Oracle® Business Intelligence Applications

Administrator's Guide

11*g* Release 1 (11.1.1.8.0)

E49134-01

February 2014

Provides topics of interest to system administrators, including customization, multi-language support, localizing deployments, and analyzing data lineage.

ORACLE®

Oracle Business Intelligence Applications Administrator's Guide, 11*g* Release 1 (11.1.1.8.0)

E49134-01

Primary Author: Dan Hilldale

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

Contents

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.

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 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 for a list of related Oracle Business Intelligence Applications documents:
`http://docs.oracle.com/cd/E38317_01/index.htm`

Conventions

The following text conventions are used in this document:

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

What's New in This Release

Oracle BI Applications 11.1.1.8 is a new release. This chapter describes features in Oracle Business Intelligence Applications 11g Release 1 (11.1.1.8.0) documented in this guide, *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Applications*, that may be of note.

This chapter contains the following topics:

- Notable Features in Oracle BI Applications Documented in This Guide

Notable Features in Oracle BI Applications Documented in This Guide

New features in Oracle BI Applications 11g Release 1 (11.1.1.8.0) that are documented in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Applications* include the following:

New Data Lineage Dashboards

Oracle BI Applications optionally provides data lineage dashboards, providing data analysis from the transactional source application through the Business Intelligence repository metadata and the underlying Extract-Transform-Load mappings. Chapter 5, "Researching Data Lineage," explains how multi-language support is implemented in Oracle BI Applications 11g Release 1 (11.1.1.8.0).

1

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.

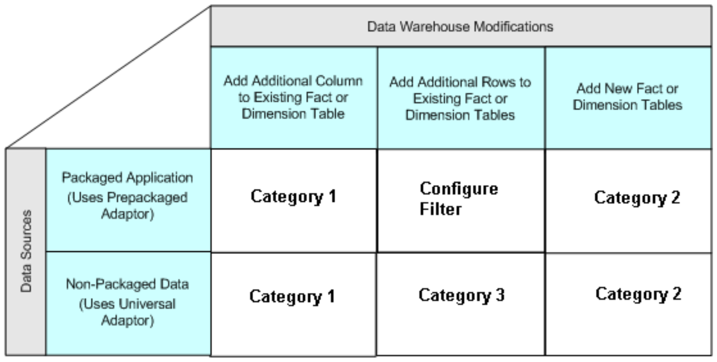
- Non-packaged data sources, which use the Universal adapter.

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.

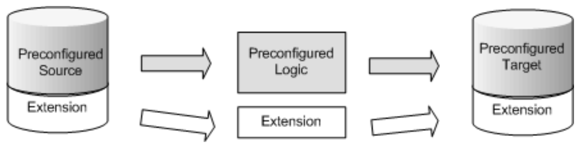
Oracle Business Intelligence Applications recognizes two types of customization: extension and modification. The supported extension logic allows you to add to existing objects. For example, you can extract additional columns from a source, pass them through existing mappings, and populate new columns 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. Generate scenarios for any new Adaptors, using the option to generate the scenario as if all underlying objects are materialized. The scenario will be generated reflecting the custom adaptor name. 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. Update Load Plan steps in the generated Load Plan to reference the custom scenario.
13. 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

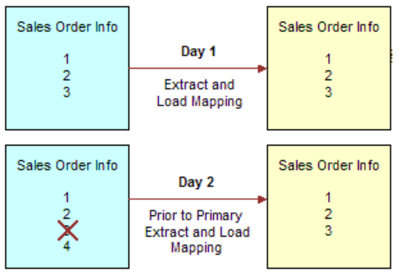
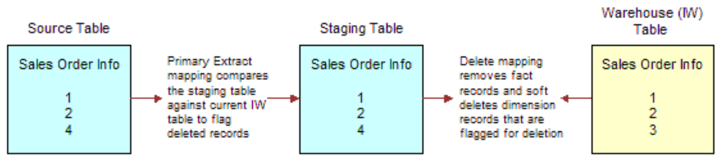


Figure 1–4 shows the primary extract and delete process that occurs when day two's information is extracted and loaded into the Oracle Business Analytics Warehouse from the source. The initial extract brings record four into the Oracle Business Analytics Warehouse. Then, using a primary extract mapping,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


Source Table
Sales Order Info
1
2
4
Primary Extract mapping compares the staging table against current IW table to flag deleted records
Staging Table
Sales Order Info
1
2
4
Delete mapping removes fact records and soft deletes dimension records that are flagged for deletion
Warehouse (IW) Table
Sales Order Info
1
2
3

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.
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 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, and, additionally, merge that RPD with your customized release 11.1.1.8 RPD.

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 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 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 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.

2 About Multi-Language Support

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

This chapter contains the following topics:

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

INTEGRATION_ID	NAME	LANGUAGE_CODE	SRC_LANGUAGE_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

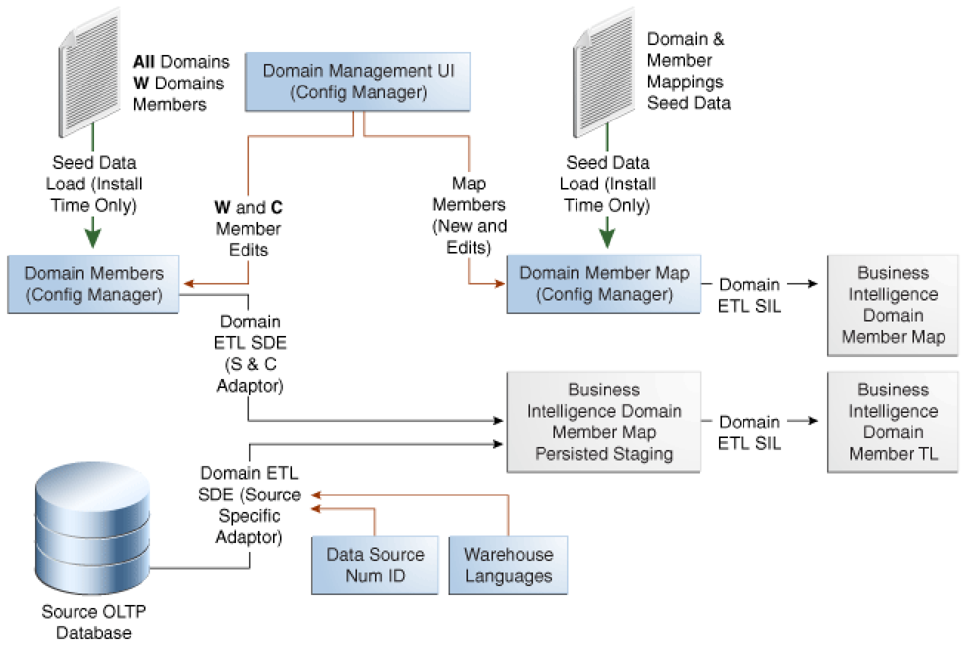
The W_DOMAIN_MEMBER_LKP_TL table stores both domains that are extracted from the source system as well as internally defined domains that are seeded in the Configuration Manager. For each of the %_CODE columns that have translated values available in the source system, an ETL process extracts the domain 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


All Domains
W Domains
Members
Domain Management UI (Config Manager)
Domain & Member Mappings Seed Data
Seed Data Load (Install Time Only)
W and C Member Edits
Map Members (New and Edits)
Seed Data Load (Install Time Only)
Domain Members (Config Manager)
Domain Member Map (Config Manager)
Domain ETL SIL
Business Intelligence Domain Member Map
Domain ETL SDE (S & C Adaptor)
Business Intelligence Domain Member Map Persisted Staging
Domain ETL SIL
Business Intelligence Domain Member TL
Domain ETL SDE (Source Specific Adaptor)
Data Source Num ID
Warehouse Languages
Source OLTP Database

