

Oracle® Business Intelligence Applications

ETL Guide

11g Release 1 (11.1.1.10.2)

E72285-04

June 2017

Explains how to set up, perform, and monitor ETL processes.

Copyright © 2014, 2017, Oracle and/or its affiliates. All rights reserved.

Primary Author: Padma Rao

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

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

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

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

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

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

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

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

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

Contents

Preface	v
Audience	v
Documentation Accessibility	v
Related Documents.....	v
Conventions.....	v
 1 ETL Overview	
Before Running ETL	1-1
About ETL Architecture.....	1-1
About ETL Phases.....	1-3
About the Oracle Data Integrator Repository	1-3
About Load Plans	1-4
About Changed Data Capture	1-4
About Knowledge Modules	1-5
About Reverse Knowledge Modules	1-6
About Multi-Source Environments.....	1-6
About ETL Roles	1-8
About Cloud Sources	1-9
 2 Managing Load Plans	
Overview of Load Plan Life Cycle	2-1
Defining Load Plans	2-2
Duplicating Load Plans.....	2-3
Editing Load Plans.....	2-4
Generating Load Plans.....	2-6
Scheduling Load Plans.....	2-6
Executing Load Plans	2-7
About Diagnostics Health Check and ETL Diagnostics and Automatic Correction	2-9
Downloading Diagnostic Health Check Reports	2-11
Downloading Error Data from Diagnostics.....	2-12
Monitoring Load Plan Runs.....	2-12
Restarting Load Plans	2-13

About Restartability Grain	2-13
Restarting Load Plans Using ODI Studio	2-14
Troubleshooting Load Plans	2-15
Alternate Options for Restarting Load Plans	2-16
Related Features and Considerations	2-17
Stopping Load Plans	2-18
3 Additional Administration Tasks	
Additional Administration Tasks	3-1
Using a Database Link for ETL	3-2
Fixing Event Queue Knowledge Module	3-3
4 Customizing the Oracle Business Analytics Warehouse	
What Is Customization in Oracle BI Applications?	4-1
About the Customization Process	4-2
Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables	4-5
Extending Mappings in Oracle Business Analytics Warehouse	4-5
Example of Extending the Oracle Business Analytics Warehouse	4-8
Other Types of Customizations Requiring Special Handling	4-15
Category 2 Customizations: Adding Additional Tables	4-20
Additional Information About Customizing	4-21
Adding New Fact Tables to Oracle Business Analytics Warehouse	4-21
Creating Custom Dimension Datastore and Tasks	4-24
About Creating Custom Fact and Dimension Groups	4-26
Category 3 Customizations: Adding New Data	4-30
Adding New Data as a Whole Row Into a Standard Dimension Table	4-30
Configuring Extracts	4-30
Configuring Loads	4-32
Customizing Stored Lookups and Adding Indexes	4-34
How Dimension Keys are Looked Up and Resolved	4-34
Adding an Index to an Existing Fact or Dimension Table	4-35
About Resolving Dimension Keys	4-35
Custom Domains	4-36
Extending Oracle BI Applications with Custom Source Domains	4-37

Preface

Oracle Business Intelligence Applications (Oracle BI Applications) is a 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 managers and implementers of Oracle BI Applications.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

See the Oracle BI Applications documentation library for the complete set of Oracle BI Applications documents.

Conventions

These 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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

ETL Overview

The Oracle Business Intelligence Applications (Oracle BI Applications) Extract, Transform, and Load (ETL) architecture and supporting technologies provide the ability to load data into your data warehouse.

Topics

- [Before Running ETL](#)
- [About ETL Architecture](#)
- [About ETL Phases](#)
- [About the ODI Repository](#)
- [About Load Plans](#)
- [About Changed Data Capture](#)
- [About Knowledge Modules](#)
- [About Reverse Knowledge Modules](#)
- [About Multi-Source Environments](#)
- [About ETL Roles](#)
- [About Cloud Sources](#)

Before Running ETL

Before you begin running Oracle BI Applications ETL processes, you must have completed the installation and setup of Oracle BI Applications.

To install Oracle BI Applications, see *Installing and Setting Up Oracle BI Applications* in *Oracle Business Intelligence Applications Installation Guide*.

You must also have run the domains load plan, which loads source-specific data into Oracle BI Applications Configuration Manager tables. This enables Configuration Manager to display the appropriate source-specific values as choices in drop-down lists for setup objects. To run the domains load plan, see *Running the Domains Load Plan* in *Oracle Business Intelligence Applications Installation Guide*.

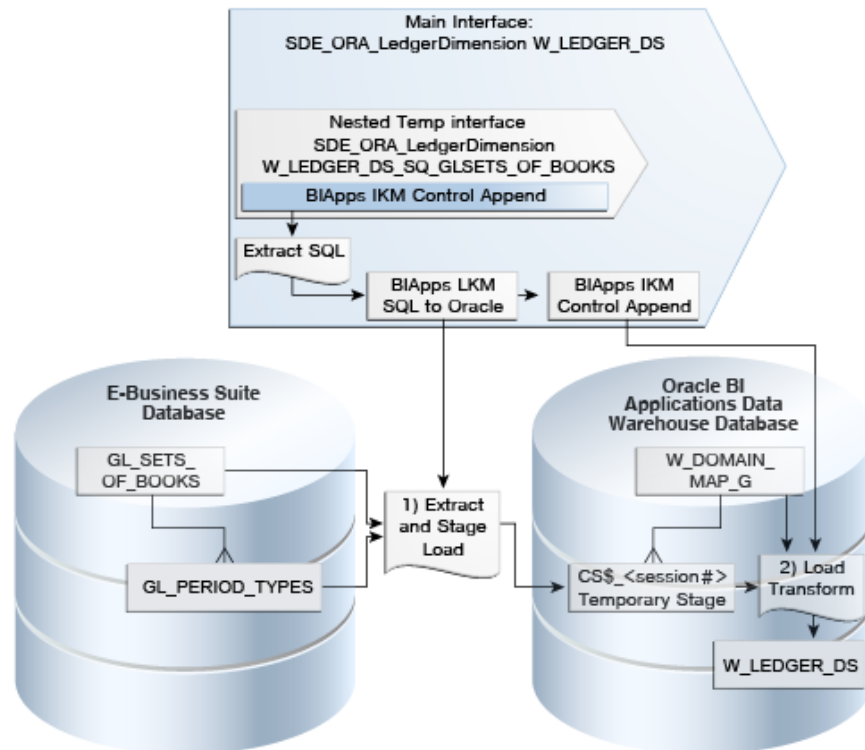
About ETL Architecture

Typically, the extract-load-transform process has two main steps: The first step is the extract and stage load step and the second step is the load transform step.

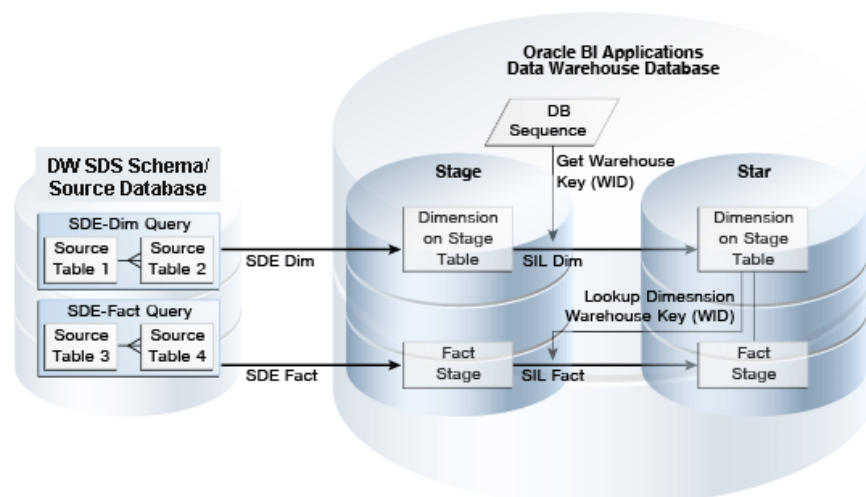
The extract and stage load step is generated from a combination of the main interface and the temporary interface. The load transform step is generated as a result of the

integration knowledge module (IKM). In this on-premises example, step 1 issues a SQL statement on the source that joins the GL_SET_OF_BOOKS table with the GL_PERIOD_TYPES table. The join is executed on the source database and the resulting data is staged. Then, a second join occurs at the load transform stage between the W_DOMAIN_G table and the temporary stage table, which results in the loading of the stage table W_LEDGER_DS.

Note: Oracle Database is the only database type supported for the Oracle BI Applications repository schemas and the Business Analytics Warehouse.



There are four main stages: The first, unique to cloud sources, is the Source Dependent Data Store (SDS) stage, which loads and incrementally maintains replicated data into a Source Dependent Data Store schema from cloud sources, for example Fusion Cloud, Taleo Cloud, and so on. A Health Check stage generates a diagnostic report identifying problematic source data that might cause ETL failure, or data loss, or corruption in the data warehouse. The Source Dependent Extract (SDE) tasks then extract data from either SDS schema tables in the case of cloud sources, or source dimension and fact tables in the case of on-premises sources, and load the data into universal dimension and fact staging tables. The Source Independent Load (SIL) tasks are common and load data from the universal staging tables into the warehouse staging tables. This figure depicts a dependency between the dimension table and the fact table. Therefore, the SIL DIM must be executed before the SIL FACT to resolve the dimension key. The SIL DIM has a database sequence that generates the key, and then the SIL FACT looks up that key when loading the fact staging table.



About ETL Phases

Oracle BI Applications ETL processes include these phases: SDS, Health Check, SDE, SIL, and PLP.

- **SDS** stands for Source Dependent Data Store. In this phase, a separate schema on the data warehouse database is maintained as a replication of the source transactional systems' tables, deletions, as well as additional optimizations for incremental ETL. Each SDS requires its own separate schema because there can be multiple SDS each having the same object names. Typically, you see a corresponding load plan step, SDS Load Phase, in your generated load plan when you enable extraction from cloud sources, for example when extracting data from Fusion Cloud, Taleo Cloud, and so on. SDS Load Phase tasks extract data from cloud sources and stage it in SDS tables.
- **Health Check**. Health Check is a preliminary ETL phase in which a diagnostic report is generated to identify problematic source data that might cause ETL failure or data loss or corruption in the data warehouse. The report is downloaded from Configuration Manager and includes any problematic data, a description of detected issues, and actions to resolve health check failures.
- **SDE** stands for Source Dependent Extract. In this phase, SDE tasks extract data from the source system and SDS and stage it in staging tables. The SDE tasks are source specific.
- **SIL** stands for Source Independent Load. Load tasks transform and port the data from staging tables to base fact or dimension tables. The SIL tasks are source independent.
- **PLP** stands Post Load Process. PLP tasks are only executed after the dimension and fact tables are populated. A typical usage of a PLP task is to transform data from a base fact table and load it into an aggregate table. The PLP tasks are source independent.

About the Oracle Data Integrator Repository

The Oracle Data Integrator (ODI) Repository for Oracle BI Applications comprises two repositories: master and work.

- **Master Repository.** Contains topology of resources, security, and version management. A master repository is usually associated with multiple work repositories.
- **Work Repository.** Contains data models and projects. A work repository is the development and execution repository and is attached to a single master repository.

The master and work repositories must reside in the same database schema and the database type must be Oracle. Both the master and work repositories are set up during the Oracle BI Applications installation process.

The default ODI repository ID is 500. This ID is part of the internal ID for every object that is created within the ODI repository. Having a repository ID greater than or equal to 500 is critical to ensure that the objects you create do not overlap with any current or future Oracle BI Applications objects. If you change the default ODI repository ID, make sure the new value is greater than 500.

About Load Plans

A load plan is an executable object that comprises and organizes the child objects (referred to as steps) that carry out the ETL process. A load plan is made up of a sequence of several types of steps. Each step can contain several child steps. Depending on the step type, the steps can be executed conditionally, in parallel or sequentially.

You define a load plan in Configuration Manager by selecting a data source and one or more fact groups. This selection determines which steps you should perform during the ETL process. Each fact group belongs to a specific functional area or areas that are associated with one or more offerings, which, in turn, are related to a data server. A transactional data source is associated with one or more data servers.

After you define the load plan, you then generate it to build it in the ODI repository. You then execute the load plan to perform the ETL process.

