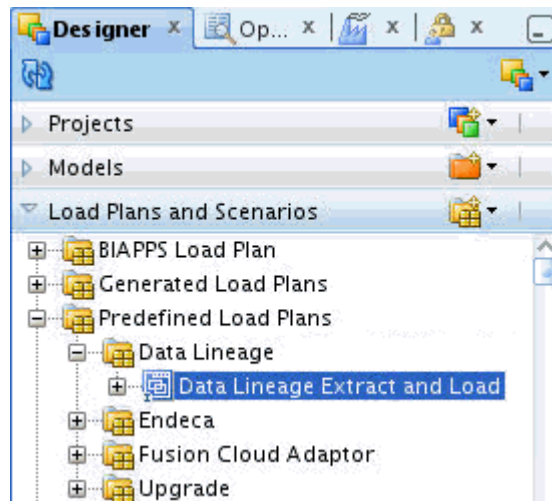
```
if [ "${SERVER_NAME}" = "odi_server1" ] ; then
    USER_MEM_ARGS="-Xms512m -Xmx3072m -XX:PermSize=256m -XX:MaxPermSize=512m"
    export USER_MEM_ARG
fi
```

### 6.3.3 Setup Step: Disable Fusion Step (Optional)

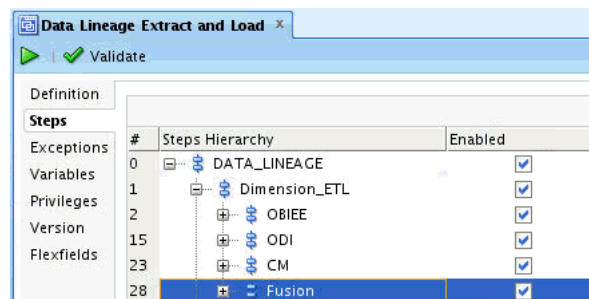
If you are not using a Fusion source for the data warehouse, you must disable Fusion steps in the Data Lineage ETL. This step is applicable only for non-Fusion sources. If you use Fusion as a source, you can skip this step.

To disable Fusion steps:

1. In ODI Designer's tree view, navigate to the Load Plans and Scenarios view and open the Data Lineage Extract and Load load plan under Predefined Load Plans > Data Lineage.



2. In the Steps tab, in the Steps Hierarchy, expand DATA LINEAGE > Dimension\_ETL.



3. Deselect the Enabled check box for the Fusion step (step 28) to disable the step.

#	Steps Hierarchy	Enabled	Scenario/Variable	Restart
0	DATA_LINEAGE	<input checked="" type="checkbox"/>		Restart from failure
1	Dimension_ETL	<input checked="" type="checkbox"/>		Restart from failure
2	OBIEE	<input checked="" type="checkbox"/>		Restart from failure
15	ODI	<input checked="" type="checkbox"/>		Restart from failure
23	CM	<input checked="" type="checkbox"/>		Restart from failure
28	Fusion	<input type="checkbox"/>		Restart from failure

- Expand DATA\_LINEAGE > Fact\_ETL in the hierarchy and disable the Fusion step (step 52).

#	Steps Hierarchy	Enabled	Scenario/Variable	Restart
34	Fact_ETL	<input checked="" type="checkbox"/>		Restart from failure
35	OBIEE	<input checked="" type="checkbox"/>		Restart from failure
38	ODI	<input checked="" type="checkbox"/>		Restart from failure
50	CM	<input checked="" type="checkbox"/>		Restart from failure
52	Fusion	<input type="checkbox"/>		Restart from failure

- Expand DATA\_LINEAGE > Dimension\_ETL > OBIEE, and enable the SDE\_DL\_OBIEE\_BMM\_HIERARCHY step (step 3).

**Data Lineage Extract and Load** x

Validate

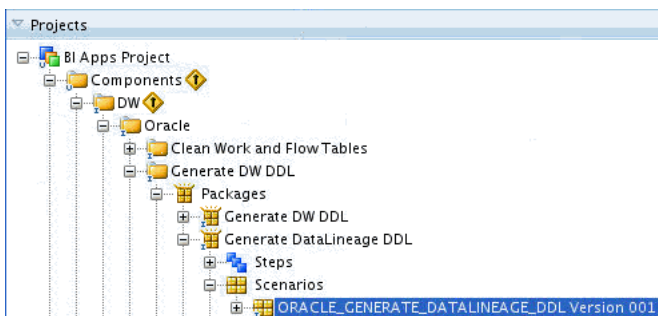
Definition

#	Steps Hierarchy	Enabled	Scenario/Varia..
0	DATA_LINEAGE	<input checked="" type="checkbox"/>	
1	Dimension_ETL	<input checked="" type="checkbox"/>	
2	OBIEE	<input checked="" type="checkbox"/>	
3	SDE_DL_OBIEE_BMM_HIERARCHY	<input checked="" type="checkbox"/>	DATALINEAGE...
4	SDE_DL_OBIEE_SQLQUERY_LIST	<input checked="" type="checkbox"/>	DATALINEAGE...

### 6.3.4 Setup Step: Create the Data Lineage Warehouse Tables

In this step, you use the Generate DataLineage DDL package to create the data lineage tables in the warehouse. These tables are created by default during data warehouse creation, but can be created at any time using this package.

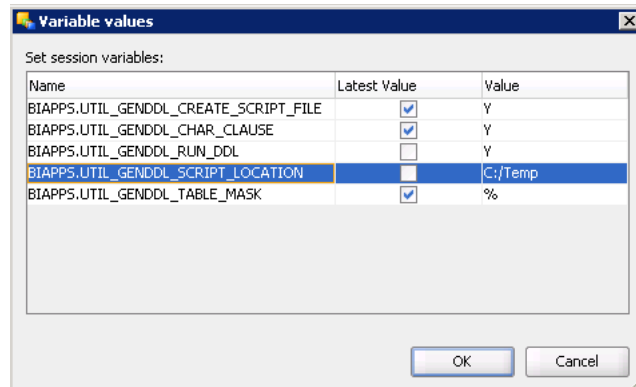
- In the ODI Studio Designer Navigator, expand BI Apps Project > Components > DW > Oracle > Generate DW DDL > Generate DataLineageDDL > Scenarios > ORACLE\_GENERATE\_DATALINEAGE\_DDL.



- Select the ORACLE\_GENERATE\_DATALINEAGE\_DDL and click the Execute (green arrow) button to execute it.
- In the Execution dialog box, choose Agent and click OK.



- In the Variable Values dialog box, change values for the third and fourth variables. Set BIAPPS.UTIL\_GENDDL\_RUN\_DDL to 'Y' and set BIAPPS\_UTIL\_GENDDL\_SCRIPT\_LOCATION to a valid local folder location.



- Click OK in the Variable Values dialog box and monitor the progress in the Operator.

## 6.4 Tasks for Loading and Refreshing Data Lineage Dashboards

This section provides detailed tasks for initial and ongoing loading and refreshing of data lineage dashboards and their required metadata. It includes the following sections:

- Section 6.4.1, "Extracting Oracle Business Intelligence Metadata Using Catalog Manager and Administration Tool."
- Section 6.4.2, "Executing and Monitoring the Data Lineage Load Plan."

**Note:** You must perform the tasks in this section in the sequence described in Section 6.2, "Setting Up Data Lineage".

### 6.4.1 Extracting Oracle Business Intelligence Metadata Using Catalog Manager and Administration Tool

In this step, you extract Presentation Catalog (webcat) and repository (RPD) metadata into files sourced during load of the Data Lineage warehouse tables. You use Windows-based OBIEE clients from versions 11.1.1.6.0 and 11.1.1.7.0 to generate the following files:

- webCat\_dashboard\_text.txt
- webCat\_text.txt
- rpd\_text.txt

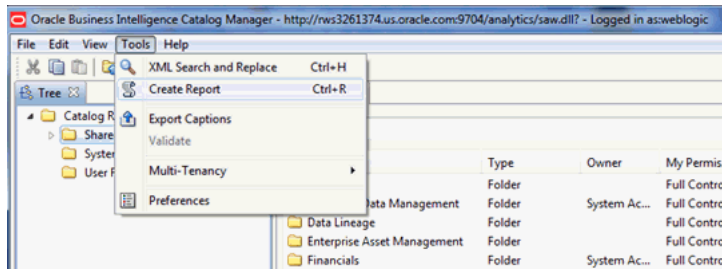
#### Prerequisites

OBIEE 11.1.1.6.0 and 11.1.1.7.0 client installations on a Windows machine are a prerequisite to extract OBIEE metadata.

#### Extract Dashboard Metadata

- Open Oracle Business Intelligence Catalog Manager 11.1.1.6.0 and connect using a connection type of online to the Oracle Business Intelligence Server, URL: `http://<Host:port>/analytics/saw.dll`.