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

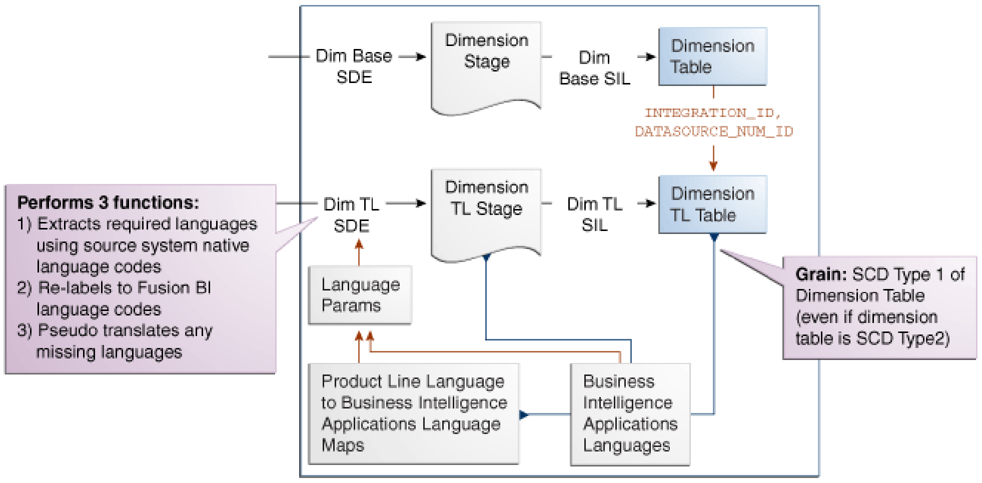
Similar to the Oracle BI Applications domain ETL process, when using dimension translation tables, ETL tasks extract the translated values from the transactional system. Rather than the domain staging table being loaded, the dimension's translation staging table is loaded. The ETL process then moves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


Dim Base SDE
Dimension Stage
Dim Base SIL
Dimension Table
INTEGRATION_ID, DATASOURCE_NUM_ID
Performs 3 functions:
1) Extracts required languages using source system native language codes
2) Re-labels to Fusion BI language codes
3) Pseudo translates any missing languages
Dim TL SDE
Dimension TL Stage
Dim TL SIL
Dimension TL Table
Grain: SCD Type 1 of Dimension Table (even if dimension table is SCD Type2)
Language Params
Product Line Language to Business Intelligence Applications Language Maps
Business Intelligence Applications Languages

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*.

3

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 Oracle BI 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 Oracle BI 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 Oracle BI 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.

4

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:

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)	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. 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.
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

Table Type	Description
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

System Column	Description
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 be 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

System Column	Description
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.

5 Researching Data Lineage

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

This chapter contains the following sections:

5.1 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.

5.2 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 5.3.1, "Setup Step: Configure ODI Topology and Load Plan."
2. Configure extraction scripts to generate Presentation Catalog, dashboard, and RPD metadata files, as described in Section 5.3.2, "Setup Step: Configure RPD and Presentation Catalog Extraction Scripts."
3. Create data lineage warehouse tables, as described in Section 5.3.3, "Setup Step: Create the Data Lineage Warehouse Tables."
4. Extract metadata using scripts, as described in Section 5.4.1, "Extracting Oracle Business Intelligence Metadatata Using Scripts."
5. Execute the data lineage load plan in ODI to load the data lineage warehouse tables, as described in Section 5.4.2, "Executing and Monitoring the Data Lineage Load Plan." This step loads the data lineage warehouse tables.

5.3 Tasks for Setting Up Data Lineage Dashboards

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

- Section 5.3.1, "Setup Step: Configure ODI Topology and Load Plan."
- Section 5.3.2, "Setup Step: Configure RPD and Presentation Catalog Extraction Scripts."
- Section 5.3.3, "Setup Step: Create the Data Lineage Warehouse Tables."

Note: You must perform the tasks in this section in the sequence described in Section 5.2, "Setting Up Data Lineage".

5.3.1 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 a prebuilt data lineage home directory variable which provides access to extracted metadata files, configure an adaptor list variable, then configure prebuilt ODI and Configuration Manager sources in the ODI topology so that ODI and Configuration Manager data can be sourced during data lineage ETL.

1. In ODI Studio, open the Data Lineage Exract and Load load plan.
2. Select Steps, then select the DATA_LINEAGE root step.
3. In the Property Inspector, set the DL_HOME variable to $ORACLE_BI_HOME/biapps/DataLineage.
4. Set the ADAPTOR_LIST variable by populating up to eight adaptors. If more than eight are required, you can use the ADAPTOR_LIST1 variable to add up to eight more.

Based on your source system, one or more value 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'.

5. In the Physical Architecture view of the Topology Navigator's tree view in ODI Studio, expand the technology that hosts your ODI work repository and locate the prebuilt BIAPPS_ODIREP Data Server.
6. Double-click the data server and, in the editor, verify or enter the correct connection details for the server, including the instance name and a correctly privileged ODI user and password.
7. In the JDBC tab of the editor, verify the JDBC Driver and verify or specify a valid JDBC URL for the connection.
8. Click Test Connection to verify that the connection details you have entered are correct.
9. Repeat the above steps to configure each of the following:
   - FILE_BIAPPS_DL_DEFAULT: Connects to the file location where input files are placed.
   - BIAPPS_DW: Connects to the data warehouse.
   - BIAPPS_BIACOMP: Connects to the Configuration Manager Schema.
10. Copy Fusion EAR files from $ORACLE_BI_HOME/biapps/DataLineage to $DL_HOME.

5.3.2 Setup Step: Configure RPD and Presentation Catalog Extraction Scripts

In this step, you set ODI memory heap size to accommodate extraction scripts, then customize extraction scripts which generate RPD and Presentation Catalog metadata files sourced during load of the data lineage warehouse tables.