To work with Oracle BI Applications load plans, see [Managing Load Plans](#). To work with load plans in the context of Oracle Data Integrator, see Introduction to Load Plans in *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

About Changed Data Capture

Oracle BI Applications has two ETL modes for loading data into the Oracle Business Analytics Warehouse: full and incremental.

During a full load, Oracle BI Applications extracts:

- All records from tables that are sources for dimension tables.
- Records created after an Initial Extract Date from tables that are sources for fact tables. The Initial Extract Date defines a cut-off so that not all records are loaded into the data warehouse. You set the Initial Extract Date value for each data source in Configuration Manager.

An ETL process can extract a record from a single table or from multiple tables. When a record is the result of joining multiple tables, one of these tables is identified as the *base* table, which defines the granularity of the record. When extracting fact records, Oracle BI Applications only compares the Created Date of the base table to the Initial Extract Date.

During an incremental load, Oracle BI Applications extracts records that have changed or were created after a Last Extract Date. This is done by comparing the Last Extract Date value to a Last Updated Date (LUD) type column in the source table. If the source table does not have such a column, Oracle BI Applications extracts all records from that table. The Last Extract Date is a value that is calculated based on the last time data was extracted from that table less a Prune Days value. The Prune Days parameter is used to extend the window of the ETL extract beyond the last time the ETL actually ran. This is to ensure records that might have somehow been missed in a prior ETL are picked up in the next ETL. Records can be missed in an ETL process when a record is being updated while the ETL process is running and was not committed until after the ETL completed.

You set the Prune Days parameter value in Configuration Manager before the first full load ETL and the value automatically takes effect in any subsequent incremental ETL. Bear the following in mind:

- Setting a small value means the ETL extracts fewer records, thus improving performance. However, this setting increases the chances that records are not detected.
- Setting a large number is useful if ETL runs are infrequent, but this increases the number of records that are extracted and updated in the data warehouse. Therefore, you should not set the Prune Days value to a very large number. You can also use large Prune Days number to trigger re-extracting records that are processed, but have not changed.
- Setting a value to 0 means the ETL should not extract any record. Hence, you should never set the Prune Days to 0.

If you have not set the Prune Days parameter prior to running a full ETL, but want to use one for a current incremental load, you can do so using the RESET_LAST_EXTRACT_DATE scenario in the RESET_LAST_EXTRACT_DATE procedure. For the scenario, in ODI Designer, set the BIAPPS.DATASOURCE_NUM_ID parameter to the source system's ID and set the BIAPPS.PRUNE_DAYS parameter to the desired prune days. Execute the RESET_LAST_EXTRACT_DATE procedure from the ODI Console to set the Prune Days.

An ETL process can extract a record from a single table but more commonly extracts records that are the result of joining multiple tables. When extracting from multiple tables, one table is identified as the base table, which defines the granularity of the record. When there is a change in the base table, an extract is triggered for the record. However, there can be cases where a change occurs in a non-base table but not in the base table itself. If a change occurs in a non-base table and this should trigger an extract of the record, these tables are referred to as auxiliary tables. Thus, when determining if a record should be extracted, Oracle BI Applications compares not only the LUD column in the base table but the LUD columns in all auxiliary tables. If the LUD column changed in any of these tables, the record is extracted. If changes can occur in a table that is not relevant to triggering an extract, this table's LUD column is not compared in the incremental filtering logic.

About Knowledge Modules

Knowledge Modules (KMs) implement different tasks within the Oracle Business Analytics Warehouse system.

The following are the different types of KMs available:

- **Reverse-engineering (RKM).** Used for reading the table and other object metadata from source databases and to import tables, columns, and indexes into a model. See [About Reverse Knowledge Modules](#).
- **Loading (LKM).** Used for efficient extraction of data from source databases for loading into a staging area (database-specific bulk unload utilities can be used where available).
- **Integration (IKM).** Used to load data into a target with different strategies, for example, slowly changing dimensions and insert/update strategies.
- **Check (CKM).** Used to validate and cleanse data.
- **Journalizing (JKM).** Used to record the new and changed data within either a single table or view or a consistent set of tables or views.
- **Service (SKM).** Used to expose data in the form of web services.

About Reverse Knowledge Modules

Oracle BI Applications uses the ODI reverse engineering process to populate the repository with metadata from the source system. RKMs retrieve metadata from data storage and load it into the repository.

For example, RKMs detects the description of tables, columns, data types, constraints, and comments from a database to load the repository. RKMs support various technologies, such as databases, XML files, and various types of flat files. You can also use RKMs to retrieve non-standard metadata from databases or applications, such as Oracle E-Business Suite, Siebel CRM, PeopleSoft, and so on.

The RKM role is to perform customized reverse engineering for a model. The RKM handles connecting to the application or metadata provider then transforming and writing the resulting metadata into the ODI repository. The metadata is written into temporary tables, and then the RKM executes the ODI API to read from these tables and write to the ODI metadata tables of the Work repository in incremental update mode.

Note that the ODI repository of Oracle BI Applications contains the relevant source data models. Therefore, you run an RKM only if you have customized tables in the source system and want to import these changes to the ODI repository. To customize tables and tasks, see [Customizing the Oracle Business Analytics Warehouse](#).

About Multi-Source Environments

Oracle BI Applications supports the loading of data into the Oracle Business Analytics Warehouse from multiple source systems. If the same adaptor is required for two different sources, then this requires the adaptors' model and maps to be duplicated in the ODI repository.

Consider a scenario in which you have a PeopleSoft 9.0 source and an Oracle EBS 11.5.10 source, both loading the target fact table ASTXACT_FG. Loading this target table includes three serial steps:

1. Initialize Target Table (ASTXACT_FG).
2. Load Target Table (ASTXACT_FG).
3. Finalize Target Table (ASTXACT_FG).

This figure shows these load plan steps in ODI Studio.

#	Steps Hierarchy	Enab...	Scenario/Variable	Restart	Context
0	Start Load Plan	✓		Restart from failure	
1	Global Variable Refresh	✓		Restart from failure	
2	1 Initialize Batch Load	✓		Restart from failure	
4	Source Extract Phase	✓		Restart from failure	
5	1 Generate	✓		Restart from failure	
218	1 SDE Extract	✓		Restart from failure	
219	2 SDE Fact Group	✓		Restart from failure	
220	Parallel(Persisted/Temporary Staging)	✓		Restart from failed children	
230	Parallel (Fact Groups)	✓		Restart from failed children	
231	3 SDE Fact ASTXACT_FG	✓		Restart from failure	
232	Initialize ASTXACT_FG	✓		Restart from failure	
233	EXEC_TABLE_MAINT_PROC	✓	EXEC_TABLE_MAINT_PROC V...	Restart from failed step	
234	Load Target Table	✓		Restart from failed children	
235	EBS_11_S_10 - DSN 310	✓		Restart from failure	
241	PSFT_9_0 - DSN 517	✓		Restart from failure	
246	Finalize ASTXACT_FG	✓		Restart from failure	
247	EXEC_TABLE_MAINT_PROC	✓	EXEC_TABLE_MAINT_PROC V...	Restart from failed step	
248	2 SDE Dimension Group	✓		Restart from failure	
629	Warehouse Load Phase	✓		Restart from failure	
817	1 Finalize Batch Load	✓		Restart from failure	

Data from both sources, PeopleSoft 9.0 and Oracle EBS 11.5.10, is being loaded into the target table (ASTXACT_FG) for the first time, which means a full load is occurred; and, generally, when a full load occurs, the target table is truncated. In this case, the load plans can run in parallel because the data sources are not dependent on each other; however, the load from the second source should not truncate the target table if it already contains data from the first source. The following serial actions are performed in the process:

1. The Initialize Target Table step truncates the target table.
2. The Load Target Table step, in turn, has two child parallel steps. Each of these loads the target from each source system.
3. The Finalize Target Table step creates any indexes and analyzes the table.

The generated load plan ensures that you can have separate load plans for each source, but load plans should not run in parallel. Hence, the table is truncated only at the required time (in this case, only once before any source system loads it).

Note:

Configuration Manager and ODI do not stop load plans from running in parallel; however, it is recommended that you do not do so because of the following reasons:

- All load plans truncate target tables upon the initial load. If you run load plans in parallel, one load plan can truncate the data loaded by the preceding load plan.
- The mappings from SILOS onwards are common and not based on the source system. If load plans are run in parallel, you can have a situation in which only partial data is loaded from the second source due to the data that was loaded from the first source. To resolve this issue, make sure that the first load plan completes successfully before the second load plan runs.

In other words, when you load a fact table, bear in mind that the fact table might be connecting with multiple dimensions and lookup tables to get the final data. When load plans are running in parallel, some of the dimension, lookup, and staging tables can also have data from a second source. This might lead to some lookups and dimensions not returning appropriate value for the second source, since they have not yet been completely loaded.

When extracting data from different source types (Siebel and Peoplesoft for example) or different versions of the same source type (eBusiness Suite 12.0 and eBusiness Suite 12.1 for example), no additional steps are required. However, if extracting data from two or more instances of the same source version (such as two instances of eBusiness Suite 12.0), this is specifically referred to as 'Multi-Instance Support.' Multi-Instance requires additional steps to support. Refer to My Oracle Support document 1633996.1 —OBIA 11g: Multi-Instance Source Support in BI Applications 11.1.1.x.1, for more details.

About ETL Roles

Access to Configuration Manager and Functional Setup Manager (FSM) is controlled through these duty roles.

Oracle BI Applications has two duty roles for ETL operations:

- Load Plan Operator Duty
- Load Plan Administrator Duty

Oracle BI Applications has the following additional duty roles:

- BI Applications Administrator Duty
- BI Applications Functional Developer Duty
- BI Applications Implementation Manager Duty

The security administrator must grant the appropriate duty roles to a user based on the user's job responsibilities. For information on the Configuration Manager and FSM screens that each duty role has access to, see Duty Roles for Access to Functional Setup

Manager or Configuration Manager in *Oracle Business Intelligence Applications Security Guide*.

The BI Applications administrator, load plan operator, and load plan administrator users require appropriate access to ODI. In addition to these users being created in the LDAP system, these users must also be created in the ODI Repository, and they must be granted the Supervisor profile or an appropriate ODI profile. The BI Applications administrator must be granted the Supervisor role in ODI. Work with your security administrator to obtain the appropriate duty roles.

See Managing Security in Oracle Data Integrator in *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* .

About Cloud Sources

Performing ETL from Fusion Cloud sources is a collaborative effort. Data extracted from cloud-hosted source application database tables using standard SDE knowledge module logic is uploaded automatically to Oracle Universal Content management for uptake by on-premises ETL developers and operators.

At appropriate intervals, new or incremental datasets required to create and maintain the on-premises data warehouse are updated in an Oracle Source Dependent Schema (SDS), which serves as the on-premises copy of the cloud data for the purposes of aggregating, transforming, and loading it into the on-premises data warehouse. Once data is available in the cloud SDS database schema, you can create and run load plans to load the warehouse tables in Configuration Manager.

In the case of Taleo and RightNow data sources, data is sourced automatically directly from the respective cloud transactional tables and loaded into the data warehouse staging tables during SIL processing.

Managing Load Plans

The tasks for managing load plans are usually performed by either Extract, Transform, and Load (ETL) developers or ETL operators using Oracle BI Applications Configuration Manager.

ETL Developer Tasks

- [Defining Load Plans](#)
- [Duplicating Load Plans](#)
- [Editing Load Plans](#)
- [Generating Load Plans](#)
- [Scheduling Load Plans](#)
- [Executing Load Plans](#)

ETL Operator Tasks

- [About Diagnostics Health Check and ETL Diagnostics and Automatic Correction](#)
- [Downloading Diagnostic Health Check Reports](#)
- [Downloading Error Data from Diagnostics](#)
- [Monitoring Load Plan Runs](#)
- [Restarting Load Plans](#)
- [Stopping Load Plans](#)

Overview of Load Plan Life Cycle

A load plan life cycle comprises four phases.

- Phase 1: Define load plan
In this phase, you define load plan properties in Configuration Manager, including selecting a data source and one or more fact groups. This selection determines which steps you must perform during the ETL process.
- Phase 2: Generate load plan
In this phase, you launch a generation process from Configuration Manager that propagates the load plan properties to the Oracle Data Integrator (ODI) Repository, where the load plan is built.
- Phase 3: Execute load plan

In this phase, you start a load plan run from Configuration Manager, which executes the steps of the load plan. Executing a load plan creates a load plan instance and a first load plan run. If a run is restarted, a new load plan run is created under this load plan instance. Each execution attempt of the load plan instance is preserved as a different load plan run in the log.