2. Select Tools > Create Report.

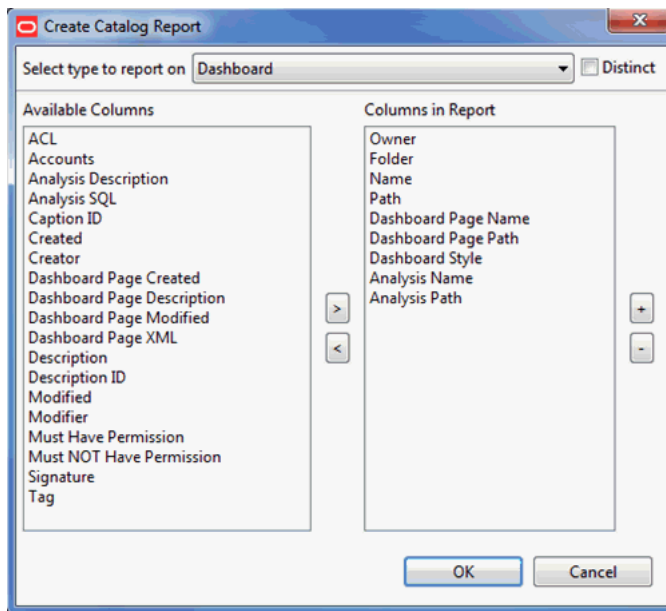


3. In the Create Catalog Report dialog box, select Dashboard in the Select type to report on drop-down list.
4. In the Available Columns list, select the following columns and arrange them in order:
  - Owner
  - Folder
  - Name
  - Path
  - Dashboard Page Name
  - Dashboard Page Path
  - Dashboard Style
  - Analysis Name
  - Analysis Path

---

**Note:** The column order must be as above.

---



5. Click OK. It will take few minutes to generate the report. Click OK if an Error window pops up after few minutes.

6. Save the file as webCat\_dashboard\_text.txt in a local folder.

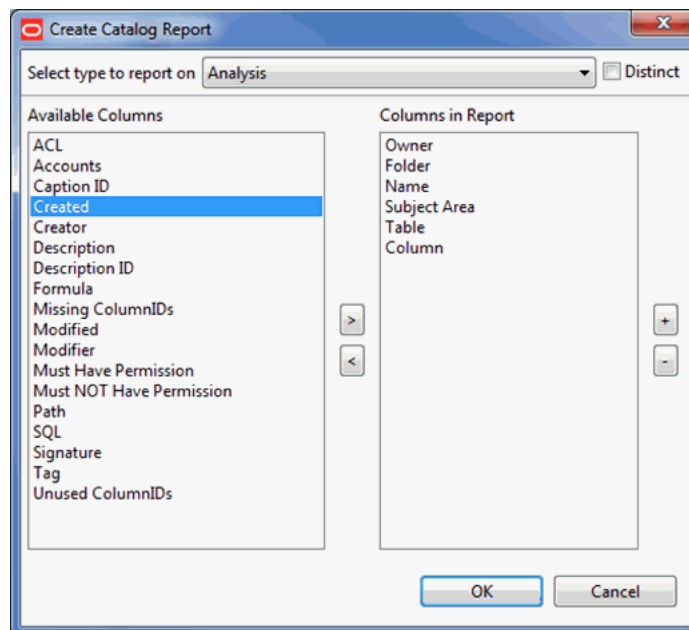
### Extract Report Metadata

1. Select Tools > Create Report.
2. In the Create Catalog Report dialog box, select Analysis in the Select type to report on drop-down list.
3. In the Available Columns list, select the following columns and arrange them in order:
  - Owner
  - Folder
  - Name
  - Subject Area
  - Table
  - Column

---

**Note:** The column order must be as above.

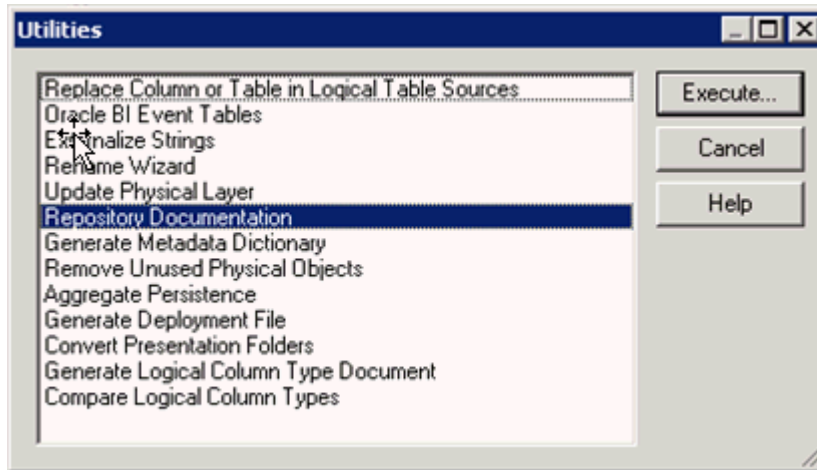
---



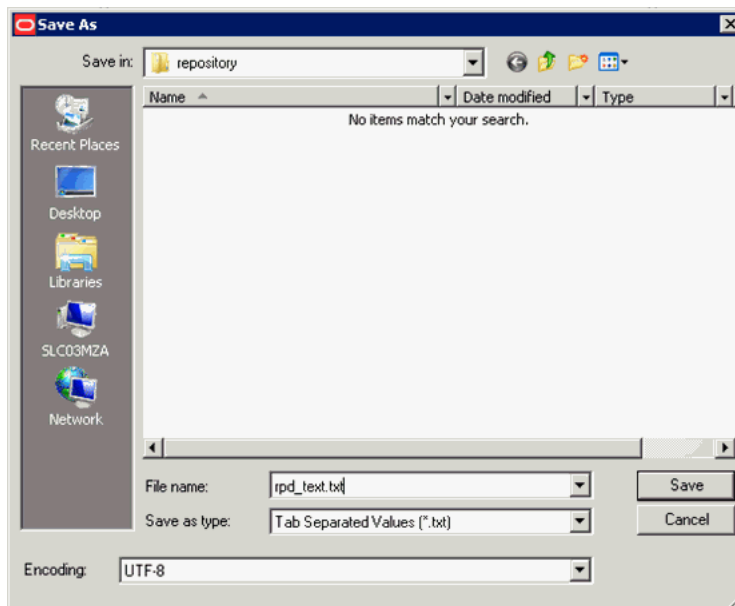
4. Click OK.
5. Save the file as webCat\_text.txt in a local folder.

### Extract RPD Metadata

1. Open the BI Applications RPD using the 11.1.1.7.0 version of Oracle BI Administration Tool.
2. Select Tools > Utilities.
3. In the Utilities dialog box, select Repository Documentation and click Execute.



4. In the Save As dialog box, save the file as rpd\_text.txt. Select Tab Separated Values (\*.txt) in the Save as Type drop-down list.



#### Copy to the \$DL\_HOME directory

In this step, you copy the metadata source files from the local directory on the Windows machine you saved them on to the Data Lineage home directory, where they can be sourced by the Data Lineage Extract and Load load plan. You also copy a Fusion EAR file from its installation location to the Data Lineage home directory.

1. Copy the webCat\_dashboard\_text.txt, webCat\_text.txt, and rpd\_text.txt from the local directory on the Windows machine to \$DL\_HOME.
2. Copy Fusion EAR files from \$ORACLE\_BI\_HOME/biapps/DataLineage to \$DL\_HOME.

### 6.4.2 Executing and Monitoring the Data Lineage Load Plan

In this step, you execute the Data Lineage Extract and Load load plan in ODI. This load plan uses the sources and extracted metadata files you have configured and generated as sources to load and refresh the prebuilt data lineage warehouse tables.

1. In the ODI Studio Designer Navigator, expand Load Plans and Scenarios > Predefined Load Plans > Data Lineage.
2. Execute the DataLineage Extract and Load load plan and monitor its execution using the Operator.

## 6.5 Performing Analysis with Data Lineage Dashboards

This section provides an overview of the BI Applications Data Lineage Dashboard and a case study in its use in Oracle Business Intelligence Applications, and contains the following topics:

- [Section 6.5.1, "Overview of the Oracle BI Applications Data Lineage Dashboard"](#)
- [Section 6.5.2, "Analyzing Data Lineage for Oracle Business Intelligence Applications"](#)

### 6.5.1 Overview of the Oracle BI Applications Data Lineage Dashboard

Data lineage is analyzed across a variety of Oracle BI Applications entities, using dashboard prompts for the following in the DataLineage Summary page of the OBIA DataLineage dashboard:

- Dashboard Name
- Report Name
- Presentation Table Name
- Presentation Column Name
- Logical Table Name
- Logical Column Name
- Warehouse Datastore Resource Name
- Warehouse Column Name
- Source Application
- Source Table Name
- Source Column Name

You can select dashboard prompt values at any level, and subsequent selections are constrained by your other selections, so that, for example, if you have made a selection in the Report Name prompt, any selection from the Logical Table Name prompt will be limited to those included in the selected report.

The OBIA Data Lineage dashboard has five pages, including a summary whose tabular results can be drilled into for detailed lineage reports in detail pages. These pages include:

- **DataLineage Summary:** Provides an end-to-end data lineage summary report for physical and logical relationships. To navigate to detailed reports and dashboard pages for an entity in the Summary report, click its link.
- **Dashboard Implementation:** For a selected dashboard, provides reports detailing: dashboard pages, reports, and Presentation columns; Presentation catalog, tables, and columns; RPD implementation, detailing how a Presentation column is derived from a physical column in the OBAW.

- **Report Implementation:** For a selected report, provides details on the derivation on the Presentation column from its underlying physical column. Also includes reports describing the warehouse tables, and listing the ODI interfaces and adaptors that load both from the OLTP source into the staging tables and from the staging tables into the warehouse dimension and fact tables.
- **Presentation Layer Implementation:** For a selected Presentation table and column, provides details on the logical and physical column derivation in the RPD. Also includes reports describing the warehouse tables, and listing the ODI interfaces and adaptors that load both from the OLTP source into the staging tables and from the staging tables into the warehouse dimension and fact tables.
- **Warehouse Implementation:** For a selected warehouse table name and column name, provides a summary of the warehouse data store and its OLTP source tables and columns. Also includes reports listing the ODI interfaces and adaptors that load both from the OLTP source into the staging tables and from the staging tables into the warehouse dimension and fact tables.

## 6.5.2 Analyzing Data Lineage for Oracle Business Intelligence Applications

This section presents one usage scenario for the OBIA Data Lineage dashboard to illustrate its reports and usage.

To analyze data lineage for Oracle Business Applications:

1. Navigate to the OBIA Data Lineage dashboard.
2. In the Data Lineage Summary page, make a selection from one or more of the available prompts. You can make multiple selections in prompts, and available prompt selections are constrained by earlier selections in other prompts.

The screenshot shows a series of dropdown menus for selecting data lineage parameters. The selected values are as follows:

- Dashboard Name: Account Analysis Repi
- Report Name: Beginning Balances
- Presentation Table Name: --Select Value--
- Presentation Column Name: Fiscal Period
- Logical Table Name: --Select Value--
- Logical Column Name: --Select Value--
- Warehouse Datastore Resource Name: --Select Value--
- Warehouse Column Name: --Select Value--
- Source Application: --Select Value--
- Source Table Name: --Select Value--
- Source Column Name: --Select Value--

At the bottom of the form are two buttons: "Apply" and "Reset".