1. Set the heap size appropriately. To do this in Windows:

   To do this in Windows, open the <DOMAIN_HOME>/bin/setDomainEnv.sh file for editing. 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 do this in Linux, open the <DOMAIN_HOME>/bin/setDomainEnv.cmd file for editing. 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
   ```

2. In $DL_HOME, open the webCatExtract script.
3. Set `OBIEE_INSTANCE_HOME` to your Oracle BI EE instance home location.

4. Set `OBIEE_WEBCAT` to the location of your Oracle BI Application Presentation Catalog location.
5. Set `TARGET_DIR` to the location where the extracted metadata files, webcat_text.txt and webcat_dashboard_test.txt, are to be created.
6. Save and close the file.
7. Open the webCatExtract_dashboard script and set the `OBIEE_INSTANCE_HOME`, `OBIEE_WEBCAT`, and `TARGET_DIR` variables as above.
8. Save and close the file.
9. Open the rpdExract script. and set OBIEE_ORACLE_HOME to the Oracle BI EE instance home location, and Presentation Catalog and repository locations.
10. Set `OBIEE_ORACLE_HOME` to the Oracle BI EE instance home location.
11. Set `OBIEE_RPD` to the Oracle BI EE repository name and location.
12. Set `TARGET_DIR` to the location where the extracted metadata file, rpd_text.txt, is to be created.
13. Save and close the file.

5.3.3 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.

1. In the ODI Studio Designer Navigator, expand BI Apps Project > Components > Generate DW DDL > Packages.
2. Execute the Generate DataLineage DDL package and monitor its execution using the Operator.

5.4 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 5.4.1, "Extracting Oracle Business Intelligence Metadatata Using Scripts."
- Section 5.4.2, "Executing and Monitoring the Data Lineage Load Plan."

Note: You must perform the tasks in this section in the sequence described in Section 5.2, "Setting Up Data Lineage".

5.4.1 Extracting Oracle Business Intelligence Metadatata Using Scripts

In this step, you copy and execute the metadata extraction scripts, then copy output files to the data lineage home directory. You can run the metadata extraction scripts to refresh your metadata.

1. Copy and execute the scripts on the Windows machine where Oracle Business Intelligence Enterprise Edition is installed.
2. Verify that the following files are generated in the location specified in the `TARGET_DIR` variable set in the scripts:
   - WebCat_text

- Webcat_dashboard_text
- rpd_text

These files contain extracted Presentation Catalog and repository metadata, and are sourced during load of the data lineage warehouse tables.

3. Copy the output files from the script execution from the TARGET_DIR location to $DL_HOME.
4. Copy Fusion EAR files from $ORACLE_BI_HOME/biapps/DataLineage to $DL_HOME

5.4.2 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.

5.5 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 5.5.1, "Overview of the Oracle BI Applications Data Lineage Dashboard"
- Section 5.5.2, "Analyzing Data Lineage for Oracle Business Intelligence Applications"

5.5.1 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:

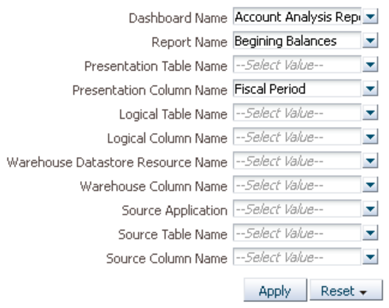
- DataLineage Summary: Provides an end-to-end data lineage summary report for physical and logical relationships. To navigate to detailed reports and dashboard pages for an entity in the Summary report, click its link.
- Dashboard Implementation: For a selected dashboard, provides reports detailing: 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.

5.5.2 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.



Dashboard Name	Account Analysis Rep
Report Name	Begining 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--

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.

A

Enabling Fusion Flex Fields for BI and Using ApplCore Grants for Data Security

This appendix describes how to enable Fusion Flex Fields for use with Oracle Business Intelligence and how to use the BI Extender to Propagate Flex Object Changes to the Oracle BI metadata repository and the Oracle Business Analytics Warehouse. It also explains how to implement ADF data security in the Oracle BI Server. To implement ADF data security, you must have an application with secured View Objects and user setup.

This appendix includes the following sections:

- Section A.1, "Setting the Extender for BI Applications Administration Tool Option"
- Section A.2, "Importing Metadata from Fusion Application ADF Data Sources"
- Section A.3, "Using the BI Extender to Propagate Flex Object Changes"
- Section A.4, "Setting Up and Using ApplCore Grants for ADF Data Security"

A.1 Setting the Extender for BI Applications Administration Tool Option

The Oracle BI Administration Tool is a Windows application that you can use to create and edit repositories. To use BI Extender for Oracle BI Applications, you must go to the Administration Tool's Options dialog and set the Extender for BI Apps option. For more information, see Section A.3, "Using the BI Extender to Propagate Flex Object Changes,".

This section describes how to open the BI Administration Tool and set the Extender for BI Apps option.

This section contains the following topics:

- Section A.1.1, "Opening the Administration Tool"
- Section A.1.2, "Setting the Extender for BI Apps Option"

A.1.1 Opening the Administration Tool

To open the Administration Tool:

To open the Administration Tool, choose **Start > Programs > Oracle Business Intelligence > BI Administration**.

Note: Do not open the Administration Tool by double-clicking a repository file. The resulting Administration Tool window is not initialized to your Oracle instance, and errors will result.

You can also launch the Administration Tool from the command line, as follows:

1. In Windows Explorer, go to the location appropriate for your install type:
   - Client installations:

     ```
     ORACLE_HOME/bifoundation/server/bin
     ```

   - All other installations:

     ```
     ORACLE_INSTANCE/bifoundation/OracleBIApplication/coreapplication/setup
     ```

2. Double-click `bi-init.cmd` (or `bi-init.bat` for client installations) to display a command prompt that is initialized to your Oracle instance.
3. At the command prompt, type admintool and press Enter.

A.1.2 Setting the Extender for BI Apps Option

You can use the Options dialog to set the Extender for BI Apps preferences for the Administration Tool. You must set this option to use BI Extender for Oracle BI Applications.

To set the Extender for BI Apps option:

1. In the Administration Tool, select **Tools**, then select **Options** to display the Options dialog.
2. On the General tab, locate the **Extender for BI Apps** option and enable it.
3. Click **OK**.

A.2 Importing Metadata from Fusion Application ADF Data Sources

This section describes how to use the Import Metadata Wizard to import metadata from the Oracle Fusion Applications ADF data sources.

See Chapter 6 Working With ADF Data Source in *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition* for more information about using ADF data sources with Oracle Business Intelligence.

This section contains the following topics:

- Section A.2.1, "Opening the Import Metadata Wizard"
- Section A.2.2, "Performing an Initial Import"

A.2.1 Opening the Import Metadata Wizard

To open the Import Metadata Wizard:

- In the Administration Tool, select **File**, then select **Import Metadata**. The Import Metadata Wizard appears.

Note: If you have already defined an existing ADF data source and connection pool, you can right-click the connection pool in the Physical layer and select **Import Metadata**. The Import Metadata Wizard appears with the information on the Select Data Source screen pre-filled.

A.2.2 Performing an Initial Import

This section describes the options you need to set when using the Administration Tool's Import Metadata Wizard to perform an initial import from Oracle Fusion Applications ADF data sources.

For more information about setting all other options in the Import Metadata Wizard, click the help button on the wizard.

- In the Metadata Wizard's Select Data Source screen, in the **User Name** and **Password** fields, you must connect as the FUSION_APPS_BI_APPID user.
- If you are importing flexfields from Oracle Fusion Applications sources, then in the Metadata Wizard's Select Metadata Objects screen, you must always import both the *name*_ and *name*_c attributes for each segment. The *name*_ attribute contains the value. The *name*_c attribute contains the code of the value set that the value comes from. Both attributes are mapped to the corresponding dimension view object. You typically use the *name*_ attribute in reports.

  For DFF segments, you can also optionally import:

  - DESC_*name*_ attribute: contains a description of the value
  - TRAN_*name*_ attribute: contains translated values, when available
- In the Map to Logical Model screen, click **Finish** to close the wizard, or click **Next** to continue to the Publish to Warehouse screen. See Section A.3.3, "Publishing Changes to the Data Warehouse and Propagating Changes to the Repository" for more information.

Note that the **Hide object if** field is predefined for Presentation layer objects imported from an ADF view object that includes the "hide" display property. When this display property is set and the ADF data source is imported into the repository, the ADF_IS_HIDDEN property is added to the physical column's properties. When a presentation column is based on a physical column with the ADF_IS_HIDDEN property, the presentation column's **Hide object if** field is set to `1`, which indicates that it is hidden.

A.3 Using the BI Extender to Propagate Flex Object Changes

You can configure and enable the BI Extender functionality to propagate changes made on Flex objects to the BI repository and optionally to the data warehouse.

Note: See also the following resources in *Oracle Fusion Applications Developer's Guide* on designating flexfields as business intelligence-enabled:

- "Preparing Descriptive Flexfield Business Components for Oracle Business Intelligence"
- "How to Prepare Key Flexfield Business Components for Oracle Business Intelligence"
- "Preparing Extensible Flexfield Business Components for Oracle Business Intelligence"

This section contains the following topics:

- Section A.3.1, "About Propagating Changes to Flex Objects to the Data Warehouse"
- Section A.3.2, "Performing Preconfiguration Tasks for the BI Extender"
- Section A.3.3, "Publishing Changes to the Data Warehouse and Propagating Changes to the Repository"
- Section A.3.4, "Automatically Publishing and Mapping Flex Object Changes Using the biserverextender Utility"
- Section A.3.5, "Setting Up XSL Transform Files to Customize XML Output to the Oracle BI Extender"

A.3.1 About Propagating Changes to Flex Objects to the Data Warehouse

You can use the Administration Tool to propagate data warehouse changes in your ADF applications through the following system components:

- Oracle Data Integrator (ODI)
- The Physical and Business Model and Mapping layers of the Oracle BI Repository.

In this scenario, the BI Extender is the driver that coordinates the information exchange between the ADF objects and the other targets.

The BI Extender feature supports changes made to *flexfields* in ADF data sources. Flexfields are columns within tables that can be reused based on a user-specified context. There are two types of flexfields:

- **Key (KFF).** These objects are modeled as dimension view objects. KFF segments are imported as new dimensions joined to an existing fact table.
- **Descriptive (DFF)** These objects are modeled as view object attributes. DFF segments are imported as new attributes (on both facts and dimensions) on existing tables.

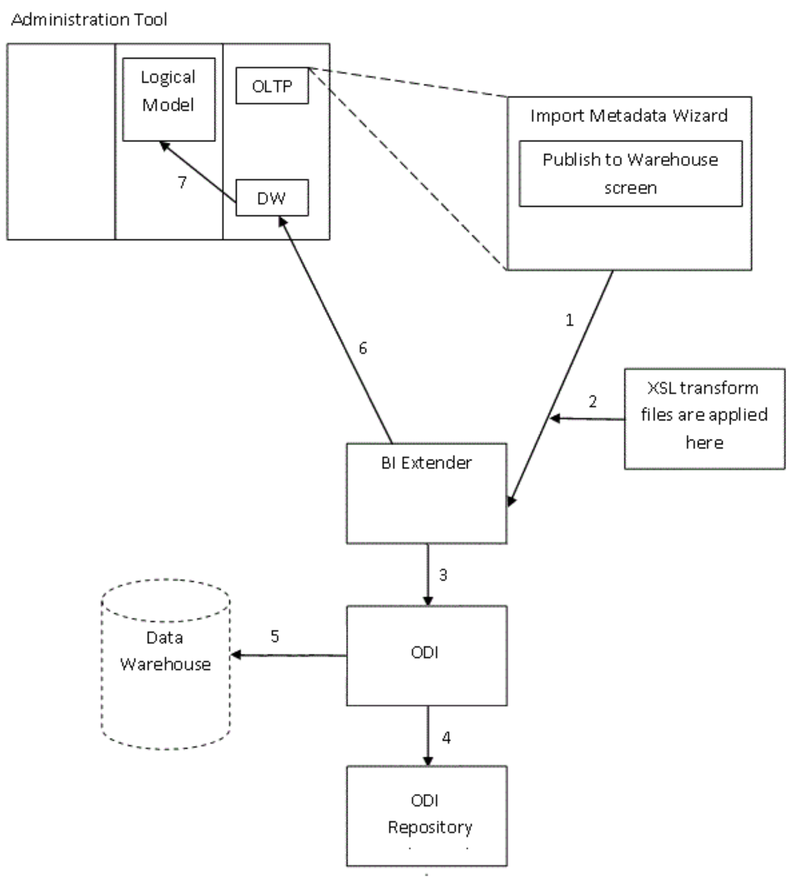
The BI Extender uses the JavaHost service to propagate flexfield changes. Because of this, JavaHost must be running for this feature to work, and the NQSConfig.INI file on the Administration Tool computer must be configured for the correct JavaHost location.

The rest of this section contains the following topics:

- Section A.3.1.1, "Overview of BI Extender Use with Oracle BI Applications"
- Section A.3.1.2, "Use Cases for Propagating Flexfield Changes"

A.3.1.1 Overview of BI Extender Use with Oracle BI Applications

Figure A–1 Flexfield Change Propagation with BI Extender and Oracle Data Integrator



In Figure A–1, numbers indicate the steps in the flexfield change propagation process. These numbers represent the following steps:

1. The Import Metadata Wizard sends XML containing the flexfield object changes to the BI Extender.
2. XSL transform files are applied to the base XML.
3. The BI Extender calls ODI.
4. The BI Extender writes to the ODI Repository.
5. ODI propagates the flexfield changes to the data warehouse.
6. The BI Extender propagates the changes to the database object for the data warehouse in the Physical layer of the Oracle BI repository.
7. The BI Extender maps the changes to the logical model.

A.3.1.2 Use Cases for Propagating Flexfield Changes

The BI Extender supports a variety of use cases for propagating changes made to flexfields. The primary use cases include:

- New attribute added on a dimension (Dimension DFF - DescriptiveFlexExtensionStandard)

- New attribute added on a fact (Fact DFF - DescriptiveFlexExtensionStandard)
- New Dimension added, joined to an existing Fact (Dimension KFF - KeyFlexCreationStandard for dimension, KeyFlexExtensionStandard for the fact foreign key)
- New Dimension added, joined to an existing Dimension (Dimension on Dimension KFF - KeyFlexCreationStandard on the new dimension, KeyFlexExtensionStandard for the foreign key)

In addition to these standard use cases, some more complex, advanced use cases are supported. The following sections describe these advanced use cases.

ETL Only View Object A view object that exists only to extend the ETL mappings, and that will not be used for queries. Imported view objects of this type are marked as disabled in the Business Model and Mapping layer. View objects of this type are marked as ETL Only in the Map to Logical Model screen of the Import Metadata Wizard.

Query Only View Object A view object that is only to be used for queries. View objects of this type are not passed to the BI Extender. View objects of this type are marked as Query Only in the Map to Logical Model screen of the Import Metadata Wizard.

View Objects that Map to Existing Dimensions This occurs when the imported view object is mapped to an existing logical dimension table that already maps to the data warehouse. In this case, the BI Extender uses the existing columns to perform the extensions. A different transformation template (from the specified XSL transform file, discussed in Section A.3.5, "Setting Up XSL Transform Files to Customize XML Output to the Oracle BI Extender") is applied for this use case, as shown in Table A–1.

Table A–1 Normal and Mapped to Existing Transformation Template Names

Normal Template	"Mapped to Existing" Template
KeyFlexCreationStandard	KeyFlexCreationExtract
DescriptiveFlexCreationStandard	DescriptiveFlexCreationExtract
KeyFlexHierarchyCreationStandard	KeyFlexHierarchyCreationExtract
DescriptiveFlexExtensionStandard	DescriptiveFlexExtensionExtract
KeyFlexExtensionStandard	KeyFlexExtensionExtract

Hierarchy View Object A special type of view object that is treated differently by the BI Extender. You can specify a Hierarchy view object in the updatehierarchy.xsl file, where you specify the HIERARCHY_NAME and DATASOURCE_NUM_ID (required attributes for Hierarchy view objects). This tells the Administration Tool which view objects are Hierarchy view objects. In the Map to Logical Model screen of the Import Metadata Wizard, the Hierarchy check box appears as selected for each Hierarchy view object.

Striping (W_GL_SEGMENT_D) Some view objects come with predefined filters. These filter definitions are automatically propagated to the appropriate logical table source content filter.

The BI Extender configures the logical table source filters for GL accounts by putting appropriate segment labels in the filters. There is a right-click option for GL-SegmentX dimension tables to pull in the synchronized custom AM properties into the logical table source filter.

A.3.2 Performing Preconfiguration Tasks for the BI Extender

To enable the BI Extender feature with ODI, you must configure a connection to ODI and the data warehouse in the biextension.properties file. You also need to provide configuration information in the loaders.xml file for JavaHost.

The following sections describe the steps needed for this configuration:

- Section A.3.2.1, "Configuring the biextension.properties File"
- Section A.3.2.2, "Configuring the JavaHost loaders.xml File"
- Section A.3.2.3, "Updating NQSConfig.INI"
- Section A.3.2.4, "Optionally Changing the Location of the BI Extender Files"

A.3.2.1 Configuring the biextension.properties File

This section explains how to configure the biextension.properties file.

To configure connections in biextension.properties:

1. Open the biextension.properties file for editing. You can find this file at:

   ```
   MW_HOME/Oracle_BI1/bifoundation/javahost/lib/obisintegration/biextender
   ```

2. Set the parameters in biextension.properties as needed for your deployment.

   The odi.* parameters define the connection to ODI.

   The following table describes the parameters in biextension.properties:

Parameter	Description
odi.connection.sdk.masterdriver	The Oracle JDBC Driver Set this to oracle.jdbc.driver.OracleDriver *(exact value, do not change).*
odi.connection.sdk.masterurl	The JDBC url for the ODI repository database (for example, jdbc:oracle:thin:@localhost:1521:orcl).
odi.connection.sdk.workrepositoryname	The ODI work repository name.

   Note that for Oracle Database, the driver parameters can use either a service name or SID. For example:

   - jdbc:oracle:thin:@host:port/service name
   - jdbc:oracle:thin:@host:port:SID

3. Save and close the file.

A.3.2.2 Configuring the JavaHost loaders.xml File

This section explains how to configure the JavaHost loaders.xml file.

To configure the IntegrationServiceCall loader in loaders.xml:

1. Open the JavaHost loaders.xml file for editing. You can find this file at:

   ```
   ORACLE_HOME\bifoundation\javahost\config
   ```

2. Find the Loader section for IntegrationServiceCall. Then, edit the section to include the location of oracle.odi-sdk.jse_11.1.1.jar, as shown in the following example. Typically ODI client files are installed into C:\oracle\product\11.1.1\Oracle_ODI_1.