- Phase 4: Monitor load plan

In this phase, you monitor the load plan run in the Load Plan Details page of Configuration Manager. The Load Plan Details page provides a view of the ODI Repository through ODI Console.

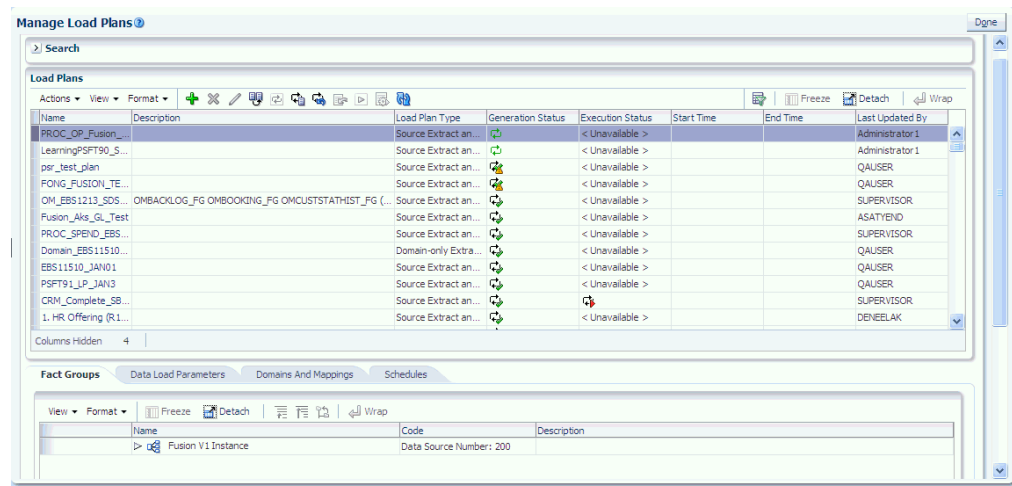
Defining Load Plans

Define load plan properties in Configuration Manager, including selecting a data source and one or more fact groups. This selection determines which steps you must perform during the ETL process.

To define a load plan:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.



2. In the Load Plans toolbar, click the **Add** icon.

The Define Load Plan page is displayed.

3. On the first page of the Define Load Plan series, specify the following information:

Field	Description
Name	Enter a unique name for the load plan.
Description	(Optional) Enter additional information about the load plan.

Field	Description
Load Plan Type	<p>Select a load plan type. The possible values are the following:</p> <ul style="list-style-type: none"> • Source Extract (SDE) - Includes only those tasks that extract from the source and load data into staging tables. • Source Extract and Load (SDE, SIL and PLP) - Includes all tasks to extract from the source and load the data warehouse tables. • Warehouse Load (SIL and PLP) - Includes only those tasks that extract from the staging tables and load the data warehouse tables. <p>Note that it might be useful to generate separate source-specific and data warehouse-specific load plans. By decoupling the load plans, this allows scheduling a source-specific load plan to run during hours when most users are not using the source system and scheduling a separate load of the data warehouse when most users are not using the data warehouse.</p> <ul style="list-style-type: none"> • Domain-only Extract and Load (SDE and SIL) - Includes all tasks required to extract domain-related records from the source and load the data into the domain-related tables in the Oracle Business Analytics Warehouse. Note that domains are used extensively in Configuration Manager and you must configure properties before executing a regular load plan. These properties depend on the domain values found in the transactional database.
Source Instances	Select the data sources from which the fact groups are selected.

4. Click **Next**.

The second page of the Define Load Plan series is displayed.

5. In the Available Selections tab, select the fact groups you want to include in the load plan definition.

Note: Fact groups might belong to a hierarchy of fact groups. You can select only the top level parent fact group and not a child fact group.

A load plan must contain at least one fact group; multiple fact groups might be selected from one or more data sources.

6. Click **Save**.

- Click **Save** to save the load plan. After a load plan is saved, it is displayed in the Load Plans master list.
- Click **Save and Generate Load Plan** to save the load plan and immediately generate it.

Duplicating Load Plans

You can duplicate an existing load plan from the Manage Load Plans page of Configuration Manager. A duplicated load plan is displayed in the Load Plans master list immediately. You can generate the duplicated load plan instantly.

To duplicate an existing load plan:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan you want to duplicate.
3. In the Load Plans toolbar, click the **Duplicate** icon.

The Duplicate Load Plan page is displayed.

4. On the first page of the Duplicate Load Plan series, specify the following information:

Field	Description
Name	Enter a unique name for the load plan.
Description	(Optional) Enter additional information about the load plan.
Load Plan Type	(Optional) Select a load plan type. The possible values are the following: <ul style="list-style-type: none">• Source Extract (SDE)• Source Extract and Load (SDE, SIL and PLP)• Warehouse Load (SIL and PLP)• Domain-only Extract and Load (SDE and SIL)
Source Instances	(Optional) Select the data sources from which the fact groups are selected.

5. Click **Next**.

The second page of the Duplicate Load Plan series is displayed.

6. In the Available Selections tab, select the fact groups you want to include in the load plan definition.

Note: Fact groups might belong to a hierarchy of fact groups. You can select only the top level parent fact group and not a child fact group.

A load plan must contain at least one fact group, and multiple fact groups might be selected from one or more data sources.

7. Click **Save**.

A submenu is displayed with the following options:

- Click **Save** to save the load plan. After a load plan is saved, it is displayed in the Load Plans master list.
- Click **Save and Generate Load Plan** to save the load plan and immediately generate it.

Editing Load Plans

You can edit details of an existing load plan, such as load plan name, load plan type, and the data sources from the Manage Load Plans page of Configuration Manager. An

edited load plan is displayed in the Load Plans master list immediately. You can generate the edited load plan instantly.

To edit an existing load plan:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan you want to edit.
3. In the Load Plans toolbar, click the **Edit** icon.

The Edit Load Plan page is displayed.

4. You can edit these properties:

Field	Description
Name	Enter a unique name for the load plan.
Description	(Optional) Enter additional information about the load plan.
Load Plan Type	(Optional) Select a load plan type. The possible values are the following: <ul style="list-style-type: none"> • Source Extract (SDE) • Source Extract and Load (SDE, SIL and PLP) • Warehouse Load (SIL and PLP) • Domain-only Extract and Load (SDE and SIL) You can't change the Load Plan Type of an existing Load Plan.
Source Instances	(Optional) Select the data sources from which the fact groups are selected.

5. Click **Next**.

The second page of the Edit Load Plan series is displayed.

6. In the Available Selections tab, select the fact groups you want to include in the load plan definition.

Note: Fact groups might belong to a hierarchy of fact groups. You can select only the top level parent fact group and not a child fact group.

A load plan must contain at least one fact group, and multiple fact groups might be selected from one or more data sources.

7. Click **Save**.

A submenu is displayed with the following options.

- Click **Save** to save the load plan. After a load plan is saved, it is displayed in the Load Plans master list.

- Click **Save and Generate Load Plan** to save the load plan and immediately generate it.

Generating Load Plans

When you generate a load plan, the load plan is built in the ODI Repository. A load plan must be successfully generated before it can be executed.

Load plans must be generated serially, otherwise the process fails. Do not launch a second load plan generation if one is already underway. You must wait until the first generation process completes before you launch the next generation process.





To generate load plans:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan you want to generate.
3. In the Load Plans toolbar, click the **Generate** icon.

The possible icons that are displayed in the Generation Status column of the Load Plan master list are described in the following table. Click the **Refresh** icon to refresh the display.

Generation Status Icon	Description
	Starting
	In Progress
	Succeeded
	Failed

You can execute a load plan or schedule it for execution after it has been successfully generated.

Scheduling Load Plans

You can schedule a load plan by defining its ODI context, ODI local agent, a level of logging information, status of the schedule, and frequency of occurrence in Configuration Manager.

To schedule a load plan for execution:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.

2. In the Load Plans list, select the load plan you want to schedule.
3. Select the Schedules tab.
4. Click **Add** in the Schedules tab toolbar.

The Schedule Load Plan dialog is displayed.

5. Select:

Field	Description
Context	The ODI context to be used when the load plan is run. Note that Global is the only supported context.
Local Agent	The ODI local agent to be used when the load plan is run.
Log Level	The level of logging information to retain. Configuration Manager uses Oracle Diagnostic Logging. See Managing Log Files and Diagnostic Data in <i>Oracle Fusion Middleware Administrator's Guide</i> .
Status	Status of the schedule. The possible values are the following <ul style="list-style-type: none"> • Active • Inactive • Active for the period
Recurrence	Frequency of occurrence. The possible values are the following: <ul style="list-style-type: none"> • On Agent Startup • Simple • Hourly • Daily • Weekly • Monthly (day of the month) • Monthly (week day) • Yearly Depending on the Recurrence option you select, options for selecting the date and time are dynamically displayed.

6. Click **Schedule**.

Note: If the ODI Agent is already running at the time you define the schedule in Oracle BI Applications Configuration Manager, then you must click the Update Schedules icon in the toolbar to add the schedule to the ODI Agent scheduler.

Executing Load Plans

You can execute a load plan after you generate and schedule it.

Bear these guidelines in mind while executing a load plan:

- You can have separate load plans for each source, but load plans should not run in parallel.

- You can only execute a load plan if it was successfully generated. See [Generating Load Plans](#).

To execute a load plan:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.

2. In the Load Plans list, select the load plan you want to execute.
3. In the Load Plans toolbar, click the **Execute** icon.






The Execute Load Plan dialog is displayed.

4. Specify the following information:

Field	Description
Context	The ODI context to be used when the load plan is run. Note that Global is the only supported context.
Logical Agent	The ODI logical agent to be used when the load plan is run.
Oracle Data Integrator Work Repository	The name of the ODI Work repository.

5. Click **OK**.

The following icons show the possible execution statuses that are displayed in the Execution Status column in the Load Plan master list. Click the **Refresh** icon to refresh the display.

Generation Status Icon	Description
	Starting
	In Progress
	Succeeded
	Failed
	Not Executed

6. (Optional) Click **Show Execution Status Details** to display detailed information about the status of the execution.

About Diagnostics Health Check and ETL Diagnostics and Automatic Correction

Diagnostic Health Check is a preliminary ETL phase in which a diagnostic report is generated to identify problematic source data that might cause ETL failure or data loss or corruption in the data warehouse. In the event of a failure of an ETL task, diagnostics are run and error handling and automatic correction are performed to enable the load plan to restart and continue.

During ETL, when you run a load plan, the relevant source data is queried to generate a health check report. The health check queries are seed data XML files and can be found at `{bi.oracle.home}/biapps/etl/data_files/src_files/BIA_11/metadata/healthcheck.xml`. You can download a health check report from Configuration Manager and the health check report includes any problematic data, a description of detected issues, and actions to resolve health check failures.

Email Notifications

If an email address is configured for ETL email notification during Functional Setup Manager configuration, an email is sent in the following cases:

- If the health check scenario fails, an email is sent with a summary of the health check results.
- If the load plan execution fails, an email is sent with a list of failed tasks during that load plan execution.
- If the load plan execution completes, an email is sent with a list of tasks that are in failed state in any of the load plan instance runs and a summary of auto-corrections performed (if any), during the ETL.

Automatic Correction of Data Errors

Some common data errors that can cause ETL tasks to fail are automatically diagnosed and corrected during the ETL, allowing for the load plan to restart and complete.