3. Click Apply to display the OBIA - LineageSummary report, which provides lineage details from the presentation dashboards and reports through BI Applications repository metadata, and finally to the warehouse table and column.
4. To drill to the detailed reports in the Dashboard Implementation page of the dashboard, click the Dashboard Name value in the summary report. This opens the Dashboard Implementation page with the Dashboard Name dashboard prompt pre-selected with the dashboard name. You could also click other entities in the report to navigate to other pages. For example, clicking a Warehouse Column Name and selecting Implementation would navigate to the Warehouse Implementation page, which focuses on ETL implementation. Examine the reports in the Dashboard Implementation page.

Examine the reports in the Dashboard Implementation page:

- The first report in the page, DashboardImplementation- OBIA -LineageSummary, provides a similar lineage to the default LineageSummary

report, tracking lineage from the dashboard through the Presentation catalog to the warehouse table and column.

- The second report, `DashboardImplementation-DashboardDetails`, focuses on the dashboard alone, detailing dashboard pages, reports, the Presentation Catalog, and the associated Presentation table and column names.
  - The third report, `DashboardImplementation-OBIA-RPDImplementation`, focuses on the lineage of the data in the BI repository metadata, covering all three layers of the repository, starting from Presentation Catalog names through business model and logical table and column names, and down to the physical catalog, table, and column names. The logical layer details include the expression, which often is just the logical expression of the table and column resources.
  - The fourth report, `DashboardImplementation-ODIImplementation-SourceToStaging`, focuses on the ETL interfaces used to load the warehouse staging tables, providing the warehouse table and associated adaptor and interface details.
  - The fifth report, `DashboardImplementation-ODIImplementation-StagingToWarehouse`, focuses on the ETL interfaces used to load the warehouse target fact and dimension tables, providing the warehouse table and associated adaptor and interface details.
5. Navigate back to the `DataLineageSummary` page, click a Report name value, and select Report Implementation to open that report as the dashboard prompt value in the Report Implementation page. Scroll through the reports on this page and notice that they closely mirror those provided for dashboards.
  6. Navigate to the Presentation Layer Implementation page, which includes reports detailing the logical and physical column derivation in the RPD for Presentation columns. There are also reports describing the warehouse tables, and listing the ODI interfaces and adaptors that load both from the OLTP source into the staging tables and from the staging tables into the warehouse dimension and fact tables.
  7. Navigate to the Warehouse Implementation page, in which the `WarehouseImplementation-OBIA-LineageSummary` report summarizes the relationship between warehouse tables and columns and associated models and source transactional tables and columns.





---

---

# Administering Oracle GoldenGate and Source Dependent Schemas

This chapter explains how to set up and use Oracle GoldenGate (OGG) and source dependent schemas (SDS).

This chapter contains the following sections:

- [Section 7.1, "Introduction"](#)
- [Section 7.2, "Source Dependent Schema Architecture"](#)
- [Section 7.3, "Setting Up Oracle GoldenGate and Source Dependent Schemas"](#)
- [Section 7.4, "Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema"](#)
- [Section 7.5, "Customization"](#)
- [Section 7.6, "Patching"](#)
- [Section 7.7, "Troubleshooting"](#)

## 7.1 Introduction

In a conventional ETL scenario, data is loaded from source online transaction processing (OLTP) schemas, which in many cases support full-time transactional systems with constant ongoing updates. Contention can arise during complex extracts from these sources, particularly in cases where significant OLTP data changes have occurred which must be processed and loaded by ETL processes.

To relieve this contention, you can set up source dependent schemas which replicate OLTP schemas in the same database as the Oracle Business Analytics Warehouse schema. In addition to segregating extract processing on the analytical system and eliminating contention on transactional systems, physical architecture and ETL performance benefits accrue from maintaining source data in the same physical location as the warehouse tables, consolidating multiple sources, regions and timezones, and eliminating network bottlenecks and incremental change capture during extraction and load.

## 7.2 Source Dependent Schema Architecture

The SDS is a separate schema usually stored on the same database as the Oracle Business Analytics Warehouse, which contains data extracted from an OLTP schema on a separate machine. The OLTP schema is treated as the source and the SDS schema as the target of the Oracle GoldenGate processes which maintain the replicated SDS.

The SDS Architecture is an optional addition to the existing BI Applications Architecture that solves many problems associated with data transport from the source OLTP system to the data warehouse and change data capture required for incremental ETL. The architecture consists of these main components:

- Source Dependent Data Store (SDS): A separate schema on the OBAW database that is a replication of the source OLTP systems tables. Also stores deletes and additional optimizations for incremental ETL.
- Oracle GoldenGate: This replication system is deployed on both source and OBAW database systems. On the source database system, Oracle GoldenGate supports continuous asynchronous change data capture at a low level in the database, then compresses and ships the changed data across the network to the target SDS schema on the analytical warehouse database instance. On the target OBAW database instance, it receives the changed data from one or more source systems and loads them into the target database, specifically into the SDS schemas, one per ETL OLTP source.
- ODI: ODI metadata stores definitions used to generate the SDS schemas and to support the Oracle GoldenGate replication processes.
- BI Application SDS Components: Components used to support generation of the SDS schema and Oracle GoldenGate replication processes.

## 7.3 Setting Up Oracle GoldenGate and Source Dependent Schemas

This section provides instructions for setting up and using Oracle GoldenGate and SDS.

### List of Steps for Setting Up Oracle GoldenGate and Source Dependent Schemas

To install and set up Oracle GoldenGate and SDS, you must complete the following tasks, in order. Each high-level task breaks down into a list of detailed steps provided in the next section.

1. Configure the source and target databases, as described in [Section 7.4.1, "Setup Step: Configure Source and Target Database."](#)
2. Install Golden Gate software, as described in [Section 7.4.2, "Setup Step: Install Oracle GoldenGate on Source and Target Systems."](#)
3. Configure BI Applications Configuration Manager and ODI, as described in [Section 7.4.3, "Setup Step: Configure BI Applications Configuration Manager and Oracle Data Integrator to Support the Source Dependent Schema."](#)
4. Generate and run the Data Definition Language to create the SDS tables on the SDS schema in the target database, as described in [Section 7.4.4, "Setup Step: Generate, Deploy, and Populate the Source Dependent Schema Tables on Target Database."](#)
5. Generate and deploy OGG Parameter files, as described in [Section 7.4.5, "Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines."](#)
6. Start OGG, as described in [Section 7.4.6, "Setup Step: Start Oracle GoldenGate on Source and Target Machines."](#)

## 7.4 Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema

This section provides detailed tasks for setting up Oracle GoldenGate and SDS.

**Note:** You must perform the tasks in this section in the sequence described in [Section 7.3, "Setting Up Oracle GoldenGate and Source Dependent Schemas"](#).

### 7.4.1 Setup Step: Configure Source and Target Database

In this step, you create Oracle GoldenGate database users on source and target databases. Unlike other database schemas used by BI Applications, the SDS and OGG schemas are not automatically created during installation. Only the installation process can automatically create database users; because datasources are defined in Configuration Manager after installation is complete, the required Source Dependent Schemas associated with these datasources must be manually created. For this reason, an SDS schema must be manually defined on the target database. Additionally, the BI Apps installer is not able to create the OGG database user on the source OLTP system. This section describes how to create the OGG database user on the source database system and the OGG and SDS database users on the target database system.

#### Create OLTP Oracle GoldenGate Database User

Each OGG process requires a dedicated database user. On the source system, the OGG user needs to be able to query various metadata. Secure database practice is to avoid granting privileges to tables not in use, so `SELECT ANY TABLE` is not granted to the OGG database user. Instead, as part of the SDS DDL, `SELECT` privileges are granted only to those tables in the OLTP schema being replicated.

The user creation scripts use the following parameters:

Parameter	Description
&BIAPPS_OGG	Oracle GoldenGate Database User Name
&BIAPPS_OGG_PW	Oracle GoldenGate Database User Password

Run the following script on the source database to create the source database OGG user.

```
-- Create OGG User
CREATE USER &BIAPPS_OGG
IDENTIFIED BY &BIAPPS_OGG_PW
DEFAULT TABLESPACE USERS QUOTA UNLIMITED ON USERS;
```

```
GRANT CREATE SESSION TO &BIAPPS_OGG;
GRANT ALTER SESSION TO &BIAPPS_OGG;
GRANT SELECT ANY DICTIONARY TO &BIAPPS_OGG;
GRANT FLASHBACK ANY TABLE TO &BIAPPS_OGG;
```

```
-- OGG user requires ALTER ANY table to set up supplemental logging for individual
tables. Once accomplished, this privilege can be revoked:
GRANT ALTER ANY TABLE TO &BIAPPS_OGG;
```

### Prepare OLTP Database and Redo Logs

Oracle GoldenGate requires that the database be configured for supplemental logging. Execute the following statement in the source database with a user with sufficient privileges.

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

### Create Target Oracle GoldenGate Database User

Each OGG process requires a dedicated database user. On the target system, the OGG user needs to be able to execute various DML operations on the SDS tables as well as optionally create a checkpoint table. Secure database practice is to avoid granting privileges to tables not in use, so `SELECT ANY TABLE`, `INSERT ANY TABLE` and so on are not granted to the OGG database user. Instead, as part of the SDS DDL, required privileges are granted only to those tables in the SDS schema for the OGG database user.