```
<Loader>
 <Name>IntegrationServiceCall</Name>
 <Class>oracle.bi.integration.javahost.ServiceCallLoader</Class>
 <ConfigNodePath>ServiceCall</ConfigNodePath>
 <ClassPath>
  {%ORACLE_BIJH_ROOTDIR%}/lib/obisintegration/javahostservice.jar;
  {%ORACLE_BIJH_ROOTDIR%}/lib/obisintegration/aw/11g/ojdbc5.jar;
  C:\oracle\product\11.1.1\Oracle_ODI_
1\setup\manual\oracledi-sdk\oracle.odi-sdk-jse_11.1.1.jar
 </ClassPath>
</Loader>
```

Important: Be sure to replace the path name in the highlighted line with the appropriate path names for your deployment.

3. If the JavaHost is running on a Linux system, you must also download log4j-1.2.16.jar from http://www.apache.org and specify the classpath to this jar file in loaders.xml.
4. Save and close the file.

A.3.2.3 Updating NQSConfig.INI

If you are using a client installation of the Administration Tool, such as when you are running the JavaHost on a Linux system, you must also update the NQSConfig.INI file on the Administration Tool computer to point to the location of a running JavaHost. To do this, follow these steps:

1. Close the Administration Tool, if it is open.
2. On the same computer as the Administration Tool, open the local NQSConfig.INI file in a text editor. You can find this file at:

   ```
   ORACLE_INSTANCE/config/OracleBIServerComponent/coreapplication_obisn
   ```

3. Locate the JAVAHOST_HOSTNAME_OR_IP_ADDRESSES parameter, near the bottom of the file. Update this parameter to point to a running JavaHost, using a fully-qualified host name or IP address and port number. For example:

   ```
   JAVAHOST_HOSTNAME_OR_IP_ADDRESSES = "myhost.example.com:9810"
   ```

   Note that in a full (non-client) Oracle Business Intelligence installation, you cannot manually edit this setting because it is managed by Oracle Enterprise Manager Fusion Middleware Control.
4. Save and close the file.

These steps are only required for client installations of the Administration Tool.

A.3.2.4 Optionally Changing the Location of the BI Extender Files

Optionally, you can change the location of the BI Extender files. To do this, move the files from the default location to a new location, then create an operating system environment variable called ORACLE_BI_ETL_EXTENDER and set its value to the value of the new path. The default path for the BI Extender files is:

```
ORACLE_HOME/bifoundation/javahost/lib/obisintegration/biextender
```

A.3.3 Publishing Changes to the Data Warehouse and Propagating Changes to the Repository

This section explains how to use the BI Extender to propagate the flex object changes to ODI, as well as to the Physical layer and logical model objects. You must complete the preconfiguration steps in Section A.3.2, "Performing Preconfiguration Tasks for the BI Extender" before performing the steps in this section.

This section contains the following topics:

- Section A.3.3.1, "Running the BI Extender to Update ODI and the RPD"

A.3.3.1 Running the BI Extender to Update ODI and the RPD

This section explains how to run the BI Extender to update ODI and the Oracle BI repository for flex object changes. This section assumes that you have already added a new attribute in your ADF application.

To run the BI Extender to update ODI and the RPD:

1. Ensure that the JavaHost process is running.
2. Open your repository in the Administration Tool.
3. In the Physical layer, right-click the connection pool for your ADF OLTP source and select **Import Metadata**.
4. Complete the steps in the Select Metadata Objects and Map to Logical Model screens. For more information, click the help buttons on these screens.
5. On the Publish to Warehouse screen, perform the following steps:

   ODI Implementations

   a. For ODI implementations, select the warehouse database, and select ODI.
   b. For User Name and Password, enter the ODI Repository user name and password.
   c. For the Schema Owner Name and Password, enter the database user name and password where the ODI Master Repository resides.
   d. Click **Test Connection**.
6. Click **Finish**.

See Section A.3.1, "About Propagating Changes to Flex Objects to the Data Warehouse" for a description of what happens when you click **Finish** in the Publish to Warehouse screen.

A.3.4 Automatically Publishing and Mapping Flex Object Changes Using the biserverextender Utility

You can use the `-E` option with the `biserverextender` utility to automatically publish and map flex object to the repository for Fusion Applications ADF data sources. This topic explains the `-E` option, only, which is specific to using the `biserverextender` utility with Oracle Fusion Applications ADF data sources. See "Automatically Mapping Flex Object Changes Using biserverextender Utility" in *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition* for complete information about the biextender utility, syntax options, and setting up the required XML parameter file.

Using the `biserverextender` utility does not require the Administration Tool, and is therefore especially useful when you want to publish changes and map flex object changes on Linux and UNIX systems where the Administration Tool is not available.

Syntax

```
biserverextender -P repository_password -R base_repository_name -I input_XML_file
-O output_repository_name -J oracle_enterprise_scheduler_id -E run_ETL_extension_
process
```

Example

```
biserverextender -R scratch/my_repos.rpd -I /scratch/ADFSource.xml -O /scratch/my_
repos_modelled.rpd -J 300 -E
Give password: password
```

Sample ODI XML Parameter File