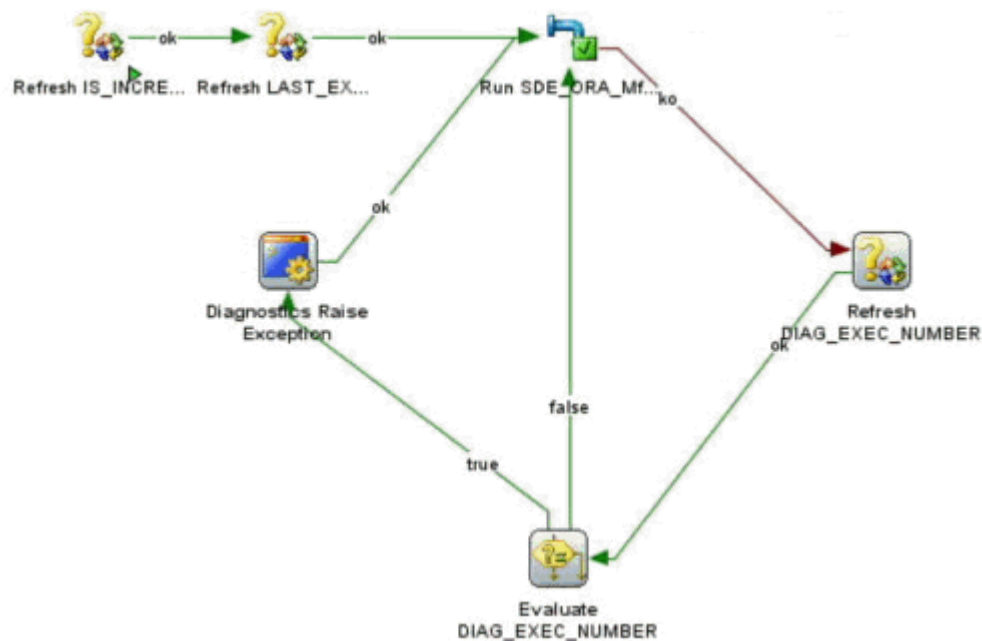
Data Error Type	Automatic Correction
Duplicate rows being inserted into the target table causes a unique constraint violation.	One row is inserted and the duplicate rows are rejected.
String values larger than allowed being inserted into <code>varchar</code> or <code>char</code> columns.	All <code>varchar</code> columns are truncated to the precision allowed by each column before insert. Exception: key columns defined in the interface are not truncated.
All other issues, including: <ul style="list-style-type: none"> • Number values larger than allowed being inserted into numeric columns. • NULL values being inserted into a NOT NULL column. • Invalid data, for example alphanumeric values being inserted into a number column, invalid date strings, and so on. 	Row is rejected. For dimension load tasks, instead of rejecting rows, the row is inserted into the target table with the unspecified value for all the columns except the key columns (typically <code>INTEGRATION_ID</code> and <code>DATASOURCE_NUM_ID</code>). This is done to ensure that fact rows inserted during ETL which refer to a dimension row, don't have a dangling foreign key.

The following Knowledge Modules have been enhanced to support diagnostic features during load plan execution process:

- IKM BIAPPS Oracle Control Append
- IKM BIAPPS Oracle Incremental Update
- IKM BIAPPS Oracle Fact Incremental Update
- IKM BIAPPS Oracle Slowly Changing Dimension
- IKM BIAPPS Oracle Event Queue Delete Append
- IKM BIAPPS Oracle Fact Event Queue Delete Append
- IKM BIAPPS CLOUD Oracle Incremental Update
- CKM BIAPPS Oracle

Package Structure for Error Handling and Automatic Correction

All ODI packages that have a main interface step using one of the diagnostic-enabled IKMs include a loop as shown in the example here. Each interface step in the package that uses one of the IKMs has a *ko* flow. At a high level, if there is no failure in the interface step, the scenario execution completes successfully after executing the interface step. However, if there is a failure in the interface step, the control is transferred to the *ko* flow, which executes Refresh DIAG_EXEC_NUMBER and Evaluate DIAG_EXEC_NUMBER steps and then re-runs the interface step in diagnostic mode. The interface step now tries to capture Data Manipulation Language (DML) data errors (if any) in an error table and tries to auto-correct them based on a set of predefined rules as described, for example trimming string lengths in case of a string length overflow error.



If the diagnostic framework can't automatically correct the error records, they are rejected in an error table so that the non-error rows can be loaded in the target and the scenario execution can complete successfully.

If the failure was caused by a system error or error in LKM steps, the interface step is more likely to fail again, in which case the control is transferred to Diagnostic Raise Exception procedure step and fail the scenario execution.

Enabling and Disabling Diagnostic Features

ETL diagnostics and automatic correction can be turned on or off using two parameters in Configuration Manager, both of which are set to Y by default:

- **DIAG_ERR_LOG_SUPPORTED** — This parameter controls whether the ODI IKM code performs any auto-corrections for DML data errors or not. If disabled, ODI IKM does not attempt to perform any automatic corrections on data. Upon failure of a task, it still re-runs the interface step in diagnostic mode, but raises an error after attempting to capture the data errors.
- **DIAG_AUTOCORRECT** — This parameter controls the ETL diagnostic and auto-correction feature. When disabled, failed interface steps are not re-run in diagnostic mode.

To reset these parameters, navigate to the Manage Data Loads Parameters page in Configuration Manager and set them to N. Your load plans are regenerated after resetting these parameters.

Downloading Diagnostic Health Check Reports

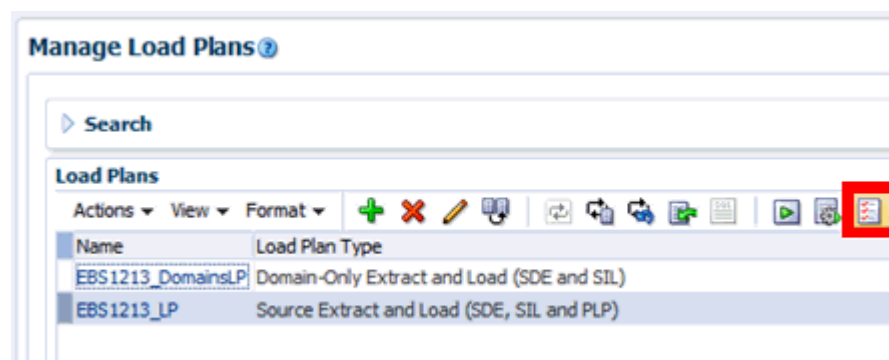
The health check report includes any problematic data, a description of issues, and actions to resolve health check failures. You can generate a health check report of your load plan by querying the relevant source data in Configuration Manager.

To download a diagnostic health check report:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan that has failed and whose report you want to download.
3. Click **Show Data Problems**.



4. In the Load Plan Data Problems window, select the row showing the task name as **DIAGNOSTIC HEALTHCHECK** and click **Download**.

5. In your browser, download the zip file containing the report, then unzip the file on your system.

Downloading Error Data from Diagnostics

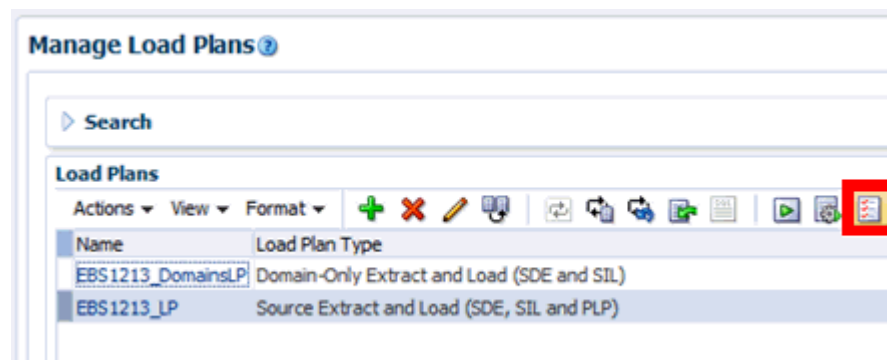
You can download error data from automatic ETL diagnostics in a zip file. In this zip file, you see some system columns to describe the error number, error message, and automatic correction done by the ETL program.

To download error data:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan that has failed and whose report you want to download.
3. Click **Show Data Problems**.



4. In the list of tasks that encountered data problems and were corrected, the task name and the target table this task is inserting into or updating are noted. These are the rows where the Component column has a value of ETL. Select each of these rows and click **Download**.
5. In your browser, download the zip file containing the report, then unzip the file on your system.

Each line in the file is an entire row of data for the target table with all of the columns. Along with the columns of the target table, you see some system columns in the file to describe the error number, error message, other relevant information in understanding the problem with each row of data, and the automatic correction done by the ETL program.

Monitoring Load Plan Runs

You can monitor a load plan run by viewing the execution status information on the Load Plan Execution Details page of Configuration Manager.

To monitor a load plan:

1. In the Tasks pane of Configuration Manager, select **Manage Load Plans** under **Load Plans Administration**.

The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan whose execution details you want to view.
3. In the Load Plans toolbar, click **Show Execution Status Details**.

The ODI Console is displayed.

4. Log into the ODI Console by entering an appropriate **User ID** and **Password**.

The Load Plan Details page is displayed. For a description of the information displayed on this page, see Load Plan Details Page.

Restarting Load Plans

When you run ETL to load data from a source system into the Oracle Business Analytics Warehouse, you might need to restart the ETL load after a failure. This section explains the options available for restart and describes the implications of using each of those options.

Examples of circumstances and reasons for load plan failure include the following:

- Issues identified in the health check ETL phase. Fix these issues before restarting the load plan.
- Problem with access either to source or target database due to network failure or expired or otherwise incorrect user names and passwords.
- Failure of ODI agent.
- Problem with space or storage. You can connect to the source or target database, but the query fails to run due to lack of temp space, disk space, and so on. It might be due to inadequate space where files need to be placed.
- Problem with data, for example incorrect data with lengths larger than the target column can hold, or null values in Not Null columns.

After such a failure during ETL, to avoid restarting the entire load plan after a failure, which requires inefficient re-runs of all ETL tasks, you must restart the load from the same point in its execution once the cause of failure has been diagnosed and resolved. Some of these circumstances for failure are automatically diagnosed and resolved by ETL diagnostics and auto-correction. See [About Diagnostics Health Check and ETL Diagnostics and Automatic](#).

About Restartability Grain

When you restart a load plan after a failure, you might not restart again from the exact point of failure, depending on where it occurred and on dependencies between load plan steps. The point of restartability is that the end result of the load plan execution is the same regardless of any load plan failure.

The following example describes one such dependency-driven requirement for re-running a step that has already completed: In a load plan with two steps, the first step truncates the table, and the second inserts records into the table, intermittently committing the records. The load plan is run and fails at the second step due to a space issue. After the issue is resolved, restarting the load plan from the second step is incorrect, because the target has some inserted rows. Restart should instead begin with the first step so that the target table is truncated again and newly inserted data does not cause any duplicates.

To maintain data integrity in the case of restart, the grain varies depending on the location in the step hierarchy of the failed step and on the Restart setting for the step in the Load Plan Manager.

Within the Steps Hierarchy in Load Plan Manager, you can view and edit the Restart setting of a step in the Restart column. The default settings for different steps in the hierarchy support data integrity in restarts:

- Root steps are set to Restart From Failure if Serial and Restart from failed Children if Parallel.
- Sub steps are set to Restart From Failure if Serial and Restart from failed Children if Parallel.
- Scenario steps are set to Restart from Failed Step.

The following examples highlight the implications for each type of load plan step:

Serial Load Plan Step

Serial steps are represented by a horizontal icon in the Steps hierarchy in Load Plan Manager, and by default have a Restart setting of Restart from Failure. In a case where the load plan fails when running such a step to load a Dimension Group with multiple serial sub-steps loading individual dimensions, the load plan on restart, starts from the individual sub-step that failed. Any successfully completed serial sub-steps are run again.

Parallel Load Plan Step

Parallel steps are represented by a vertical icon in the Steps hierarchy in Load Plan Manager and by default have a Restart setting of Restart from Failed Children. In a typical run, a parallel step with five parallel sub-steps under it has all five sub-steps executed in parallel, subject to free sessions being available. If two of those five steps completed and then the load plan is to fail, when the load plan is restarted, all the steps that did not complete or failed are started again.

Scenario Step

At the lowest order in any load plan are the scenario steps. While the parent steps, whether serial or parallel, are used to set the dependencies, the scenario steps are those, which load the tables. A scenario step, in turn, can have one or more sub-steps, corresponding to the number of steps inside the package.

In the case of a scenario step failure during execution, consider that the scenario step might have multiple steps, all under the same session in the Operator log but identified with different step numbers: 0, 1, 2, and so on. In the case of restart, the scenario executes from the failed parent scenario step, re-running all sub-steps.

Note: If you use the Load Plan Generator to generate a load plan, it automatically conforms to the standard. If you are manually altering a generated load plan or creating a new load plan without using Load Plan Generator, then you should ensure that you conform to the standard.

Restarting Load Plans Using ODI Studio

Use ODI Studio or ODI Console to restart a load plan.