The user creation scripts use the following parameters:

Parameter	Description
&BIAPPS_OGG	Oracle GoldenGate Database User Name
&BIAPPS_OGG_PW	Oracle GoldenGate Database User Password

Run the following script on the target table to create the target database OGG user.

```
-- Create OGG User
CREATE USER &BIAPPS_OGG
IDENTIFIED BY &BIAPPS_OGG_PW
DEFAULT TABLESPACE USERS QUOTA UNLIMITED ON USERS;

GRANT CREATE SESSION TO &BIAPPS_OGG;
GRANT ALTER SESSION TO &BIAPPS_OGG;
GRANT SELECT ANY DICTIONARY TO &BIAPPS_OGG;

-- Create Table privilege only required to create checkpoint table. Can be revoked
once table is created. Not required if not creating this table
GRANT CREATE TABLE TO &BIAPPS_OGG;
```

### Create SDS Database User

A separate SDS database user must be configured in the target database for each OLTP system that will leverage the SDS. Each supported source instance requires a separate SDS schema. The recommended naming convention for the schema owner is `<BIAPPS>SDS<Model Code>_<DSN Number>` where `<BIAPPS>` is a user defined code representing BI Applications content, `<Model Code>` is the unique code assigned to each datasource type and `<DSN Number>` is the unique datasource ID assigned to a specific datasource instance. For example, if you have the following two datasources defined as supported source systems in the BI Applications Configuration Manager you would have the corresponding SDS schemas defined in the data warehouse database:

Source Instance Name	Model Code	Data Source Number	SDS
Oracle EBS 11.5.10	EBS_11_5_10	310	BIAPPS_SDS_EBS_11_5_10_310

Source Instance Name	Model Code	Data Source Number	SDS
Siebel CRM 8.1.1	SEBL_8_1_1	625	BIAPPS_SDS_SEBL_8_1_1_625

Use the following DDL as a template for creating each SDS database user. The following only represents a bare minimum of required DDL statements; adjust for your environment as necessary. Rerun for each supported source instance.

Parameter	Description
&BIAPPS_SDS_DATA_TS	Table space name
&ORADATA	Path where tablespace should be located
&BIAPPS_SDS	SDS User name
&BIAPPS_SDS_PW	SDS User password
&BIAPPS_OGG	Oracle GoldenGate Database User Name

```
-- Create tablespace. Following is only an example and may not reflect PSR
guidance:
CREATE TABLESPACE &BIAPPS_SDS_DATA_TS
DATAFILE '&ORADATA/&BIAPPS_SDS_DATA_TS..dbf' SIZE 100M AUTOEXTEND ON NEXT 10M
LOGGING
DEFAULT COMPRESS FOR OLTP;

-- Create SDS User
CREATE USER &BIAPPS_SDS
IDENTIFIED BY &BIAPPS_SDS_PW
DEFAULT TABLESPACE &BIAPPS_SDS_DATA_TS QUOTA UNLIMITED ON &BIAPPS_SDS_DATA_TS;

-- Required Grants
GRANT CREATE SESSION TO &BIAPPS_SDS;
GRANT CREATE TABLE TO &BIAPPS_SDS;

-- OGG user must be granted Quota to insert and update data
ALTER USER &BIAPPS_OGG QUOTA UNLIMITED ON &BIAPPS_SDS_DATA_TS;
```

## 7.4.2 Setup Step: Install Oracle GoldenGate on Source and Target Systems

Download and install Oracle GoldenGate software first on the source and then on the target machines. The software is available from Oracle Technology Network. For information about installation of Oracle Golden Gate, refer to the Oracle GoldenGate Installation and Setup Guide for your platform and database.

### Installation Recommendations

When installing and configuring the OGG software, consider the following recommendations:

- For each OLTP instance supported, install a separate Replicate process on the target machine. As each OLTP instance has its own separate SDS schema on the target database, the Replicate process is populating different targets so a separate Replicate process is required for each.
- Install a Data Pump process on the source machine.

- The name of the Extract, Data Pump and Replicat processes are limited to eight characters. The suggested naming convention is as follows:

Process	Naming Convention	Example
Extract	EXT_<Datasource Num Id>	EXT_310
Data Pump	DP_<Datasource Num Id>	DP_310
Replicate	REP_<Datasource Num Id>	REP_310

- Follow the steps in the Oracle GoldenGate documentation to configure an instance of OGG on the source and target systems up to the point of starting the OGG processes.
- Note that as part of the installation and configuration, a procedure is run to generate BI Applications-specific parameter files, as discussed in the following section. For more information, see [Section 7.4.5, "Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines."](#) The install and configuration of the OGG processes are completed at this point.

### Example Steps to configure the Oracle GoldenGate processes

The following is a set of example steps to configure the OGG processes, modify as appropriate for your environment. For the source system, configure Extract and Data Pump processes. The initial steps in the example below effectively remove an existing instance of both processes. If none already exist, start with the `START MGR` command.

```
--Stop Manager on primary database
dblogin USERID <GG User's DB ID, requirement depends on database>,
PASSWORD <GG User's DB password, requirement depends on database >

STOP MGR

--Stop GG processes
STOP <name of Extract process>
DELETE EXTTRAIL <relative or fully qualified path where Extract Trail files are
created on source system>/*
DELETE <name of Extract process>

STOP <name of Data Pump process>
DELETE RMTTRAIL <relative or fully qualified path where Replicat Trail files are
created on target system>/*
DELETE <name of Data Pump process>

--Delete Previous Trail Files
SHELL rm <relative or fully qualified path where Extract Trail files are created
on source system>/*

--Start Manager on primary database
START MGR

--Primary database capture configuration
ADD EXTRACT <name of Extract process>, TRANLOG, BEGIN NOW
ADD EXTTRAIL <relative or fully qualified path where Extract Trail files are to be
created on source system>, EXTRACT <name of Extract process>, MEGABYTES 50

--Primary database pump configuration:
```

```
ADD EXTRACT<name of Data Pump process>, EXTTRAILSOURCE <relative or fully
qualified path where Extract Trail files are to be created on source system>,
ADD RMTTRAIL <relative or fully qualified path where Replicat Trail files are to
be created on target system>, EXTRACT<name of Data Pump process>, MEGABYTES 50
```

**Example:**

```
--Stop Manager on primary database
dblogin userid gg, password gg
STOP MGR

--Stop GG processes
STOP EXT_310
DELETE EXTTRAIL ./dirdat/*
DELETE EXT_310

STOP DP_310
DELETE RMTTRAIL ./dirdat/*
DELETE DP_310

--Delete Previous Trail Files
SHELL rm ./dirdat/*

--Start Manager on primary database
START MGR

--Primary database capture configuration
ADD EXTRACT EXT_310, TRANLOG, BEGIN NOW
ADD EXTTRAIL ./dirdat/tr, EXTRACT EXT_310, MEGABYTES 50

--Primary database pump configuration:
ADD EXTRACT DP_310, EXTTRAILSOURCE ./dirdat/tr
ADD RMTTRAIL ./dirdat/tr, EXTRACT DP_310, MEGABYTES 50
```

Implement similar steps for the Replicate process in the target system. The initial steps effectively remove an existing instance of the Replicate process. If none already exist, start with the START MGR command.

```
--Stop Manager on target database
dblogin USERID <GG User's DB ID, requirement depends on database>,
PASSWORD <GG User's DB password, requirement depends on database >
STOP MGR

--Stop GG processes
STOP <name of Replicat process>
DELETE <name of Replicat process>

--Delete CHECKPOINTTABLE
DELETE CHECKPOINTTABLE <GG User's DB ID>..GGSCHKPT

--Delete Previous Trail Files
SHELL rm <relative or fully qualified path where Replicat Trail files are created
on target system>/*

--Start Manager on target database
START MGR

--Create CHECKPOINTTABLE in target database
dblogin USERID <GG User's DB ID>,
PASSWORD <GG User's DB password>
ADD CHECKPOINTTABLE <GG User's DB ID>..GGSCHKPT
```

```
--Target database delivery configuration
ADD REPLICAT <name of Replicat process>, exttrail <relative or fully qualified
path where Replicat Trail files are to be created on target system>
```

**Example:**

```
--Stop Manager on target database
dblogin userid gg, password gg
STOP MGR

--Stop GG processes
STOP REP_310
DELETE REP_310

--Delete CHECKPOINTTABLE
DELETE CHECKPOINTTABLE

--Delete Previous Trail Files
SHELL rm ./dirdat/*

--Start Manager on target database
START MGR

--Create CHECKPOINTTABLE in target database
dblogin userid gg, password gg
ADD CHECKPOINTTABLE

--Target database delivery configuration
ADD REPLICAT REP_310, exttrail ./dirdat/tr
```

### 7.4.3 Setup Step: Configure BI Applications Configuration Manager and Oracle Data Integrator to Support the Source Dependent Schema

**Enable SDS in Configuration Manager**

Enable the SDS option for each datasource defined in Configuration Manager. You can enable the SDS option for the entire datasource or for individual Fact Groups. The SDS option is enabled by setting the value for the IS\_SDS\_DEPLOYED parameter to 'Yes'.

1. In Configuration Manager, select the Source Instance.
2. Click Manage Data Load Parameters.
3. Locate the IS\_SDS\_DEPLOYED parameter in the list.
4. In the Global Parameter Value, replace <Edit Value> with 'Yes'.
5. A warning is displayed indicating that the parameter is being updated globally for all Fact and Dimension Groups. Click Yes to confirm or, if you prefer, set the global parameter to 'No', and then edit the parameter value at the Fact Group or Dimension Group level.

**Add SDS Physical Schemas in ODI**

For each source instance, you must manually add a corresponding physical schema under the 'BIAPPS\_DW' physical server in ODI.

1. In ODI Studio's Topology Navigator, expand the Oracle technology in the Physical Architecture.
2. Right-click on Oracle\_BI\_Apps\_DW and select New Physical Schema.



3. In the Definition, set Schema (Schema) and Schema (Work Schema) both to the SDS schema owner.
4. Select Flexfields.
5. For the DATASOURCE\_NUM\_ID flex field, uncheck the Default checkbox and assign the DSN value associated with that source as defined in Configuration Manager.
6. Save the physical schema definition. Ignore the message about the physical server not being assigned a context.

#### 7.4.4 Setup Step: Generate, Deploy, and Populate the Source Dependent Schema Tables on Target Database

In this step, you generate and run the Data Definition Language to create the SDS tables on the SDS schema in the target database.

##### Generate the SDS DDL

Procedures are provided to generate the required objects to enable the SDS. To generate the required DDL, in ODI Designer execute the 'Generate DDL - SDS' scenario found under 'BI Apps Project > Components > SDS > Oracle > Generate SDS DDL. Provide an appropriate context when prompted.

As the procedure can only accept a single source type, this process needs to be repeated for each different type of Source system to be enabled.