```
<BIExtenderParameters updateAMPropertiesForTest="true">
   <ConnectionDetails>
    <ConnectionPool>
     <ConnectionPoolName>
"oracle.apps.fscm.model.analytics.applicationModule.FscmTopModelAM_
FscmTopModelAMLocal"."Connection Pool"
     </ConnectionPoolName>
    </ConnectionPool>
   </ConnectionDetails>
   <ETLExtensionDetails dataWarehouse="Oracle DataWarehouse" sendExtension="true"
generateXML="true">
    <ODI>
     <Username>username1</Username>
     <Password> D7EDED84BC624A917F5B462A4DCA05CDCE256EEEEEDC97D5AC4D07C3A079829F
     </Password>
      <MasterUserName>username2</MasterUserName>
      <MasterPassword>
D7EDED84BC624A917F5B462A4DCA05CDCE256EEEEEDC97D5AC4D07C3A079829F
      </MasterPassword>
    </ODI>
   </ETLExtensionDetails>
</BIExtenderParameters>
```

Where:

`dataWarehouse` is the name of the data warehouse to where the changes will be propogated.

`sendExtension` specifies whether to log the connection and XML in the Admintool log file. The default values is `false`.

`generateXML` specifies whether to generate the XML output file. The default value is `false`.

`ODI` specifies the ODI connection details of the user and the master user.

Note that in this example, `Password` and `MasterPassword` are encrypted values. When submitting the job through Oracle Enterprise Scheduler Service, the scheduler provides the encryption.

If you run the `biserverextender` utility manually by entering the details in the input XML file, then you can use the QSecUDMLGen utility to generate the encrypted password. When you run the QSecUDMLGen utility with the `-P` option, you will be prompted to supply a password. When you enter the password, the utility generates the encrypted string for the password.You then add this string (encrypted password) to the input XML file.

A.3.5 Setting Up XSL Transform Files to Customize XML Output to the Oracle BI Extender

The biextension.properties file specifies the XSL files to be used to transform and customize the default XML generated by the Administration Tool and sent to the BI Extender. By default, the following XSL files are applied:

- biextension.xsl: Contains default transformations for the Extender. Contains templates for both normal and "mapped to existing" cases.
- updatehierarchy.xsl: Specifies rules for how Hierarchy view objects should be handled.
- LastUpdateDate.xsl: Identifies which input columns need to be filtered for Change Data Capture.

All XSL files are located in the same directory as biextension.properties.

In rare cases, you might need to make additional customizations the default XML that is generated by the Administration Tool and sent to the BI Extender. To do this, you can create other XSL files to be applied in addition to the three default XSL files. Note the following:

- Additional XSL files need to conform to the XML schema defined in the biextension.xsd file, located in the same directory as the biextension.properties file.
- It is a best practice to define changes in an additional XSL file rather than updating one of the default files.
- The replaceName.xsl file provides examples on how to do XSL transforms for name changes.

To specify additional XSL files:

1. Open the biextension.properties file for editing. You can find this file at:

   ```
   ORACLE_HOME/bifoundation/javahost/lib/obisintegration/biextender
   ```

2. Add additional XSL files to the xsl_transforms line, as follows:

   ```
   xsl_transforms = updatehierarchy.xsl,LastUpdateDate.xsl,biextension.xsl,
   replaceNames.xsl
   ```

3. Save and close the file.

A.3.5.1 Sample XML Output

Sample base XML output generated by the Administration Tool might look like the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Document>
  <extension mode="KeyFlexCreationStandard" type="Dimension">
    <table name="W_COST_CENTER_D">
      <columns>
        <column datatype="Varchar2(50)" name="COST_CENTER_NAME" type="Attribute">
          <source column="COST_CENTER_NAME" table="SnowflakesalesApp.
          ADF_COST_CENTER_VO"/>
```

```
        </column>
        <column datatype="Varchar2(10)" name="COST_CENTER_LOCATION"
        type="Attribute">
          <source column="COST_CENTER_LOCATION" table="SnowflakesalesApp.
          ADF_COST_CENTER_VO"/>
        </column>
        <column datatype="Varchar2(5)" name="COST_CENTER_ID" type="Key">
          <source column="COST_CENTER_ID" table="SnowflakesalesApp.
          ADF_COST_CENTER_VO"/>
        </column>
      </columns>
    </table>
  </extension>
</Document>
```

A.4 Setting Up and Using ApplCore Grants for ADF Data Security

Note: This section applies to Fusion Applications source systems only.

This section explains how to implement ADF data security in the Oracle BI Server. To implement ADF data security, you must have an application with secured View Objects and user setup.

Perform the steps in the following sections:

- Section A.4.1, "Setting Up Oracle Business Intelligence to Use ApplSession"
- Section A.4.2, "Setting Up Authentication for ApplSession Integration"

A.4.1 Setting Up Oracle Business Intelligence to Use ApplSession

When you set up Oracle Business Intelligence to use ApplSession, the following occurs:

- Pillar-specific AOL sessions are created for data queries. A new AOL session is created for every pillar within a BI session when a data query is fired against the pillar. This AOL session is reused for all subsequent data queries against the same pillar. There are, at most, as many AOL sessions as there are ADF pillars (databases) defined in the Oracle BI repository.

  For the examples used in this section, there are three or less AOL sessions ("AOL_SESSION_ID_HCM", "AOL_SESSION_ID_CRM", "AOL_SESSION_ID_FSCM") created within a BI Session for data queries.
- AOL context variable values are propagated to the newly-created pillar-specific AOL sessions. Presentation Services propagates the AOL context variables when it logs in to the Oracle BI Server.
- The SQL Bypass query reattaches to the previously created AOL session for that ADF Database.

This section contains the following topics:

- Section A.4.1.1, "Setting Up Database Objects and Connection Pools for ApplSession Integration"
- Section A.4.1.2, "Setting Up Initialization Blocks for ApplSession Integration"

- Section A.4.1.3, "About the Client Login Process in an ApplSession Integrated Environment"

A.4.1.1 Setting Up Database Objects and Connection Pools for ApplSession Integration

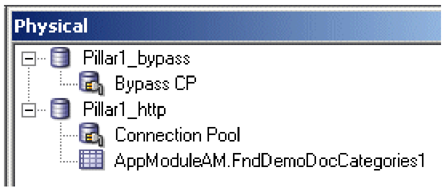
To set up the Oracle BI repository to enable ApplSession integration, you must first configure the appropriate database object and connection pools in the Physical layer.

To set up database objects and connection pools for ApplSession integration:

1. Open your repository in the Administration Tool.
2. Create database objects and connection pools in the physical layer for each pillar, as follows:
   - Create a database object and connection pool for SQL bypass during data queries (for example, Pillar1_bypass).
   - Use the Import Metadata Wizard to create a database object and connection pool that correspond to the application EAR file deployed in WLS (for example, Pillar1_http). This database object contains the physical table and column mappings to view objects and attributes. Be sure to specify the SQL bypass database you want to use during metadata import.

   Figure A–2 shows the pillar-specific database objects.

Figure A–2 Pillar-Specific Database Objects



3. Open the connection pool for each SQL bypass database and click the Connection Scripts tab. To ensure that SQL bypass queries are issued within the pillar-specific AOL session (for example, "AOL_SESSION_ID_Pillar1_http,") you must provide the appropriate pre-query and post-query scripts, as follows:

   a. Expand **Execute before query** and click **New**.

   b. Enter a pre-query script similar to the following:

```
BEGIN
  fnd_session_mgmt.attach_session('VALUEOF(NQ_SESSION.AOL_SESSION_ID_
Pillar1_http)');
END;
```

   c. Click **OK**.

   d. Expand **Execute after query** and click **New**.

   e. Enter the following post-query script:

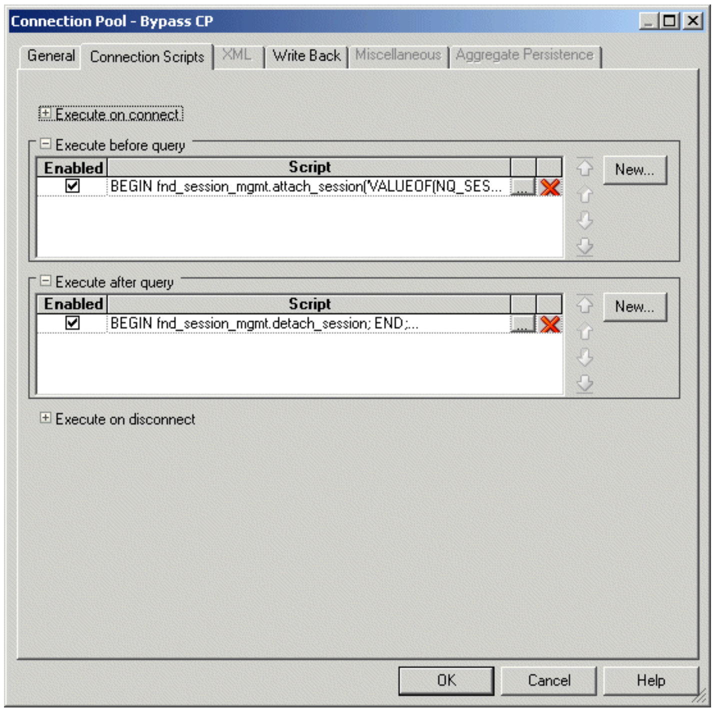
```
BEGIN
  fnd_session_mgmt.detach_session;
END;
```

   f. Click **OK**.

g. Click **OK** in the Connection Pool dialog.