Follow this procedure to restart a load plan using ODI Studio. The restart option is enabled only on the last run for a load plan. You can start a load plan any number of times and each time it progresses from the last failure.

Note: When you restart the load plan, set the Log Level to 6. Log Level 6 enables you to see variable values resolved in session logs.

To restart a load plan using ODI Studio:

1. In ODI Operator, navigate to the Operator log, and select the last failed run for a load plan.
2. Double-click the load plan run, and select the **Restart** option.

You can also right-click the last run in the Operator log and select **Restart**.

Restarting Load Plans Using ODI Console

Follow this procedure to restart a load plan using ODI Console. The restart option is enabled only on the last run for a load plan. You can start a load plan any number of times and each time it progresses from the last failure.

Note: When you restart the load plan, set the Log Level to 6. Log Level 6 enables you to see variable values resolved in session logs.

To restart a load plan using ODI Console:

1. In ODI Console, go to **Runtime**, select **Sessions/Load Plan Executions**, and then select the load plan execution that has failed.
2. Click **Restart**.

The **Restart** button is displayed only when the selected load plan is the most recent run of the load plan.

Troubleshooting Load Plans

A load plan must be restarted when it has stopped with an error. An alternate case where restart might be required is when a load plan is not doing anything at all, for example, when a load plan is executed and nothing has changed after 30 minutes.

To troubleshoot a non-responsive load plan:

1. Check the maximum number of sessions set to run against the agent.

In ODI Operator, verify that the number of sessions running is equal to the maximum. If so, then the other sessions are waiting for the running sessions to complete. Proceed to the next step.

2. Clean out stale sessions.

The stale sessions are sessions that are incorrectly left in a running state after an agent or repository crash. If an agent crashes or loses its connection to the repository after it has started a session, the agent can't update the status of the session in the repository, and such a session becomes stale. Until the stale session is cleaned, it displays as running in the repository, but actually is not.

The stale sessions are cleaned in multiple ways. The following are some examples:

- You can manually request specific agents to clean stale sessions in Operator Navigator or Topology Navigator.
- Stale sessions are cleaned when you restart an agent.
- When an agent starts any new session, it checks for and resolves stale sessions. However, if the agent has lost connection to the repository, then it cannot clean stale sessions.

3. Check if the agent is alive.

To test the agent to see if it is running and still has a connection to the repository, open it in the Topology Navigator in ODI Studio and select the Test tab. If the agent test fails, then restart the agent after fixing the issue.

4. Verify that the ODI Repository and the server hosting it are running and have not experienced a failure.
5. If your load plan is in error and you have verified all of the above, then restart the load plan.

Note: When you restart the load plan, set the Log Level to 6. Log Level 6 enables you to see variable values resolved in session logs.

Alternate Options for Restarting Load Plans

You can restart failed load plans using these alternate methods.

Using Mark as Complete

In most cases the load plan restart method described earlier in this section is the recommended approach. This approach ensures data integrity and leaves no scope for manual error. However, at times you might want to run a load plan step manually. For example, if a step is inserting duplicate records which are causing failure, rerunning the step still inserts duplicates. In such a case, you might need to manually correct the data outside of the load plan and then skip that step when you restart the load plan. For this kind of situation, you can use the **Mark as Complete** option.

When you mark a load plan step as complete, it ensures that when the load plan is restarted, the marked step is not executed. It is then the responsibility of the person making this setting to ensure that the load for that step is carried out outside the load plan.

To mark a step as complete, right-click the step and select **Mark As Complete**. You can do this at the scenario step or at any step higher than that.

Marking a step complete at a higher level in the step hierarchy means that none of the child steps under that parent step are executed upon load plan restart, even if they are otherwise eligible. For this reason, you should treat marking a step as complete as an advanced task and must mark a step complete with a full understanding of its impact. There is no single recommendation that pertains in all cases. Hence, you should make the setting carefully and on a case-by-case basis.

Running a Scenario Standalone

When you are monitoring a load plan, you might not completely know how to fix a scenario step failure, but you might want to use the **Mark as Complete** option for the

failed scenario step instead of waiting for complete resolution. This prevents a step failure from precluding an entire load plan completing, while allowing you to inform the ETL team about the failed scenario step and work on a resolution. The ETL team might then fix the scenario and want to run it standalone outside the load plan to complete the load.

As in marking a step as complete, you should treat running a scenario standalone as an advanced task and you must bear these guidelines in mind:

- A scenario run outside of a load plan by itself invokes the Table Maintenance process. This can happen, depending on the setting, truncate the table before the load.

To understand this, consider that when a scenario is run inside a load plan table maintenance tasks are carried out as explicit steps (the parent step name can be either Initialize or Finalize). The scenario by itself does not invoke the Table Maintenance process when run from within the load plan. Rather, this is controlled by the EXECUTION_ID variable, which is set to the load plan instance ID. If this variable has a value greater than 0 when a scenario is run, the Table Maintenance process is not invoked. This is the case when a scenario is run from within a load plan with an instance ID. However, if this variable does not have a value greater than 0, then the scenario invokes the Table Maintenance process. This is the case when a scenario is run outside the load plan. If you set a value for the EXECUTION_ID when invoking the scenario from outside a load plan, the table maintenance steps are not executed.

- A scenario step can have many variable values set, either dynamically in the case of a refresh variable or explicitly by overriding its value at that scenario step in the load plan. When running a scenario outside the load plan, all the scenario variables must have only their default values. For this reason, you should set the variables appropriately before executing a scenario from outside the load plan. You can check the variable values that are present in the load plan by looking at the Operator log, provided the log level was set to 6 when the load plan ran. Configuration Manager uses Oracle Diagnostic Logging. See *Managing Log Files and Diagnostic Data* in *Oracle Fusion Middleware Administrator's Guide*.

Related Features and Considerations

These features of Oracle Business Intelligence Applications are related to restartability and describe some other related considerations.

Using CKM to Filter Erroneous Data

If a scenario step is failing due to bad source data, you can enable the CKM option to load the valid records and route the error records to a separate table. Examples of situations where this might be appropriate are the load of null values when they should have a value or data lengths longer than allowed target column lengths. Once the load completes, you can correct the erroneous data and have it automatically picked up in a subsequent load.

Note: Use of CKM can slow the load considerably because every record and column might be checked before loading. For this reason, this is not an option that you want to turn on across the entire load plan.

Note: Error handling and logging of erroneous data is now automatically handled by the ETL process. See [About Diagnostics Health Check](#).

Regenerating a Scenario During a Load Plan Execution

Consider a case where a load plan is started and fails at a scenario step. You fix the issue and regenerate the scenario, then restart the load plan and might expect it to pick the new scenario, but this is not what happens. If a load plan has been started and a scenario regenerated, the regenerated scenario code is not picked up when the load plan is restarted. To force the regenerated scenario to be picked up, you have two options:

- Start a new load plan run, accepting the overhead associated with restarting the load from the beginning.
- Run the regenerated scenario as stand-alone outside the load plan, marking that scenario step as complete in the load plan before restarting the load plan.

See [Alternate Options for Restarting Load Plans](#) for implications of using these options.

Restarting Long Running Jobs

Consider a case where you have a scenario that takes two hours to run. The scenario fails at the insert new rows step, after loading the C\$ and I\$ steps. On restart, the scenario attempts to reload the C\$ again. You instead want it to restart from the insert new rows steps only. This use is not supported. The restartability mechanism has been put in place in such a way that restarting a load plan is all you need to do. You do not need to clean up any data in between load plan executions, because the data is committed to the target table only after all the Knowledge Module steps are successful. If the load fails before complete success, no data is committed to that specific target table as part of the failed session.

Note: New C\$ and I\$ tables are created on restart and hence data to these tables is committed in between load plan start and restart.

Stale C\$ and I\$ Tables

On restart, new C\$ and I\$ tables are created, but since the previous load did not complete, these tables from the previous session are not dropped. The load plan generated using Load Plan Generator has a step at the end of the load plan executed Clean Stale Work Tables which takes a variable called ETL_DTOP_STG_OLDER_THAN_DAYS whose default value is 30 days. When this step runs, it drops any C\$ and I\$ tables that are older than the specified variable value.

Note: The C\$ and I\$ tables are useful tables when you want to debug a failed scenario. It might not be advisable to set the ETL_DTOP_STG_OLDER_THAN_DAYS value as too small—for example, one day—as you might lose valuable information for debugging.

Stopping Load Plans

You can stop a load plan run from by accessing the Load Plan Execution page in Configuration Manager (click **Show Execution Status Details** in the toolbar) or from ODI Studio.

To stop a load plan:

1. In Operator Navigator, select the running or waiting load plan run to stop from the Load Plan Executions accordion.
2. Right-click, and select **Stop Normal** or **Stop Immediate**.
 - **Stop Normal** — In normal stop mode, the agent in charge of stopping the load plan sends a Stop Normal signal to each agent running a session for this load plan. Each agent waits for the completion of the current task of the session and then end the session in error. Exception steps are not executed by the load plan and once all exceptions are finished the load plan is moved to an error state.
 - **Stop Immediate** — In immediate stop mode, the agent in charge of stopping the load plan sends a Stop Immediate signal to each agent running a session for this load plan. Each agent immediately ends the session in error and does not wait for the completion of the current task of the session. Exception steps are not executed by the load plan and once all exceptions are finished the load plan is moved to an error state.
3. In the Stop Load Plan dialog, select an agent to stop the load plan.
4. Click **OK**.

To stop load plan runs from ODI Studio, see *Stopping a Load Plan from ODI Studio* in *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* .

Additional Administration Tasks

The Extract, Transform, and Load (ETL) administrators perform a variety of diagnostic, configuration, and deployment tasks.

Topics

- [Additional Administration Tasks](#)
- [Using a Database Link for ETL](#)
- [Fixing Event Queue Knowledge Module](#)

Additional Administration Tasks

Here are some administration topics related to Oracle Business Intelligence Applications ETL processes.

Generating DDL Warehouse Scripts

The Business Analytics Warehouse tables are automatically deployed during the installation process when the Business Analytics Applications Suite Repository Creation Utility (RCU) executes a shipped Data Definition Language (DDL) script. The RCU does not prompt for which tablespace to assign to the individual tables and related indexes nor does it provide a mechanism for you to alter the shipped DDL. To introduce changes to the Business Analytics Warehouse data model, you use Oracle Data Integrator (ODI) to generate a new DDL script.

To generate DDL scripts and assign tablespaces to tables and indexes, see *Generating DDL and Assigning Tablespaces to Tables and Indexes* in *Oracle Business Intelligence Applications Installation Guide*.

Logging and Diagnostics

Oracle BI Applications Configuration Manager uses Oracle Diagnostic Logging. See *Managing Log Files and Diagnostic Data* in *Oracle Fusion Middleware Administrator's Guide*.

Creating ETL Tables for Siebel Sources

If your source system is Siebel and you are deploying one of the following offerings, you must create S_ETL tables in the Siebel database:

- Oracle Marketing Analytics
- Oracle Price Analytics
- Oracle Sales Analytics

- Oracle Service Analytics

To create the S_ETL tables, you generate a DDL script, which then must be provided to the Siebel database administrator, who must create the necessary tables in the Siebel database.

Note: The database user used for the ETL connection requires all Data Manipulation Language (DML) privileges (SELECT, INSERT, DELETE, UPDATES) and DDL (TRUNCATE) privileges to S_ETL tables.

To create the S_ETL tables:

1. Launch ODI Studio, and display the Designer navigator.
2. In the Projects editor, expand the following folders: BIApps Project, Components, Sources, Siebel, Oracle, Generate Siebel DDL, Packages, Generate Siebel DDL, and Scenarios.
3. Right-click **GENERATE_SIEBEL_DDL Version 001**, and then click **Execute**.
4. Select the **Context**, **ODI Agent**, and **Log Level**, and then click **OK**.
5. Indicate the information for each option.

Option	Description
CREATE_SCRIPT_FILE	Select Latest Value and enter Y in the Value field.
REFRESH_MODE	Enter FULL.
CHAR_CLAUSE	Provided for Unicode support. If set to Y, the CHAR clause is included in the DDL.
RUN_DDL	Enter N.
SCRIPT_LOCATION	Enter the path where you want the script to be created. The file name is similar to BIA_Siebel_Schema_DDL_<run ID>.sql.
TABLE_MASK	The default value % compares all tables.

Using a Database Link for ETL

Set up direct links to replace the default Open Database Connectivity (ODBC) links to improve ETL performance.