To execute the scenario, right-click it and select Execute. When the scenario is executed, a prompt appears to provide values for the DDL execution options. Refer to the following table describing the options to provide appropriate values.

Option	Description
GG_USER_DW	Golden Gate database user on the BI Applications database
GG_USER_SOURCE	Golden Gate database user on the OLTP database.
SDS_MODEL	The OLTP model to be used to generate the SDS schema. Each model is associated with a logical schema. The logical schema is associated with a physical schema. The logical and physical schema DSN flexfields must match an SDS physical schema's DSN flexfield defined under the BI Apps DW physical server in order for this utility to determine the appropriate schema name to be used for the SDS. The SDS physical schema with the matching DSN flexfield value is used to identify changes and execute the DDL against if the RUN_DDL option is set to Y.
CREATE_SCRIPT_FILE	Y or N. Set to Y if you want to review the DDL or manually execute the script.
REFRESH_MODE	FULL or INCREMENTAL. Full drops and creates all tables. Incremental compares the repository with the SDS schema and applies changes using ALTER statements without dropping tables. Incremental process can take much longer than Full mode and should be used with filters to limit the number of tables compared.
CHAR_CLAUSE	Y or N. For Unicode support, will include the CHAR clause when defining string columns. For example FULL_NAME VARCHAR2(10) would be created as FULL_NAME VARCHAR2(10 CHAR). This ensures that the right length is chosen when the underlying database is a unicode database.

Option	Description
RUN_DDL	<p>Y or N. Determines whether to execute the DDL commands. The script will be executed against the physical SDS schema associated with the SDS_MODEL. Note that this script will be executed via the user defined in the BIAPPS_DW physical server which is usually the owner of the BIAPPS_DW schema and which may not have appropriate privileges to create or alter tables in another schema. To have the utility execute the DDL script, you may need to grant CREATE ANY TABLE, CREATE ANY INDEX, ALTER ANY TABLE and ALTER ANY INDEX to the BIAPPS_DW database user.</p> <p>It is recommended to set this option to N. If set to Y, both the tables and indexes will be created. Having the indexes on the tables will impact the performance of initially loading the tables. Rather, it is recommended that you set this option to N, manually execute the Table DDL, perform the initial load of the tables, then manually execute the Index DDL.</p> <p>Also, if an index or primary key constraint is not defined correctly in ODI, uniqueness or not null errors could be generated and a table could fail to be loaded. Indexes and primary keys are useful for Oracle GoldenGate but are not required. It is better to build the indexes and primary keys after the data is loaded and make any necessary corrections to the constraint's definition in ODI and attempt to build the index or primary key again.</p>
SCRIPT_LOCATION	The location where the script should be created if the CREATE_SCRIPT_FILE option is true.
TABLE_MASK	To generate the DDL for all tables, use a wildcard (the default). To generate for only a subset of tables with names matching a particular pattern, use that pattern with a wildcard, such as PER_%.

If you set CREATE\_SCRIPT\_FILE to Y, four files are generated by the Generate SDS DDL procedure in the location specified by SCRIPT\_LOCATION. One is a .SQL script to creates the tables. Another is a .SQL script to create the indexes and analyze the tables. This allows you to create the tables, perform an initial load of the tables without any indexes that could hurt performance, and then create the indexes and analyze the tables after they are loaded. Another .SQL script is generated which grants SELECT privileges to the OGG database user only for those tables that need to be selected from. The final file is a log file.

### Grant privileges to OLTP Tables

The OGG user must be able to select from the tables in the OLTP database. Rather than grant the SELECT ANY TABLE privilege to the OGG user, SELECT privileges are granted only to those tables that actually need to be replicated to the target system.

The SDS DDL generator procedure creates a script to grant SELECT privileges to the OGG user. Refer to the script 'BIA\_SDS\_Schema\_Source\_Grants\_DDL\_<unique ID>.sql' and execute the contents in the OLTP database with a user with sufficient privileges to grant SELECT privileges on the OLTP tables.

### Create the SDS Tables

The SDS DDL generator procedure creates a .SQL script that follows the naming convention BIA\_SDS\_Schema\_DDL\_<unique ID>.sql which contains the CREATE or ALTER DDL statements to create or alter the tables in the SDS schema. Execute the SQL in this file against the SDS schema.

The ETL process must be able to select from the SDS tables. Typically, the ETL process uses the BI Applications data warehouse schema owner. This must be granted SELECT privileges on the SDS tables. In addition, the Oracle GoldenGate user needs read and write access to these same tables. The SDS Generate DDL procedure grants SELECT

privileges to the BI Applications data warehouse schema owner and `SELECT`, `INSERT`, `UPDATE` and `DELETE` privileges to the OGG user.

### Performing Initial Load of the SDS Tables: Create Database Link to OLTP Database

A variety of methods can be used to initially load the data from the source database to the target database. A procedure is provided to generate a script to perform an initial load as described in the steps below. Note however, that you may opt for other methods. The procedure generates a script that executes DML statements that extract data over a database link.

---

**Note:** LOB and LONG datatype columns are created in the SDS, but the provided utilities to initially copy data from the source to target system cannot support these datatypes, so columns with these datatypes are specifically excluded by these utilities. If data in these columns are required, an alternate method for performing an initial load of the SDS will need to be implemented.

---



---

**Note:** In Siebel implementations, a small number of tables in the Siebel database are created when installing the BI Applications. These tables must be manually created and always have `S_ETL` as a prefix. Be sure these tables have already been created prior to executing these steps. If these tables do not already exist, a "table or view does not exist" error can occur when executing the following commands.

---

First, create a database link to the OLTP database on the target database. The procedure to generate the DML script requires that a database link already exist named "DW\_TO\_OLTP" prior to being executed. The procedure executes using the BIAPPS\_DW physical server so the database link has to either be defined in the same schema as used in the BIAPPS\_DW physical server or else defined as a public database link. This database link must be manually created, it is not automatically created.

The procedure only populates a single SDS schema at a time. If creating multiple SDS schemas to accommodate multiple sources, this database link will need to be updated prior to each run to point to a different OLTP instance.

---

**Note:** The JDE application spreads data across four databases and is tracked under four different submodels under a single JDE specific model. The DML option will need to be executed for each separate submodel and the "DW\_TO\_OLTP" database link will need to be updated prior to executing the DML script.

---

### Performing Initial Load of the SDS Tables: Execute Procedure to generate DML script

This DML script generation procedure requires that the ODI topology is set up correctly, ensuring the OLTP model logical schema DSN matches with the desired target warehouse SDS physical schema DSN. The DSNs are set in the logical or physical schema flexfields.

In ODI Designer, execute the `COPY_OLTP_TO_SDS` scenario found under BI Apps Project > Components > SDS > Oracle > Copy OLTP to SDS.

To execute the scenario, right-click it and select Execute. Provide an appropriate context when prompted. When the scenario is executed, a prompt appears to provide values for the DML execution options. Refer to the following table describing the options to provide appropriate values.

Option	Description
TABLE_LIST	A comma-separated list of tables. A wildcard match % may be used to match multiple tables. Do not include any line breaks. For example: PER_ALL_ASSIGNMENTS_ E,PER%ORG%INFO%,HR%UNIT,FND_LOOKUP_TYPES
CREATE_SCRIPT_FILE	Y or N. Set to Y if you want to review the DDL or manually execute the script.
RUN_DDL	Y or N. Whether to execute the DML commands. The script will be executed against the physical SDS schema associated with the SDS_MODEL. Note that this script will be executed via the user defined in the BIAPPS_DW physical server which is usually the owner of the BIAPPS_DW schema and which may not have appropriate privileges to insert data into tables in another schema. To have the utility execute the DDL script, you may need to grant SELECT ANY TABLE and INSERT ANY TABLE to the BIAPPS_DW database user.  Alternatively, rather than have the procedure execute the script, create the script file and connect to the database as the SDS schema owner and execute the contents of the script file directly.
SDS_MODEL	The OLTP model to be used to generate the SDS schema.
SCRIPT_LOCATION	The location where the script should be created if the CREATE_SCRIPT_FILE option is true.

**Performing Initial Load of the SDS Tables: Execute DML Script on SDS Database**

The resulting DML script can be executed using the SDS schema owner or the BIAPPS DW schema owner. If executed by the BIAPPS DW schema owner, this user must be granted the SELECT ANY TABLE and INSERT ANY TABLE privileges in order to populate data in another schema. If executed using the SDS schema owner, a private database link named "DW\_TO\_OLTP" must be created in the SDS schema (the SDS user must be granted the CREATE DATABASE LINK privilege to create this database link) or already created as a public database link.

The DML script that is generated includes all tables used by all ETL tasks. If you are not executing all ETL tasks, you may want to consider identifying the tasks you are not executing and removing the corresponding tables from this script so that they are not replicated, thus keeping the overall size of the SDS down. Refer to the parameter files to determine the tasks that use each table and edit this script to remove the tables you do not need to replicate.

**Create SDS Indexes and Analyze the SDS schema**

When the tables are populated, execute the 'BIA\_SDS\_Schema\_Index\_DDL\_<unique ID>.sql' script to create indexes and analyze the SDS tables.

**7.4.5 Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines**

Parameter files are used to control how Oracle GoldenGate operates. These files are deployed to the source system, where the Extract and Data Pump processes are executed, and the target system, where the Replicat process is executed. An ODI procedure generates these parameter files based on the metadata defined in ODI. A

scenario that executes this procedure is provided to generate the Oracle GoldenGate parameter files to populate the SDS.

### 7.4.5.1 Generate Oracle GoldenGate Parameter Files

To generate the required parameter files, execute the 'GENERATE\_SDS\_OGG\_PARAM\_FILES' scenario found under BI Apps Project > Components > SDS > Generate SDS OGG Param Files. When the scenario is executed, a prompt appears to provide values for the parameter file options. Refer to the following table describing the options to provide appropriate values to match your environment. As the procedure can only accept a single Source type, this process needs to be repeated for each different type of Source system to be enabled.