Figure A–3 shows the Connection Scripts tab of the Connection Pool dialog.

Figure A–3 Pre-query and Post-query Scripts for the SQL Bypass Connection Pool



Enabled	Script
✓	BEGIN fnd_session_mgmt.attach_session('VALUEOF(NQ_SES...

Enabled	Script
✓	BEGIN fnd_session_mgmt.detach_session; END;...

A.4.1.2 Setting Up Initialization Blocks for ApplSession Integration

Initialization blocks are used to initialize BI session variables. They contain SQL/XML queries that are set up to return columns of values which are then mapped to BI session variables. Initialization blocks typically execute during BI session creation.

In the Initialization Block dialog, you can choose to defer the execution of an initialization block by selecting **Allowed deferred execution**. A deferred initialization block runs when its target variable is used for the first time within a BI session.

To set up initialization blocks for ApplSession integration:

1. In the Administration Tool, select **Manage**, then select **Variables**.
2. Select **Action > New > Session > Initialization Block**.
3. Provide a name for the initialization block (for example, create_AOL_SESSION_ID_Pillar1). This initialization block creates a new AOL session when a data query is issued against the data source (for example, Pillar1_http).
4. Click **Edit Data Source**.
5. Select **Default initialization string**, and provide an XML initialization string similar to the following:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<ADFQuery mode="create_applsession">
  <ContextAttribute>
    <Name><![CDATA[ADDTL_CUSTOM_LEVEL]]></Name>
    <Value><![CDATA[VALUEOF(NQ_SESSION.AOL_ADDTL_CUSTOM_LEVEL)]]></Value>
  </ContextAttribute>
  <ContextAttribute>
    <Name><![CDATA[CLIENT_ENCODING]]></Name>
    <Value><![CDATA[VALUEOF(NQ_SESSION.AOL_CLIENT_ENCODING)]]></Value>
  </ContextAttribute>
  <ContextAttribute>
    <Name><![CDATA[CURRENCY]]></Name>
    <Value><![CDATA[VALUEOF(NQ_SESSION.AOL_CURRENCY)]]></Value>
  </ContextAttribute>
</ADFQuery>
```

This initialization string causes the OBIEE broker servlet to create a Java AOL session.

The <ContextAttribute> elements represent the name-value pairs of context variables that will be propagated to the newly created AOL session. If no <ContextAttribute> elements are specified in the XML initialization string, default values of the user are used to create the AOL session.

The values provided in the <Value> elements are mapped to the session variables initialized by Presentation Services.

The following list shows a full mapping between AOL context attributes and session variables initialized by Presentation Services:

```
ACCESSIBILITY_MODE - AOL_ACCESSIBILITY_MODE
ACTION - AOL_ACTION
ADDTL_CUSTOM_LEVEL - AOL_ADDTL_CUSTOM_LEVEL
ANIMATION_ENABLED - AOL_ANIMATION_ENABLED
APPLICATION_LANGUAGE - AOL_APPLICATION_LANGUAGE
CLIENT_ENCODING - AOL_CLIENT_ENCODING
COLOR_CONTRAST - AOL_COLOR_CONTRAST
CURRENCY - AOL_CURRENCY
DATE_FORMAT - AOL_DATE_FORMAT
DECIMAL_SEPARATOR - AOL_DECIMAL_SEPARATOR
EMBEDDED_HELP_ENABLED - AOL_EMBEDDED_HELP_ENABLED
FONT_SIZE - AOL_FONT_SIZE
GROUPING_SEPARATOR - AOL_GROUPING_SEPARATOR
HISTORY_OVERRIDE_USER_NAME - AOL_HISTORY_OVERRIDE_USER_NAME
INDUSTRY - AOL_INDUSTRY
INDUSTRY_IN_TERRITORY - AOL_INDUSTRY_IN_TERRITORY
LANGUAGE - AOL_LANGUAGE
MODULE - AOL_MODULE
NLS_SORT - AOL_NLS_SORT
NUMBER_FORMAT - AOL_NUMBER_FORMAT
PRODUCT - AOL_PRODUCT
PRODUCT_FAMILY - AOL_PRODUCT_FAMILY
TERRITORY - AOL_TERRITORY
TIME_FORMAT - AOL_TIME_FORMAT
TIMEZONE - AOL_TIMEZONE
TRACE_LEVEL - AOL_TRACE_LEVEL
```

Application-specific, nonstandard variables that do not appear in the preceding list can be propagated in the same manner.

6. Click **Browse** next to **Connection Pool** and select the connection pool for the data source (for example, Pillar1_http_cp), then click **Select**.
7. Click **OK** in the Session Variable Initialization Block Data Source dialog.
8. Click **Edit Data Target**, then click **New**.

9. Create a variable to be initialized by this initialization block. The name must be in the following format:

   AOL_SESSION_ID_*physical_layer_database_object_for_pillar*

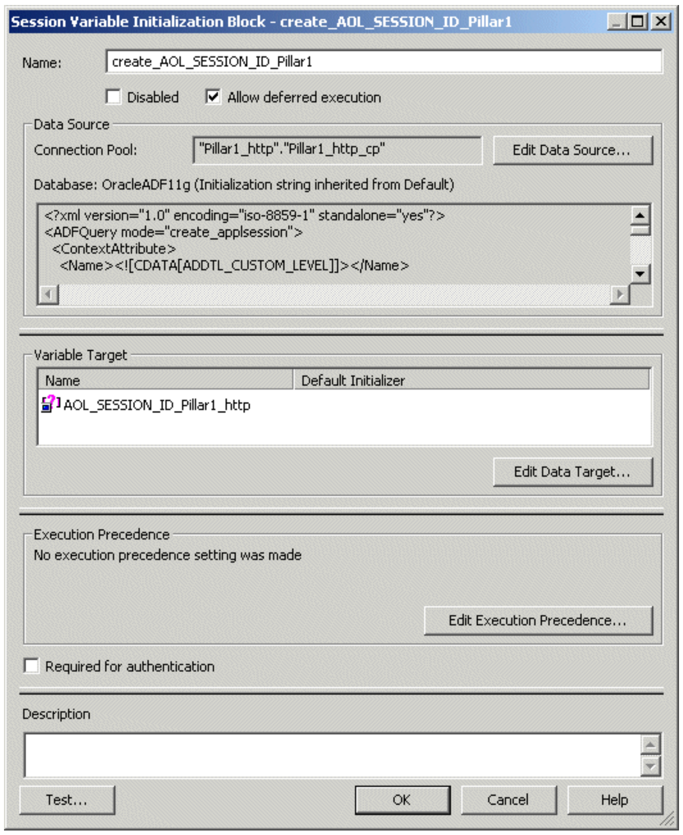
   For example:

   AOL_SESSION_ID_Pillar1_http

10. Click **OK** in the Session Variable Initialization Variable Target dialog.
11. Select **Allowed deferred execution**. Selecting this option means that this initialization block only executes when a data query is issued against Pillar1.
12. Click **OK** in the Session Variable Initialization Block dialog.
13. Repeat the steps in this procedure for each pillar, if you have more than one.

Figure A–4 shows an example pillar-specific session variable initialization block.

Figure A–4 Session Variable Initialization Block Dialog for Pillar-Specific Initialization Block


Session Variable Initialization Block - create_AOL_SESSION_ID_Pillar1

Name: create_AOL_SESSION_ID_Pillar1

☐ Disabled ☑ Allow deferred execution

Data Source

Connection Pool: "Pillar1_http"."Pillar1_http_cp"

Database: OracleADF11g (Initialization string inherited from Default)

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<ADFQuery mode="create_applsession">
 <ContextAttribute>
  <Name><![CDATA[ADDTL_CUSTOM_LEVEL]]></Name>
```

Variable Target

Name	Default Initializer
AOL_SESSION_ID_Pillar1_http	

Execution Precedence

No execution precedence setting was made

☐ Required for authentication

Description

A.4.1.3 About the Client Login Process in an ApplSession Integrated Environment

The following steps provide an example of the client login process when you have integrated Oracle Business Intelligence with ApplSession:

1. A client logs in to the Oracle BI Server with the BI session variables that represent AOL context.
2. The BI Session is established.