When the source database is Oracle, you can use Database Link Mode and create a compatible DB Link from the Oracle Warehouse to the source. The Loading Knowledge Module (LKM) assumes that a database link already exists with the following naming convention: <DATASERVER_NAME>.WORLD@DSN_<DSN_ID>. This database link must exist in the warehouse and be accessible using the warehouse credentials (user specified against the warehouse connection in ODI).

To use a database link for ETL:

1. Have a database administrator create a private database link.

2. In Configuration Manager, update the ETL_SRC_VIA_DBLINK parameter.
 - a. Log in to Configuration Manager.
 - b. On the Tasks bar, click the **Manage Data Load Parameters** link.
 - c. In the Search pane, select the source system in the **Source Instance** list.
 - d. Select Code in the **Parameter** list and enter ETL_SRC_VIA_DBLINK in the adjacent field.
 - e. Click **Search**.
 - f. In the Data Load Parameters list, click the default No value to open the Edit Parameter Value dialog.
 - g. In the **Parameter Value** list, select Yes.
 - h. Click **Save and Close**.

Fixing Event Queue Knowledge Module

Certain ETL maps that use the IKM BIAPPS Oracle Event Queue Delete Append KM can fail, requiring changes to the IKM BIAPPS Oracle Event Queue Delete Append KM.

Error Message

You must fix the IKM BIAPPS Oracle Event Queue Delete Append KM when ETL maps fail with the following error:

```
ODI-17517: Error during task interpretation.
Task: 11 java.lang.Exception: BeanShell script error: Sourced file: inline evaluation
of: `` if (errLogSupport.equalsIgnoreCase("Y") ) &&
errLogOption.equalsIgnoreCase("Y") . . . '' : Attempt to resolve method:
equals() on undefined variable or class name: isPartitioned : at Line: 8 : in
file: inline evaluation of: `` if (errLogSupport.equalsIgnoreCase("Y") ) &&
errLogOption.equalsIgnoreCase("Y") . . . '' : isPartitioned .equals ( "Y" )
```

Affected Scenarios

The maps affected by the KM are listed here. After the Knowledge Module (KM) fix, regenerate the scenarios present in these mapping folders. The Post Load Process (PLP) and Source Independent Load (SIL) scenarios are mandatory and Source Dependent Extract (SDE) scenarios of the PLVs you use need to be regenerated. For example, if you are using Oracle E-Business Suite, then regenerate those SDE, SIL, and PLP scenarios.

PLP:

- PLP_WorkforceCompensationFact_Quarter
- PLP_SupervisorHierarchy_Analysis_Top
- PLP_SupervisorStatusDimension
- PLP_WorkforceBalanceAggregate_Temp
- PLP_WorkforceCompensationFact_Year
- PLP_WorkforceEventFact_Age

- PLP_WorkforceEventFact_Merge
- PLP_WorkforceEventFact_Month
- PLP_WorkforceEventFact_Pow
- PLP_Workforce_GainLossFact_NonSupChanges
- PLP_Workforce_GainLoss_Tmp_SupChanges
- PLP_Workforce_Gen01_GainLossFact_Frozen
- PLP_Workforce_Gen01_GainLossFact_Live

SIL

- SIL_AbsenceEventFact
- SIL_AssignmentSupervisorDimension
- SIL_WorkforceEventFact

Fusion SDE

- SDE_FUSION_AssignmentSupervisorDimension
- SDE_FUSION_AssignmentSupervisorDimension
- SDE_FUSION_DomainGeneral_ProjectAwardFundingSource
- SDE_FUSION_DomainGeneral_ProjectAwardInstitution
- SDE_FUSION_DomainGeneral_ProjectAwardKeyword
- SDE_FUSION_PersistedStage_WorkforceEvent
- SDE_FUSION_PersistedStage_WorkforceEvent
- SDE_FUSION_WorkforceEventFact
- SDE_FUSION_WorkforceEventFact

E-Business Suite SDE

- SDE_ORA_AssignmentSupervisorDimension
- SDE_ORA_HRPersonLegDimension
- SDE_ORA_PersistedStage_WorkforceEvent
- SDE_ORA_PersistedStage_WorkforceEvent_Supervisor
- SDE_ORA_WorkforceEventFact

PeopleSoft SDE

- SDE_PSFT_AssignmentSupervisorDimension
- SDE_PSFT_HRPersonLegislation_XLegs
- SDE_PSFT_PersistedStage_WorkforceEvent
- SDE_PSFT_PersistedStage_WorkforceEvent_Headcount

- SDE_PSFT_PersistedStage_WorkforceEvent_PositionHolder
- SDE_PSFT_PersistedStage_WorkforceEvent_WorkerPosition
- SDE_PSFT_WorkforceEventFact

To make the changes required:

1. In ODI Designer, log in as a user with privileges to edit KMs.
2. In the repository, expand **Knowledge Modules**, then **Integration (IKM)**, and open the **IKM BIAPPS Oracle Event Queue Delete Append KM**.
3. Double-click **Create I\$ Table**.
4. Enter the following code:

```
<$ if (errLogSupport.equalsIgnoreCase("Y")
&& errLogOption.equalsIgnoreCase("Y") ) { $>

/* Creates the flow table which is the replica of Target table during Error
Logging Mode */
CREATE TABLE <%=odiRef.getTable("L", "INT_NAME", "W")%>
AS SELECT <%=odiRef.getColList("", "[COL_NAME]", " ", "\n\t", "", "(INS and REW)")%>
      CAST(' ' AS VARCHAR2(100)) ERROR_TYPE_IND
      ,CAST(' ' AS UROWID) DIAGNOSTIC_ROWID
      ,CAST(' ' AS VARCHAR2(10)) IND_UPDATE
FROM <%=odiRef.getTable("L","TARG_NAME","A")%> <
  <%=odiRef.getOption("FLOW_TABLE_OPTIONS")%>
WHERE 1=2
<$ }
else if ( "<%=odiRef.getOption("FLOW_CONTROL")%>".equals("1") ||
  ("<%=odiRef.getOption("OBI_EVENT_QUEUE_UPDATE")%>".equals("1") && "<
%=refreshType%>".equals("TARGET")) ||
  ("IS_INCREMENTAL".equals("Y") ) && (errLogSupport.equalsIgnoreCase("N") ||
errLogOption.equalsIgnoreCase("N") ) ) { $>
CREATE TABLE <%=odiRef.getTable("L", "INT_NAME", "W")%>
( <%=odiRef.getColList("", "[COL_NAME]\t\t[DEST_WRI_DT] NULL", " ", "\n\t", "", "UK")
%>
  <%=odiRef.getColList(",\n\t", "[COL_NAME]\t\t<? if (\u0022[DEST_DT]\u0022.equals
(\u0022NUMBER\u0022)) {>NUMBER<?} else if (\u0022[DEST_DT]\u0022.equals
(\u0022VARCHAR2\u0022)) {>VARCHAR2(4000)<?} else {>[DEST_WRI_DT]<?}> NULL", " ",
\n\t", "", "NOT UK")%>
)
<%=odiRef.getOption("FLOW_TABLE_OPTIONS")%>
<$ }
else { $>
/* Step bypassed. It runs only in error logging mode if ETL diagnostics is
supported */
<$ } $>
```

5. Regenerate the scenarios.

Customizing the Oracle Business Analytics Warehouse

You can customize the ETL functionality in Oracle Business Intelligence Applications (Oracle BI Applications).

Topics

- [What is Customization in Oracle Business Intelligence Applications?](#)
- [Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables](#)
- [Category 2 Customizations: Adding Additional Tables](#)
- [Category 3 Customizations: Adding New Data](#)
- [Customizing Stored Lookups and Adding Indexes](#)
- [About Resolving Dimension Keys](#)
- [Custom Domains](#)

What Is Customization in Oracle BI Applications?

In Oracle BI Applications, customization allows you to change the preconfigured behavior and 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 types of customization available for you. Data sources are one of the following types:

- Packaged applications (for example, Oracle Fusion or Oracle E-Business Suite), which use prepackaged adapters.
- Non-packaged data sources, which use the Universal adapter.

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

		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

Customizations are grouped into the following categories:

- **Category 1.** In a Category 1 customization, you add additional columns from the source systems that have pre-packaged adapters and load the data into the existing Oracle Business Analytics Warehouse tables. See [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 Source Dependent Extract (SDE) and Source Independent Load (SIL) mappings. See [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. See [Category 3 Customizations: Adding New Data](#).

When you customize ETL Packages and Interfaces, you usually work in the \Oracle BI Applications\Mappings folder in the **Projects** view in the Designer of Oracle Data Integrator (ODI).

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.

Note: You must not rename, delete, or move packaged objects in the ODI repository unless you are directed to do so by Oracle Support. For example, you must not move a datastore in the Fact Staging folder to the Dimension Staging folder, or must not rename a packaged index named W_EXAMPLE_F_U1 to W_EXAMPLE_F_U2, unless directed by Oracle Support. You can make copies of packaged ODI objects and place them in custom folders, but do not delete, rename, or move the original objects.

About the Customization Process

You can customize your ETL functionality after you have performed a Business Analysis and Technical Analysis.

However, this information does not cover the other typical tasks that you must perform:

- **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 must determine the technical changes you need to make, by identifying source tables, staging tables, target tables, and ODI Packages and Interfaces that you must modify.
- **Repository (RPD) Modification** — having made the customizations in the ETL functionality, you must modify your repository to expose the new data in your dashboards.

Patching Customized Objects

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

A patch only installs changed repository objects, not the whole Work Repository. Therefore, you must 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 must manually re-apply customizations that you have made to the Supply Chain and Order Management application. The patch does not affect customizations in other applications.

As part of customizing an ETL task (including interfaces and package under a specific task folder), you copy the task folder to be customized to a Custom folder, version the copy once to freeze the original state, and version the copy again to capture the customized state. To modify and version ETL customizations, see [Extending Mappings in Oracle Business Analytics Warehouse](#).

You can customize and patch different ODI objects. Depending on the object, you must follow different steps to merge the patch and customizations.

If an object is customized, but is not being patched, this section does not apply. Similarly, if an object is patched but is not customized, you don't need to follow these steps.

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.

Patching Customized ETL Tasks

If you have customized the ETL task to be patched, follow these steps. Note that there are two use cases where an ETL task might have been customized, either at the direction of Oracle to quickly apply a fix before a patch becomes available or as the result of normal implementation activities to enhance Oracle BI Applications with additional functionality.

Customization Performed at Direction of Oracle

If you have customized at the direction of Oracle as a means to manually implement a fix, you must have a separate copy of the original object in a separate Custom or Patch folder. The ETL task has a scenario associated with it whose version is set to a value

higher than 001. To apply a patch, simply delete this scenario. The patched out-of-the-box scenario is automatically used the next time when the load plan runs.

Customization Performed as an Extension of Functionality

If you have customized the object being patched to introduce new behavior, merge the fix and the customizations into a single ETL task. When you first customize the task, the customization methodology requires that you create a copy of the ETL task and version it, for example to version 001. This version represents the original state of the ETL task. Prior to customization, you version the ETL task again, for example to version 002. You then apply your customizations to the new version. You can use the version compare utility of ODI to note the changes that have been introduced.

Prior to patching the out-of-the-box ETL task, you must version the out-of-the-box ETL task, to version 001 for example, then version it again, for example to 002. You then apply the patch. The 001 version represents the original state of the ETL task and 002 now represents the patched state. You can compare the out-of-the-box 002 version to the out-of-the-box 001 version to determine the changes introduced by the patch.

Note that you now have two original, or 001, versions, both representing the original state of the ETL task. You also now have two 002 versions, one a custom version and the other a patched version.

To get the fixes from the patch to appear in the custom folder, one approach is to manually apply the changes noted in the patched 002 version directly in the customized 002 version and then regenerate the scenario. This is the simplest approach and is recommended when a patch fix is relatively simple to determine and deploy.

If the patch fix is relatively complex while the customizations are relatively simple, an alternate approach is to re-apply the customizations to a new copy of the patched version. In this case, you rename the 002 custom version of the ETL task, for example to include Old or Prior as a suffix, copy the 002 patched version to the Custom adaptor, version it, version it again, then re-apply your customizations to the latest version. Generate the scenario with a larger number. If the original ETL task uses 001 and your prior customized ETL task uses 002, then you use 003 with this latest copy.