Parameter	Description
PARAM_FILE_LOCATION	Location on machine where ODI client is running where parameter files will be created. Example: C:\temp\
DATASOURCE_NUM_ID	Datasource Num Id value associated with the particular source for which parameter files are to be generated. Example: 310
DATAPUMP_NAME	Name of the Datapump Process specified when installing OGG on the source machine. Limit is eight characters. Suggested naming convention is DP_<Datasource Num Id>, for example DP_310.
EXTRACT_NAME	Name of the Primary Extract Process specified when installing OGG on the source machine. Limit is eight characters. Suggested naming convention is EX_<Datasource Num Id>, for example EX_310.
EXTRACT_TRAIL	Path and name of trail file on source system. Can be a relative or fully qualified path, though actual file name must be two characters. In the example below, 'tr' is the name of the trail file. Example: ./dirdat/tr
DEFSFILE	The relative or fully qualified path on the source system where the DEFGEN definition file should be created and file name. This value is included in the DEFGEN.prm parameter file that is generated by this procedure. The DEFGEN utility is executed on the source database, so the path provided must be a path available on the system the source database runs on. Suggested naming convention is DEF_<Datasource Num Id>.def. Example: ./dirdef/DEF_310.def
SOURCE_GG_USER_ID	Database user dedicated to the Oracle GoldenGate processes on the source database. Example: GG_USER
SOURCE_GG_PASSWORD	Password for the database user dedicated to the Oracle GoldenGate processes on the source database.  By default, the password is stored as clear text in the generated parameter file. If an encrypted value is desired, use the ENCRYPT_PASSWORD utility and edit the generated parameter files accordingly. Example: GG_PASSWORD
SOURCE_PORT	Port used by the OGG Manager Process on the source system. The default value when OGG is installed is 7809.
REPLICAT_NAME	Name of the Replicat Process specified when installing OGG on the target machine. Limit is eight characters. Suggested naming convention is REP_<Datasource Num Id>, for example REP_310

Parameter	Description
SOURCE_DEF	This is the Source Definitions file created by executing the DEFGEN utility on the source database and copied over to the target machine. This can be either a relative or fully qualified path to this definition file on the target system. Include the /dirdef subfolder as part of the path. Suggested naming convention is DEF_<Datasource Num Id>.def, for example ./dirdef/DEF_310.def  Note that the file name is usually the same as the one defined for DEFSFILE but the paths are usually different as DEFSFILE includes the path where OGG is stored on the source system, while SOURCE_DEFS includes the path where OGG is installed on the target system.
REMOTE_HOST	IP address or Host Name of the target machine where the Replicat process runs.
REMOTE_TRAIL	Path and name of the trail file on target system. Can be a relative or fully qualified path though the actual file name must be two characters. In the example below, 'tr' is the name of the trail file.  Example: ./dirdat/tr
BIA_GG_USER_ID	Database user dedicated to the Oracle GoldenGate processes on the target database, for example GG_USER
BIA_GG_PASSWORD	Password for the database user dedicated to the Oracle GoldenGate processes on the target database.  By default, the password is stored as clear text in the generated parameter file. If an encrypted value is desired, use the ENCRYPT PASSWORD utility and edit the generated parameter files accordingly. Example: GG_PASSWORD
BIA_GG_PORT	Port used by the OGG Manager Process on the target system. The default value when OGG is installed is 7809.

The procedure automatically creates subfolders under a folder you specify. The naming convention is DSN\_<DATASOURCE\_NUM\_ID> where <DATASOURCE\_NUM\_ID> is the value you specify when executing this procedure. For example, if you specify 310 as the value for DATASOURCE\_NUM\_ID, there will be a folder named DSN\_310. Under this folder are two more subfolders, 'source' and 'target'. The 'source' folder contains all of the files that need to be copied to the source system, while 'target' contains all of the files that need to be copied to the target system.

**Tip:** The parameter files that are generated include all tables used by all ETL tasks. The task that uses the table is identified in the parameter file. If you are not executing all ETL tasks, you may want to consider identifying the tasks you are not executing and removing the corresponding tables from the parameter files so that they are not replicated. This keeps the overall size of the SDS down.

### About JD Edwards Support

The JDE application spreads data across up to four databases. Each database instance must be assigned its own extract/datapump processes and a separate corresponding replicat process. If the JDE components are on a single database, generate a single set of parameter files. If the JDE components are spread across two, three or four databases, generate a corresponding number of parameter files.

For more details on this configuration, refer to the "Configuring Oracle GoldenGate for real-time data warehousing" chapter in the *Oracle GoldenGate Windows and UNIX Administrator's Guide*, which discusses multiple source databases replicating data to a single target database.

Keep the following in mind when generating the parameter files. Execute the procedure for each database instance. The name of each process and trail file should be

unique. The following example assumes all four components are on different databases:

Component	Extract Name	Data Pump Name	Extract Trail	Defs File	Replicat Name	Replicat Trail	Source Defs
Control	EX_410A	DP_410A	./dirdat/ta	./dirdef/DEF_310A.def	REP_410A	./dirdat/ta	./dirdef/DEF_310A.def
Data	EX_410B	DP_410B	./dirdat/tb	./dirdef/DEF_310B.def	REP_410B	./dirdat/tb	./dirdef/DEF_310B.def
Data Dictionary	EX_410C	DP_410C	./dirdat/tc	./dirdef/DEF_310C.def	REP_410C	./dirdat/tc	./dirdef/DEF_310C.def
System	EX_410D	DP_410D	./dirdat/td	./dirdef/DEF_310D.def	REP_410D	./dirdat/td	./dirdef/DEF_310D.def

### About PeopleSoft Learning Management Support

PeopleSoft has a Learning Management pillar which is tightly integrated with the Human Capital Management pillar. HCM can be deployed without LM but LM cannot be deployed without HCM. When both are deployed, BI Applications treats the HCM with LM pillars in a similar fashion as it treats JDE: the data is spread across two databases but is treated as a single application. As with the JDE application, in this configuration each database instance must be assigned its own extract/datapump processes and a separate corresponding replicat process.

For more details on this configuration, refer to the "Configuring Oracle GoldenGate for real-time data warehousing" chapter in the *Oracle GoldenGate Windows and UNIX Administrator's Guide*, which discusses multiple source databases replicating data to a single target database.

Keep the following in mind when generating the parameter files. Execute the procedure for each database instance. The name of each process and trail file should be unique.

Component	Extract Name	Data Pump Name	Extract Trail	Defs File	Replicat Name	Replicat Trail	Source Defs
HCM Pillar	EX_518A	DP_518A	./dirdat/ta	./dirdef/DEF_518A.def	REP_518A	./dirdat/ta	./dirdef/DEF_518A.def
LM Pillar	EX_518B	DP_518B	./dirdat/tb	./dirdef/DEF_518B.def	REP_518B	./dirdat/tb	./dirdef/DEF_518B.def

### 7.4.5.2 Configure the Source System

Copy all of the files from the 'source' directory on the ODI client to the corresponding directories in the source system:

Copy the following file to the <ORACLE OGG HOME> directory:

- ADD\_TRANDATA.txt

Copy the following files to the <ORACLE OGG HOME>/dirprm directory:

- DEFGEN.prm
- <EXTRACT\_NAME>.prm where <EXTRACT\_NAME> is the value specified when generating the parameter files.
- <DATAPUMP\_NAME>.prm where <DATAPUMP\_NAME> is the value specified when generating the parameter files.

**Edit the Extract parameter file**

By default, the procedure creates a basic set of parameter files that do not include support for a variety of features. For example, the parameter files do not include support for Transparent Data Encryption (TDE) or unused columns. The procedure also does not include the options to encrypt data.

If your source tables have unused columns, edit the Extract parameter file to include `DBOPTIONS ALLOWUNUSEDCOLUMN`. If encrypting the data is desired, edit the parameter files to add the `ENCRYPTTRAIL` and `DECRYPTTRAIL` options.

To support such features, edit the generated parameter files using the `GGSCI EDIT PARAMS <parameter file>` command. Also edit the generated param files to implement various tuning options that are specific to the environment. Refer to the *Oracle GoldenGate Reference* guide for details on implementing these options.

Start the GGSCI command utility from the `<ORACLE OGG HOME>` directory. Execute the following command to edit the Extract parameter file - this should open the Extract parameter file you copied to `<ORACLE OGG HOME>/dirprm`:

```
GGSCI>EDIT PARAMS <EXTRACT_NAME>
```

Save and close the file.

**Enable Table Level Logging**

Oracle GoldenGate requires table-level supplemental logging. This level of logging is only enabled for those tables actually being replicated to the target system. The SDS Parameter file generator creates 'ADD\_TRANDATA.txt' file to enable the table-level logging. This script is executed using the GGSCI command with the Oracle GoldenGate database user. This user must be granted the `ALTER ANY TABLE` privilege prior to executing this script. Once the script completes, this privilege can be removed. Alternatively, edit the script file to use a database user with this privilege. When the OGG database user is originally created, the `ALTER ANY TABLE` privilege is granted at that time. Once the script to enable table level supplemental logging completes, this privilege can be revoked from the OGG user.

Start the GGSCI command utility from the `<ORACLE OGG HOME>` directory and execute the following command:

```
GGSCI> obey ADD_TRANDATA.txt
```

Exit GGSCI, then connect to the database and revoke the `ALTER ANY TABLE` privilege.

---

---

**Note:** If a table does not have a primary key or any unique indexes defined, you may see a warning message like the following. This is a warning that a 'pseudo' unique key is being constructed and used by Oracle Golden Gate to identify a record. Performance is better if a primary key or unique index is available to identify a record but as we generally cannot add such constraints to an OLTP table when they do not already exist, Oracle Golden Gate creates this pseudo unique key.

```
WARNING OGG-00869 No unique key is defined for table 'FA_
ASSET_HISTORY'. All viable columns will be used to represent the
key, but may not guarantee uniqueness. KEYCOLS may be used to
define the key.
```

---

---



### Generate Data Definition File on the Source System

As the source and target tables do not match exactly, configure the Replicat process to use a data definition file which contains definitions of the tables on the source system required to map and convert data. The procedure generates a basic DEFGEN.prm file used to create a data definition file. If required, edit this file to reflect your environment. Refer to the Oracle GoldenGate documentation for more details. For example, the DEFGEN.prm file does not leverage the encryption option, so if this or other options are desired, edit the parameter file to enable them.