3. The client issues a Logical SQL statement that navigates to the database object (for example, Pillar1_http).
4. The initialization block runs and initializes the session variable (for example, create_AOL_SESSION_ID_Pillar1 initializes AOL_SESSION_ID_Pillar1_http).
5. The session variable (for example, AOL_SESSION_ID_Pillar1_http) is attached within the OBIEE broker servlet before querying Composite view object SQL from the database object (for example, Pillar1_http).
6. The Oracle BI Server replans the query with Composite view object SQL.
7. The session variable (for example, AOL_SESSION_ID_Pillar1_http) is attached using the pre-query script in the connection pool before querying data directly from the SQL bypass database object (for example, Pillar1_bypass).

A.4.2 Setting Up Authentication for ApplSession Integration

Oracle Business Intelligence provides a custom identity assertion provider that must be installed in Oracle WebLogic Server. This custom asserter is used to extract a token from the HTTP request from Oracle Business Intelligence. The super user name and password provided in the repository connection pool object, as well as the session user name provided in the BI user session, are extracted from the token.

Authentication is performed in the custom asserter using the super user name and password. After the super user's credentials are authenticated, the custom asserter performs identity assertion using the session's user name. One-way SSL is required to secure the communication channel between Oracle Business Intelligence and Oracle WebLogic Server.

Note that the custom identity assertion provider only allows a user with the user name FUSION_APPS_BI_APPID to perform identity assertion. In other words, the user specified in the connection pool (the ADF HTTP connection pool, not the SQL bypass connection pool) must be FUSION_APPS_BI_APPID.

This section describes the configuration tasks you need to perform in the application before deploying to Oracle WebLogic Server for Oracle Business Intelligence consumption.

Ensure that you have correctly set up the ADF data sources before you perform the steps in the following sections.

This section contains the following topics:

- Section A.4.2.1, "Setting Up Security Constraints and Security-Related Servlet Filters in web.xml"
- Section A.4.2.2, "Configuring Role-to-Principal Mapping in weblogic-application.xml"
- Section A.4.2.3, "Configuring the Custom Identity Assertion Provider in Oracle WebLogic Server"
- Section A.4.2.4, "Configuring One-Way SSL in Oracle WebLogic Server"

A.4.2.1 Setting Up Security Constraints and Security-Related Servlet Filters in web.xml

You must set up the CLIENT-CERT authentication method in web.xml to trigger the custom identity assertion provider during log-on. In addition, you need to set up the JPS servlet filter in web.xml for data security to work properly.

To set up the CLIENT-CERT authentication method and servlet filters in web.xml:

1. In JDeveloper, expand the Web Project for OBIEEBroker and open web.xml.
2. Go to the source view of the file.
3. To set up the CLIENT-CERT authentication method, include the following elements under the <web-app> element:

   ```
   <security-constraint>
     <web-resource-collection>
       <web-resource-name>restricted</web-resource-name>
       <url-pattern>/obieebroker</url-pattern>
     </web-resource-collection>
     <auth-constraint>
       <role-name>EndUsers</role-name>
     </auth-constraint>
   </security-constraint>
   <login-config>
     <auth-method>CLIENT-CERT</auth-method>
     <realm-name>myrealm</realm-name>
   </login-config>
   <security-role>
     <role-name>EndUsers</role-name>
   </security-role>
   ```

   Note that the value for <role-name> can be anything. This value will eventually be mapped to principals that exist in the security realm; see Section A.4.2.2, "Configuring Role-to-Principal Mapping in weblogic-application.xml" for more information.
4. To set up the JPS servlet filter, include the following <filter> elements before the <filter> element for ServletADFFilter:

   ```
   <filter>
     <filter-name>JpsFilter</filter-name>
     <filter-class>oracle.security.jps.ee.http.JpsFilter</filter-class>
   </filter>
   ```

   Then, include the following <filter-mapping> elements before the <filter-mapping> element for ServletADFFilter:

   ```
   <filter-mapping>
     <filter-name>JpsFilter</filter-name>
     <servlet-name>OBIEEBroker</servlet-name>
     <dispatcher>FORWARD</dispatcher>
     <dispatcher>REQUEST</dispatcher>
     <dispatcher>INCLUDE</dispatcher>
   </filter-mapping>
   ```

A.4.2.2 Configuring Role-to-Principal Mapping in weblogic-application.xml

You must map the value you provided for <role-name> in web.xml to principals that exist in the security realm. This mapping is defined in weblogic-application.xml.

To configure role-to-principal mapping in weblogic-application.xml:

1. In JDeveloper, under Application Resources, open Descriptors > META-INF > weblogic-application.xml.
2. Include the following <security> element under the <weblogic-application> element:

   ```
   <security>
     <realm-name>myrealm</realm-name>
   ```

```
  <security-role-assignment>
    <role-name>your_role_name</role-name>
    <principal-name>your_principal_name</principal-name>
  </security-role-assignment>
</security>
```

Note the following:

- Replace *your_role_name* with the actual value you provided for <role-name> in web.xml.
- Replace *your_principal_name* with the name of either a BI session end user (for example, joeUser), or an Oracle WebLogic Server group (for example, testUsers).

3. Redeploy the application.

A.4.2.3 Configuring the Custom Identity Assertion Provider in Oracle WebLogic Server

This section explains how to configure the custom identity assertion provider.

To configure the custom identity assertion provider in Oracle WebLogic Server:

1. Locate the custom identity assertion provider jar file, located in your Oracle Business Intelligence installation directory at:

   ```
   ORACLE_HOME/bifoundation/javahost/lib/obisintegration/adf/
   biadfidentityasserter.jar
   ```

2. Copy the jar file to the following Oracle WebLogic Server location:

   ```
   MW_HOME/wlserver_10.3/server/lib/mbeantypes
   ```

3. Restart the Oracle WebLogic Server.
4. Open the WebLogic Server Administration Console. For example, if your Oracle WebLogic Server is running locally on port 7001, go to `http://localhost:7001/console`.
5. Log in to the WebLogic Server Administration Console with the credentials you created when you set up your WebLogic domain.
6. In the Change Center, click **Lock & Edit**.
7. In the left pane, select **Security Realms**, and then click the security realm that you want to configure.
8. Click the Providers tab, then click **New**.
9. Enter a name (for example, BI-ADF IdentityAsserter) and select **BIADFIdentityAsserter** for **Type**.
10. Click **OK**.
11. In the Change Center, click **Activate Changes**.
12. Restart the Oracle WebLogic Server.

A.4.2.4 Configuring One-Way SSL in Oracle WebLogic Server

One-way SSL is required to properly secure the communication between Oracle Business Intelligence and Oracle WebLogic Server.

To configure one-way SSL in Oracle WebLogic Server:

1. From the WebLogic Server Administration Console home page, click **Servers** under the **Environment** heading.
2. In the Servers table, the name of the server you want to manage. Then, on the General subtab of the Configuration tab, select **SSL Listen Port Enabled**.
3. Use the Administration Tool to update the appropriate connection pool object in the Physical layer so that the URL uses https:// instead of http://. Also, update the port number to use the SSL port (7002 by default).

Index