Patching Customized Datastores

As part of the customization methodology, datastores that are customized are first versioned to represent the original out-of-the-box state and versioned again to represent the customized state. If you have customized a datastore and want to patch the datastore, the patching process merges the datastore to reflect the custom and patch changes in the same datastore. You might want to create another version of the datastore prior to applying the patch, which represents the patched and customized state.

Patching Other Objects

Oracle does not recommend directly customizing commonly used objects such as Knowledge Modules, Variables, and User Defined Functions. If you need to customize such an object to manually apply a fix at Oracle's direction or to enhance out-of-the-box functionality, make a copy of the object, version to freeze the original state, and then version again to reflect the customized state. At this point, ETL tasks follow the same customization steps and are modified to use the custom copies of these objects. Follow the same steps as patching customized ETL tasks.

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

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

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

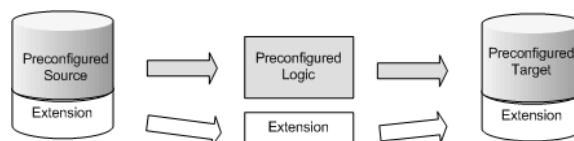
To see additional columns in the Oracle Business Analytics Warehouse, you must pass the columns first through the ETL process. The existing mappings and tables are extensible. Oracle BI Applications provides a methodology to extend preconfigured mappings to include these additional columns and load the data into the existing tables.

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

For example, if you want to calculate revenue differently from the existing logic, 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, the preconfigured logic is shaded in gray. Do not modify anything contained within these objects. Add customizations to the existing objects rather than creating new packages and interfaces, which allows them to run in parallel to the existing logic.



Extending Mappings in Oracle Business Analytics Warehouse

The most common reason for extending the Oracle Business Analytics Warehouse is to extract the 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 must 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 must 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 an existing Adaptor folder as this copies 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` and `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**, then click **Finish**.

- h. Repeat Steps a-g 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 require increasing the version in the comment so that you can compare to the original task to identify any changes.

- a. Right-click the Task folder and select **Version**, then **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. Follow Steps 1-4 and create another version of the copied task.

You should indicate in the version comment that this is the customized version.

6. Version the model that the datastore to be customized exists in, for example, Oracle BI Applications.

You cannot version submodels and datastores. You should indicate in the version comment that 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.

You can compare the models to show differences. If the model ever needs to be patched, you should version the model again to compare the patched version 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 the 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 repository, 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 a new scenario for the custom task using the same scenario name as the original.

ODI enforces unique scenario names and versions. Use the same scenario name for the custom task so that the load plan executes this ETL task rather than the out-of-the-box ETL task. Retain the same scenario name, but use a different, higher version number. The load plan always executes the scenario with the largest version number. By using the same scenario name, but with a larger higher version number, the load plan executes the custom scenario without requiring any changes.

- a. Note the name of the out-of-the-box scenario.

Navigate to the original **Task folder -Packages -Package Name -Scenarios - Scenario Name**. Make a note of the Scenario Name (you can double-click the scenario and use CTRL-C to copy the scenario name).

- b. Generate a scenario for the custom task.

Navigate to the custom Task folder **-Packages -Package Name**, then right-click the package and select **Generate Scenario**. Use the original scenario name as noted (you can use CTRL-V to paste the name if you copied it). Set the version number to 002 (or any value larger than the default, 001, used by the out-of-the-box scenario).

- c. Select the **Generate scenario as if all underlying objects are materialized** option, then click **OK**.

- d. Select **Use All** from the **Startup Parameters** list.

- e. Click **OK**.

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

11. Generate the Load Plan.

12. Execute the Load Plan.

Example of Extending the Oracle Business Analytics Warehouse

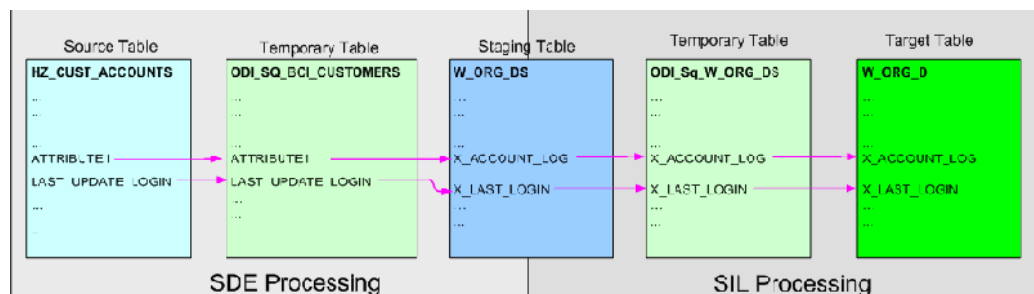
This working example describes adding columns from source systems that have pre-packaged adapters and loading the data into the existing Oracle Business Analytics Warehouse tables (known as a Category 1 customization).

In this example, a company has identified additional fields in a source system table HZ_CUST_ACCOUNTS that must be added to the Oracle Business Analytics Warehouse table W_ORG_D. Data is passed from an existing source table to an existing target table, known as a category 1 customization. The company uses an extension field to capture information related to organizations referred to as ACCOUNT_LOG. In addition, the company wants to include the name of the person who last updated the record as an attribute of the organization.

In this example, you want to extract information from the following two fields that are not extracted by the out-of-the-box application:

- HZ_CUST_ACCOUNTS.ATTRIBUTE1
ATTRIBUTE1 is currently not extracted from the source table HZ_CUST_ACCOUNTS into the temporary table ODI_SQ_BCI_CUSTOMERS.
- HZ_CUST_ACCOUNTS.LAST_UPDATE_LOGIN
LAST_UPDATE_LOGIN is currently extracted from the source table HZ_CUST_ACCOUNTS into the temporary table ODI_SQ_BCI_CUSTOMERS, but is not loaded into the staging table W_ORG_DS.

The following diagram shows the two fields ATTRIBUTE1 and LAST_UPDATE_LOGIN as they pass from the source system table to the target table via the tables: HZ_CUST_ACCOUNTS to ODI_SQ_BCI_CUSTOMERS to W_ORG_DS to ODI_Sq_W_ORG_DS to W_ORG_D.



The customization is done in two parts:

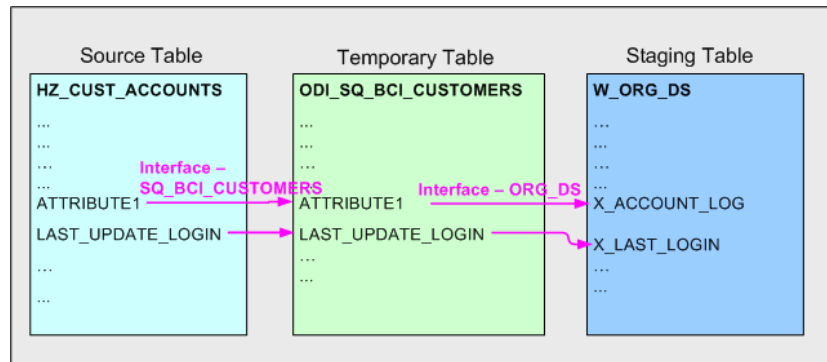
- SDE Processing that extracts data from the source system and loads it into the staging area. See [Example of Extracting Data from an Oracle EBS 11.5.10 Data Packaged Source into the ODI Staging Area](#).

- SIL Processing that extracts data from the staging area and loads it into the target table. See [Example of Loading Data from the Staging Area into an Existing Target Table](#).

Example of Extracting Data from an Oracle EBS 11.5.10 Data Packaged Source into the ODI Staging Area

This section shows how data is extracted from an existing source table into the staging area.

The following diagram shows the new ETL mappings that to load the new data into the staging area, and the ODI Interfaces to modify.



Note: The new ETL mappings diagram only shows the incremental interfaces.

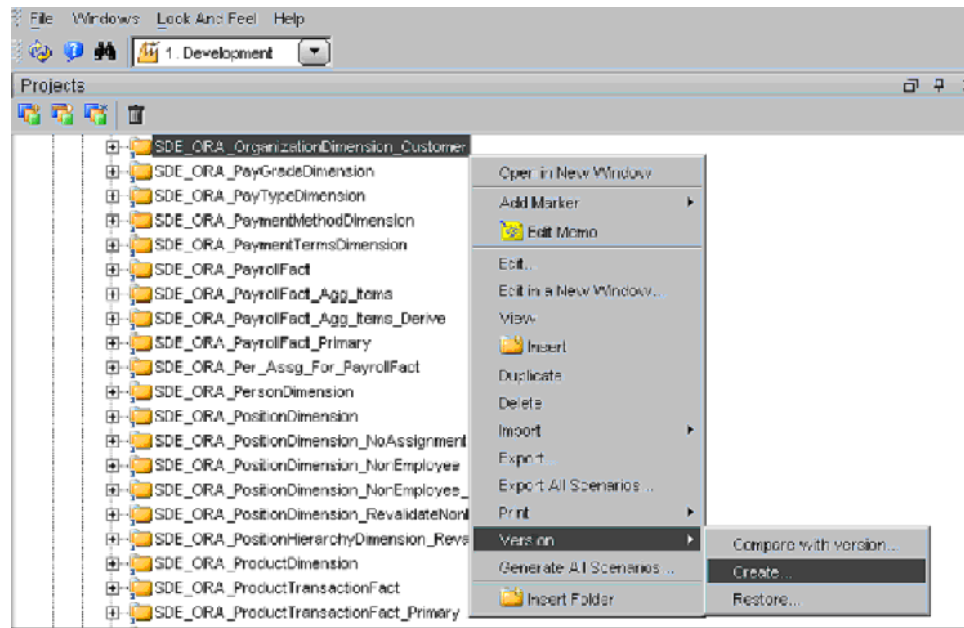
To customize the ETL process for loading these two fields into the staging area:

- Extract the `HZ_CUST_ACCOUNTS.ATTRIBUTE1` value from the source table `HZ_CUST_ACCOUNTS` into the temporary table `ODI_SQ_BCI_CUSTOMERS` using the Interfaces `SQ_BCI_CUSTOMERS` and `SQ_BCI_CUSTOMERS_FULL`.
Then, load the `ODI_SQ_BCI_CUSTOMERS.ATTRIBUTE1` value from the temporary table `ODI_SQ_BCI_CUSTOMERS` into the `X_ACCOUNT_LOG` field in the staging table `W_ORG_DS` using the Interfaces `ORG_DS` and `ORG_DS_FULL`.
- Load the `SQ_BCI_CUSTOMERS.LAST_UPDATE_LOGIN` value from the temporary table `ODI_SQ_BCI_CUSTOMERS` into the `X_LAST_LOGIN` field in the staging table `W_ORG_DS` using the Interfaces `ORG_DS` and `ORG_DS_FULL`.

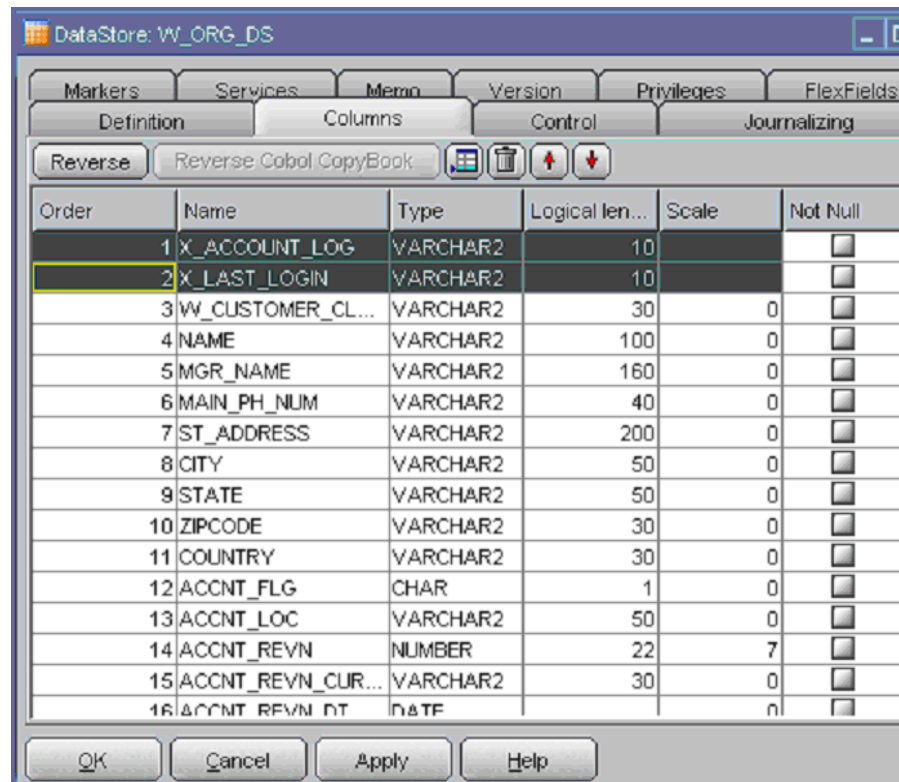
Note: Remember that `LAST_UPDATE_LOGIN` value is already extracted from the source table `HZ_CUST_ACCOUNTS` into the temporary table `ODI_SQ_BCI_CUSTOMERS`, but is not loaded into the staging table `W_ORG_DS`.