To edit the DEFGEN.prm file, start the GGSCI command utility from the Oracle GoldenGate home directory. Execute the following command to open and edit the DEFGEN.prm file you copied to <ORACLE OGG HOME>/dirprm:

```
GGSCI>EDIT PARAMS DEFGEN
```

Save and close the file and exit GGSCI, then run the DEFGEN utility. Refer to Oracle GoldenGate documentation for more information about this utility and its execution. The following is an example of executing this command on UNIX:

```
defgen paramfile dirprm/defgen.prm
```

A data definition file is created in the <ORACLE OGG HOME>/ folder with the path and name specified using the DEFSFILE parameter. FTP the data definition file to the <ORACLE OGG HOME>/dirdef folder on the remote system using ASCII mode. Use BINARY mode to FTP the data definitions file to the remote system if the local and remote operating systems are different and the definitions file is created for the remote operating system character set.

#### 7.4.5.3 Configure the Target System

Copy all of the files from the 'target' directory on the ODI client to the corresponding directories in the target system.

Copy the following file to the <ORACLE OGG HOME>/dirprm directory in the target system:

- <REPLICAT\_NAME>.prm where <REPLICAT\_NAME> is the value specified when generating the parameter files.

#### Edit the Replicat Parameter File

By default, the procedure creates a basic set of parameter files that do not include support for a variety of features. For example, the parameter files do not include support for Transparent Data Encryption (TDE) or unused columns. The procedure also does not include the options to encrypt data. If encrypting the data is desired, edit the generated parameter files to add the ENCRYPTTRAIL and DECRYPTTRAIL options. To support such features, edit the generated parameter files using the GGSCI EDIT PARAMS <parameter file> command. Also edit the generated param files to implement various tuning options that are specific to the environment. Refer to the *Oracle GoldenGate Reference* guide for details on implementing these options.

Start the GGSCI command utility from the <ORACLE OGG HOME> directory. Execute the following command to edit the Extract parameter file. This should open the Replicat parameter file - this should open the Replicat parameter file you copied to <ORACLE OGG HOME>/dirprm:

```
GGSCI>EDIT PARAMS <REPLICAT_NAME>
```

Save and close the file, and exit GGSCI.

### Create a Checkpoint Table (Optional)

The procedure does not account for a checkpoint table in the target system. A checkpoint table is not required but is recommended. If a checkpoint table is desired, follow the steps detailed in Oracle GoldenGate documentation to create a checkpoint table and edit the GLOBALS param file to reference this table.

Start the GGSCI command utility

```
GGSCI> EDIT PARAMS ./GLOBALS  
CHECKPOINTTABLE <OGG User>.<Table Name>
```

Save and close the file, then run the following commands:

```
GGSCI> DBLOGIN USERID <OGG User> PASSWORD <OGG Password>  
GGSCI> ADD CHECKPOINTTABLE <OGG User>.<Table Name>
```

## 7.4.6 Setup Step: Start Oracle GoldenGate on Source and Target Machines

### Start Oracle GoldenGate on the Source Machine

Use the following command to start the Extract and Data Pump processes on the source system.

```
START MGR  
--Start capture on primary database  
START <name of Extract process>  
  
--Start pump on primary database  
START <name of Data Pump process>
```

Example:

```
START MGR  
--Start capture on primary database  
START EXT_310  
  
--Start pump on primary database  
START DP_310
```

### Start Oracle GoldenGate on the Target Machine

Use the following command to start the Replicat process in the target system.

```
START MGR  
--Start delivery on target database  
START <name of Replicat process>
```

Example:

```
START MGR  
  
--Start capture on primary database  
START REP_310
```

## 7.5 Customization

This section describes SDS considerations for ETL customization. Customization falls into the following types:

- [Section 7.5.1, "Adding a Non-Custom Source Column to an Existing ETL Task"](#)

- [Section 7.5.2, "Adding a Custom Source Column to an Existing ETL Task"](#)
- [Section 7.5.3, "Adding a Non-Custom Source Table to an Existing ETL Task"](#)
- [Section 7.5.4, "Creating a Custom ETL Task"](#)

### 7.5.1 Adding a Non-Custom Source Column to an Existing ETL Task

All columns in each source table are replicated. If you extend an existing ETL task with a column that is a standard part of the source system, no special steps are required.

### 7.5.2 Adding a Custom Source Column to an Existing ETL Task

If you add a custom source column to the ETL task, the OGG process already extracts from this table and needs to be modified to include this column in the extract. In addition, the SDS table must be altered to include this column.

Run the RKM to add the column to the ODI model, then use the SDS DDL generator in incremental mode to add this column to the SDS table. If the SDS has already been populated with data, repopulate it by running the SDS 'Copy to SDS' procedure, providing the customized table in the Table List parameter.

### 7.5.3 Adding a Non-Custom Source Table to an Existing ETL Task

In cases where an ETL task is customized to use an additional table that is included as part of the standard OLTP application, if the table is already used by another ETL task then that table should already exist in the ODI model and is already replicated in the SDS. No special steps are required.

If the table is not already used by any other ETL task, run the RKM to add the table to the ODI model and use the SDS DDL generator in incremental mode to add this table to the SDS schema. Use one of the initial load options with this table in the Table List to repopulate the table. Regenerate the SDS parameter files to ensure the table is included as part of the replication process.

### 7.5.4 Creating a Custom ETL Task

If a custom ETL task sources a table that is already extracted from, no special steps are required. However, if the custom task extracts from a new table that is not already included as part of the standard Oracle BI Applications source-specific model, run the RKM to add the table to the ODI model and use the SDS DDL generator in incremental mode to add this table to the SDS schema. Use one of the initial load options with this table in the Table List to repopulate the table. Regenerate the SDS parameter files to ensure the table is included as part of the replication process.

## 7.6 Patching

After releasing BI Applications, Oracle may release patches. This section discusses patches that impact SDS related content and considerations when deploying those patches.

### Patch Applied to ODI Datastores or Interfaces

ODI datastores and interfaces are the only BI Applications content that impacts SDS related content. Applied patches that impact OLTP-specific datastores are relevant to the SDS.

It is possible that an applied patch could change the definition of an OLTP-specific datastore, for example adding a column or changing a column's size or datatype. A patch could also introduce a new table. In these cases, run the SDS DDL generator in incremental mode, providing the list of datastores that are patched. Execute the generated DDL against the SDS schema. In case of a new column or table being introduced, run the initial load, specifying just the new or changed table in the table list in the provided procedure.

A patch could impact an interface by adding a new OLTP table from which data must be extracted. In the previous step, you would have generated the DDL and DML to create and populate this table. Run the Oracle GoldenGate parameter generator procedure to recreate the required parameter files and redeploy to the source and target systems. For more information about creating and recreating parameter files, see [Section 7.4.5, "Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines."](#)

### **Patch Applied to SDS-Related Procedure**

In the case an SDS-related procedure is replaced by a patch, depending on the nature of the reason for the patch, it may be necessary to re-execute the procedure and re-deploy its output. If the patch is related to the SDS DDL or SDS Copy procedures, the procedure can be run in incremental mode to make some changes to the SDS or in full mode to completely replace the SDS. The patch notes will describe exactly what must be done to implement the patched procedure.

## **7.7 Troubleshooting**

This section provides troubleshooting tips and resolutions for common errors encountered during setup of Oracle GoldenGate and SDS. It includes the following sections, which correlate to setup steps:

- [Section 7.7.1, "Create the SDS Tables"](#)
- [Section 7.7.2, "Using the DML Option to Perform an Initial Load"](#)
- [Section 7.7.3, "Create SDS Indexes and Analyze the SDS schema"](#)

### **7.7.1 Create the SDS Tables**

If you encounter any issues with the script generated by the 'GENERATE\_SDS\_DDL' procedure, verify the following have been correctly set:

- The model you are specifying is associated with a logical schema.
- The logical schema's DATASOURCE\_NUM\_ID flexfield is assigned a numeric value. A value is automatically assigned when a datasource is registered in Configuration Manager.
- The logical schema is mapped to a physical schema in the context (for example, Global) you are executing the procedure with. The physical schema is automatically mapped to the Global context when the datasource is registered in Configuration Manager.
- The physical schema's DATASOURCE\_NUM\_ID flexfield is assigned the same numeric value as the logical schema. A value is automatically assigned when a datasource is registered in Configuration Manager.
- Under the same context, a physical schema is mapped to the DW\_BIAPPS11G logical schema, for example BIAPPS\_DW.OLAP.

- A new SDS physical schema has been added to the same physical server, for example BIAPPS\_DW.SDS\_EBS\_12\_1\_3\_310. This physical schema is manually added.
- The physical schema's DATASOURCE\_NUM\_ID flexfield is assigned the same numeric value as used previously. This value is manually assigned.

The following are some common error messages and issues you may encounter.

- **"com.sunopsis.tools.core.exception.SnpsSimpleMessageException: com.sunopsis.tools.core.exception.SnpsSimpleMessageException: Exception getSchemaName("[logical schema name]", "D") : SnpPschemaCont.getObjectByIdent : SnpPschemaCont does not exist"**

Verify the logical schema is mapped in the context you are executing the procedure with.

- **"java.lang.Exception: The application script threw an exception: java.lang.Exception: Model with code '[logical schema]' does not exist"**

Verify the logical schema associated with your model has been assigned a value for the DATASOURCE\_NUM\_ID flexfield.

- **"java.lang.Exception: The application script threw an exception: java.lang.Exception: Can't find physical schema for connection for DW\_BIAPPS11G with DSN 310 in context Global"**

Verify a physical schema is created under the Data Warehouse physical server and assigned the same DATASOURCE\_NUM\_ID value as assigned to the OLTP.

## 7.7.2 Using the DML Option to Perform an Initial Load

If you encounter any issues with the script generated by the 'COPY\_OLTP\_TO\_SDS' procedure, verify the following have been correctly set:

- A database link with the name DW\_TO\_OLTP is created in the database user schema used by the data warehouse Data Server (BIAPPS\_DW) that points to the OLTP database. The procedure is executed by this user so Oracle looks for this database link in the user's schema, not the SDS schema. You still need a database link with this name in the SDS schema for other reasons, so you have a total of two database links to the same source database.

The following are some common error messages and issues you may encounter.

- **"ODI-1228: Task Copy SDS Data (Procedure) fails on the target ORACLE connection BI\_APPLICATIONS\_DEFAULT"**

PL/SQL: ORA-00942: table or view does not exist

PLS-00364: loop index variable 'COL\_REC' use is invalid"

Verify a database link named DW\_TO\_OLTP exists in the schema owned by the database user associated with the DW physical server.

- **Insert statement only populates the CDC\$ columns**

The script has statements such as the following where only the CDC\$ columns are populated:

```
truncate table SDS_EBS11510_FULLL.HR_LOCATIONS_ALL;
INSERT /*+ APPEND */ INTO SDS_EBS11510_FULLL.HR_LOCATIONS_ALL (CDC$_SRC_LAST_UPDATE_DATE, CDC$_RPL_LAST_UPDATE_DATE, CDC$_DML_CODE) SELECT SYSDATE, SYSDATE, 'I'
FROM HR_LOCATIONS_ALL@DW_TO_OLTP;
```

Verify the database link DW\_TO\_OLTP is pointing to the correct OLTP database. The procedure gets a column list from the data dictionary on the OLTP database for the tables that correspond to the SDS tables. If the database link points to the wrong database, a column list will not be retrieved.

### 7.7.3 Create SDS Indexes and Analyze the SDS schema

When executing the script to create indexes and primary key constraints on the SDS tables, you may see some of the following error or warning messages:

- **"such column list is already indexed"**

You may see this message when executing the script that creates the indexes. This message can be ignored.

Oracle GoldenGate works best when a primary key is defined on the target table in order to determine which record to update. If a primary key is not found, the replicat process searches for a unique index to determine which record to update. The definition of the tables in the SDS is based on the definition in the source system (leveraging both the application dictionary and data dictionary). If the table does not have a primary key in the source system but does have multiple unique indexes, a primary key may be added in the ODI definition to ensure Oracle GoldenGate can correctly identify the record to be updated. This primary key may be defined on the same column that a unique index is already defined on. The DDL script creates the primary key as an index and a constraint before creating the unique index. When creating the unique index, the database will report that the column is already indexed.

- **"column contains NULL values; cannot alter to NOT NULL"**

This error can occur when a primary key constraint is being created. Primary key constraints are introduced in ODI when a primary key is defined in the OLTP system. A primary key constraint may also be introduced in ODI when there is no primary key in the OLTP system for the table but the table has multiple unique indexes; in this case, a primary key constraint is introduced to ensure Oracle GoldenGate does not use a unique index that may not correctly identify a record for updates. This error can occur for two reasons:

- **OLTP table has Primary Key Constraint**

Due to differences in patches and subversions, it is possible the OLTP instance used to originally import the datastores from had a primary key constraint that differs from the OLTP version you are using. If the OLTP table has a primary key constraint, ensure the definition in ODI matches this primary key. If there is a difference, you can modify the Index DDL script to use the proper definition to create the primary key constraint in the target database. You should also update the constraint in ODI to match this definition.

If the OLTP and ODI definitions of the primary key constraint match, it is possible the initial load process did not populate one or more of the columns that make up the primary key. If the primary key includes a LOB or LONG datatype, data is not replicated in these columns, which would leave the column empty. In this case, no unique index or primary key can be created, and without data in this column the record cannot be uniquely identified. Any ETL task that extracts from this table needs to be modified to extract directly from the OLTP system. This is done by modifying the load plan step for this task, overwriting the IS\_SDS\_DEPLOYED parameter for that load plan step to a setting of 'N'.

---

If the OLTP and ODI definitions of the primary key constraint match and the key does not include a column that has either the LOB or LONG datatype, review the initial load files and verify whether the column is populated or not. Refer to [Section 7.7.2, "Using the DML Option to Perform an Initial Load"](#) for more details.

- **OLTP table does not have Primary Key Constraint**

Primary key constraints in the ODI model are introduced when a primary key may not exist in the original table. This primary key generally matches an existing unique index. Due to differences in patch and subversions for a given OLTP version, it is possible that the instance used when importing a unique index had a column that is not nullable but in another OLTP version, that column may be nullable. Unique indexes allow null values but primary keys do not. In this case, a unique index is created for the SDS table but the primary key constraint fails to be created. Oracle GoldenGate uses the first unique index it finds (based on the index name) to identify a record for update; if the index that the primary key constraint is based on is not the first index, rename this index in ODI to ensure it will be the first. Generally, the first unique index is the correct index to use to identify a record, in which case this error can be ignored.

- **"cannot CREATE UNIQUE INDEX; duplicate keys found"**

Due to differences in patch and subversions for a given OLTP version, it is possible that the instance used when importing a unique index uses a different combination of columns than are used in your particular version of the same OLTP. For example, the OLTP subversion used to import an index uses 3 columns to define the unique index but a later subversion uses 4 columns, and you are using this later subversion. Check the definition of the unique index in your OLTP instance and modify the index script and corresponding constraint definition in ODI to match.





## A

---

- aggregate tables
  - data warehouse tables, 4-4
- attributes
  - historically significant attributes, identifying, 1-29

## C

---

- charts
  - local seed data, importing, 3-3
  - translation tables, process of maintaining, 3-1
  - W\_LOCALIZED\_STRING\_G table, adding custom translations, 3-4
  - Web catalog strings, about translating, 3-5
- cross-module configuration, performing
  - dimension keys, resolving, 1-32, 1-39
  - domain values, working with, 1-33, 1-40
  - loads, configuring, 1-27
  - records, filtering and deleting, 1-27
  - slowly changing dimensions, changing, 1-29
- CSV file
  - delimiter, 1-26
- currency conversion, 4-10
- customization, 1-3
  - adding an index, 1-36
  - and patch installation, 1-3
  - category 1, 1-3
  - category 2, 1-20
  - category 3, 1-25
  - Domain Values, 1-33
  - overview, 1-1
  - Stored Lookups, 1-32

## D

---

- data warehouse languages
  - configuring base and installed languages, 2-1
- data warehouse tables
  - aggregate tables, 4-4
  - dimension class tables, 4-4
  - dimension tables, 4-4
  - fact tables, 4-5
  - helper tables, 4-5
  - hierarchy tables, 4-5

- internal tables, 4-6
- mini-dimension tables, 4-5
- naming conventions, 4-1
- primary data values, 4-10
- staging tables, 4-6
- standard column prefixes, 4-7
- supported list of values, 4-10
- system columns, 4-8
- table types, 4-2
- translation tables, 4-6
- delimiter
  - for CSV file, 1-26
- dimension class tables
  - data warehouse tables, 4-4
- dimension tables
  - business role-based flags, 4-4
  - data warehouse tables, 4-4
- dimensions
  - key resolution, using a lookup, 1-32, 1-39
  - keys, resolving, 1-32, 1-39
- domain values
  - domain value conversion process, understanding, 1-33
  - domain value set, configuring, 1-36
  - domain value set, extending, 1-34
  - importance, understanding, 1-33
  - working with, 1-33, 1-40

## E

---

- extracts, configuring
  - extracting additional data, about, 1-25
  - extracting data from new source table, 1-26
  - extracting new data from existing source table, 1-26

## F

---

- fact tables
  - data warehouse tables, 4-5

## H

---

- helper tables
  - data warehouse tables, 4-5
- hierarchy tables

- data warehouse tables, 4-5
- historically significant attributes
  - identifying, 1-29

---

## I

- internal tables, 4-6

---

## L

- loads, configuring
  - about, 1-27

---

## M

- metadata
  - externalizing strings, 3-4
- mini-dimension tables
  - data warehouse tables, 4-5
- multi-language support
  - about BI Applications domains, 2-2
  - about dimension translation tables, 2-4
  - about pseudo-translations, 2-2
  - introduction, 2-1
  - translation tables, 2-1

---

## N

- naming conventions
  - data warehouse tables, 4-1

---

## O

- Oracle BI Charts
  - metadata strings, externalizing, 3-4
  - Oracle BI seed data, updating for non-English locales, 3-2
- Oracle BI Presentation layer
  - process of maintaining translation tables, 3-1
- Oracle BI repository
  - externalizing metadata strings, 3-4
- Oracle BI seed data
  - updating for non-English locales, 3-2

---

## P

- patch installation
  - impact on customizations, 1-3
- primary extract and delete mapping
  - source-archived records, deletion configuration, 1-29
  - working with, 1-29
- primary extract and delete mappings
  - understanding, 1-27
  - working with, 1-29

---

## R

- records
  - filtering and deleting, 1-27, 1-29
  - source-archived records, deletion

- configuration, 1-29
- roadmap
  - customization, 1-1

---

## S

- seed data
  - importing, 3-3
- slowly changing dimensions, configuring
  - about, 1-29
  - effective dates, 1-31
  - historically significant attributes, identifying, 1-29
  - Type I changing dimension, 1-30
  - Type II slowly changing dimension, 1-31
- source-archived records
  - deletion configuration, 1-29
- staging tables
  - data warehouse tables, 4-6
- standard column prefixes, 4-7
- standard column suffixes, 4-7
- string translations
  - adding for metadata, 3-5
- system columns, 4-8
  - multi-currency codes for, 4-9

---

## T

- table types
  - data warehouse tables, 4-2
- Transactional database
  - importing locale seed data, 3-3
- translation tables
  - data warehouse tables, 4-6
  - process of maintaining, 3-1

---

## U

- Unicode
  - local seed data, importing, 3-3
  - metadata strings, externalizing, 3-4
  - Oracle BI seed data, updating for non-English locales, 3-2
  - string translations, adding for metadata, 3-5
  - translation tables, process of maintaining, 3-1
  - W\_LOCALIZED\_STRING\_G table, adding custom translations, 3-4
  - Web catalog strings, about translating, 3-5

---

## W

- W\_LOCALIZED\_STRING\_G table, adding custom translations, 3-4
- Web Catalog strings
  - about translating, 3-5
- Work Repository
  - customization and patch application, 1-3