To extract data from an Oracle EBS 11.5.10 Data Packaged Source:

1. In ODI Designer, display the **Projects** view, and expand the Oracle BI Applications\Mappings\SDE_ORA11510_Adaptor folder.
2. Right-click the `SDE_ORA_OrganizationDimension_Customer` folder, and select **Version**, then **Create** to display the Create: *Object* dialog, and specify a unique version number and optional version description.



3. Display the **Models** view, expand the Dimension Stage folder, and edit the W_ORG_DS datastore to display the DataStore: *Name* dialog, and display the Columns tab.



4. Create the following columns:
 - X_ACCOUNT_LOG (VARCHAR2(10))
 - X_LAST_LOGIN (VARCHAR2(10))

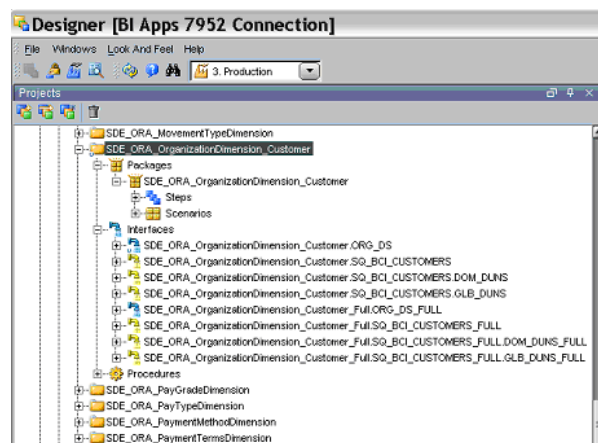
5. In the **Models** view, right-click the model **Oracle BI Applications** and select **Generate DDL** to display the Generate DDL dialog.

The Generate DDL option deploys the changes in the database.

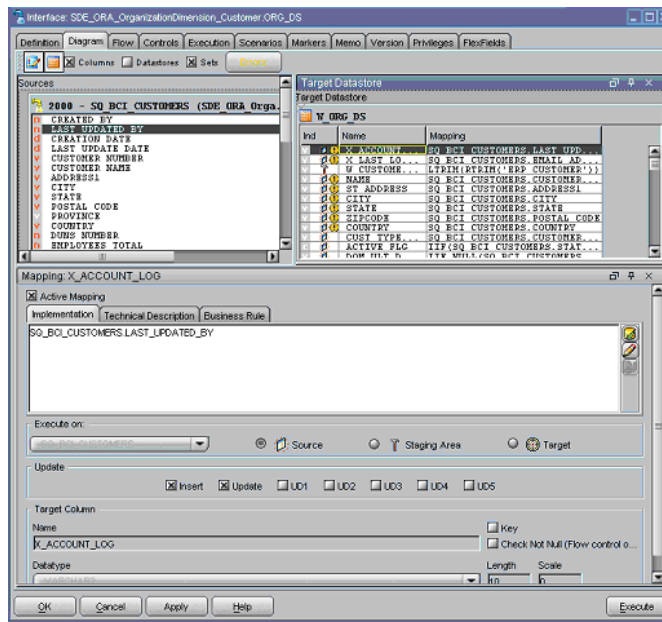
6. Select the option in the Synchronize column next for the W_ORG_DS table.
7. Click the (...) button to the right of the **Generation Folder** field to display the Select a folder dialog, and select the \Utilities\System folder, and click **OK**.
8. When the Procedure: DDL *Name* dialog is displayed, click **Execute**.

Display ODI Operator and make sure that the procedure executes successfully.

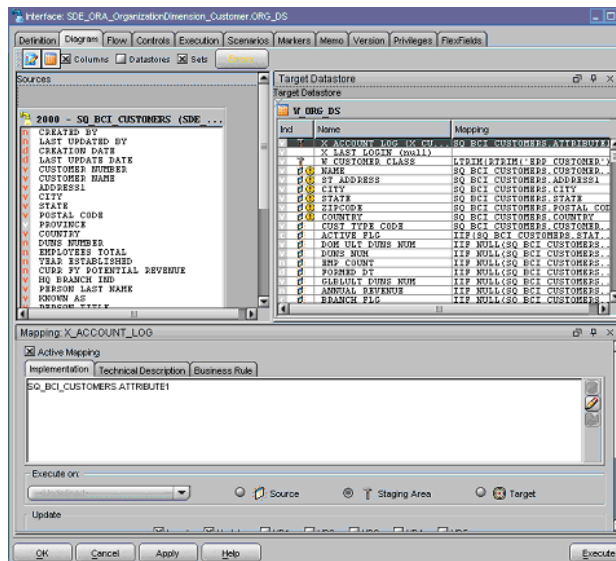
9. Display the **Projects** view, expand the Mappings folder, and expand the SDE_ORA_OrganizationDimension_Customer folder.



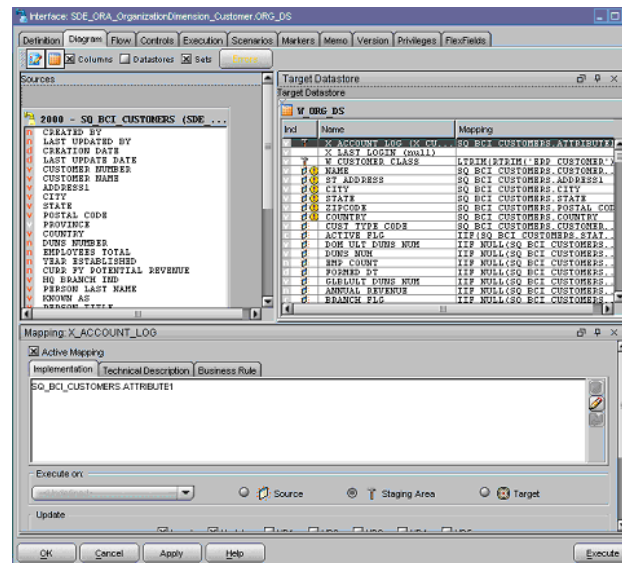
10. Edit the Interface
SDE_ORA_OrganizationDimension_Customer.SQ_BCI_CUSTOMERS to display the Interface: *Name* dialog.
 - a. Display the Diagram tab, and select the **ATTRIBUTE1** field in the Target Datastore area.
 - b. Click the **Launch Expression Editor** icon to display the Expression Editor dialog, and then select **HZ_CUST_ACCOUNTS.ATTRIBUTE1** as the value in the **Implementation** field.
 - c. Click **OK** to save the details.
11. Repeat Step 10 for the Interface
SDE_ORA_OrganizationDimension_Customer.SQ_BCI_CUSTOMERS_FULL.
12. Edit the Interface SDE_ORA_OrganizationDimension_Customer.ORG_DS to display the Interface: *Name* dialog.
 - a. Display the Diagram tab, and select the **X_ACCOUNT_LOG** field in the Target Datastore area.



- b. Click the **Launch Expression Editor** icon to display the Expression Editor dialog, and then select **SQ_BCI_CUSTOMERS.ATTRIBUTE1** as the value in the **Implementation** field.



- c. Select the **X_LAST_LOGIN** field in the Target Datastore area.
- d. Click the **Launch Expression Editor** icon to display the Expression Editor dialog, and then select **SQ_BCI_CUSTOMERS.LAST_UPDATE_LOGIN** as the value in the **Implementation** field.



e. Click **OK** to save the details.

13. Repeat Step 12 for the Interface
SDE_ORA_OrganizationDimension_Customer.ORG_DS_FULL.

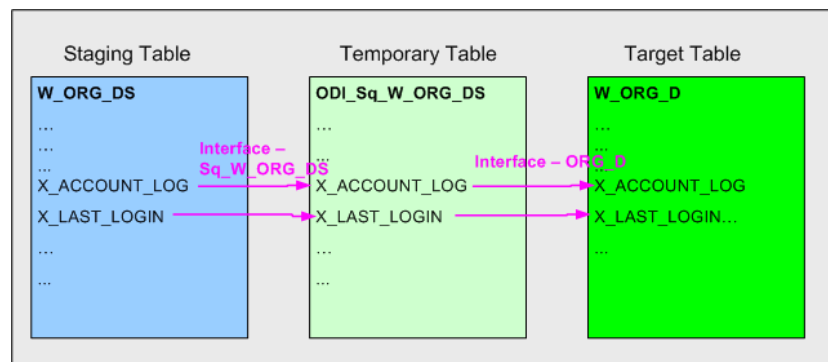
14. Right-click the scenario **SDE_ORA_OrganizationDimension_Customer** and select **Regenerate**.

Now that you have set up the ETL process for extracting and staging the data, you must load the new data into the data warehouse. See [Example of Loading Data from the Staging Area into an Existing Target Table](#).

Example of Loading Data from the Staging Area into an Existing Target Table

This section shows how data is loaded from the staging area into an existing target table.

The following diagram shows the new ETL mappings to load the new data from the staging area into the target table and the ODI Interfaces to modify.



To customize the ETL process to load these two fields into the staging area:

- Load the X_ACCOUNT_LOG value and X_LAST_LOGIN value from the staging table W_ORG_DS into the temporary table SQ_W_ORG_DS using the Interfaces Sq_W_ORG_DS and Sq_W_ORG_DS_FULL.

- Load the X_ACCOUNT_LOG value and X_LAST_LOGIN value from the temporary table ODI_Sq_W_ORG_DS into the Target table W_ORG_D using the Interfaces ORG_D, ORG_D_FULL, and ORG_D_UNSPC.

To extract data from an Oracle EBS 11.5.10 Data Packaged Source:

1. In ODI Designer, display the **Projects** view, expand the Oracle BI Applications \Mappings\SILOS folder.
2. Right-click the SIL_OrganizationDimension folder, and select **Version**, then **Create** to display the Create: *Object* dialog, and specify a unique version number and optional version description.
3. Display the **Models** view, expand the Dimension Stage folder, and edit the W_ORG_DS datastore to display the DataStore: *Name* dialog, and display the Columns tab.

4. Make sure that these columns are setup:

- X_ACCOUNT_LOG (VARCHAR2(10))
- X_LAST_LOGIN (VARCHAR2(10))

5. Repeat steps 3 and 4 for the W_ORG_D datastore.

6. In the **Models** view, right-click the **Oracle BI Applications** model and select **Generate DDL** to display the Generate DDL dialog.

The **Generate DDL** option deploys the changes in the database.

7. Select the check box in the Synchronize column next for the W_ORG_DS table.
8. Click the (...) button to the right of the **Generation Folder** field to display the Select a folder dialog, and select the \Utilities\System folder, and click **OK**.
9. When the Procedure: DDL *Name* dialog is displayed, click **Execute**.

Display ODI Operator and make sure that the procedure executes successfully.

10. Display the **Projects** view, expand the Mappings folder, and expand the SIL_OrganizationDimension folder.

11. Edit the Interface Sq_W_ORG_DS to display the Interface: *Name* dialog.

- a. Display the Diagram tab, and select the X_ACCOUNT_LOG field in the Target Datastore area.
- b. Click the **Launch Expression Editor** icon to display the Expression Editor dialog, and then select W_ORG_DS.X_ACCOUNT_LOG as the value in the **Implementation** field.
- c. Select the X_LAST_LOGIN field in the Target Datastore area.
- d. Click the **Launch Expression Editor** icon to display the Expression Editor dialog, and then select W_ORG_DS.X_LAST_LOGIN as the value in the **Implementation** field.
- e. Click **OK** to save the details.

12. Repeat Step 11 for the Interface Sq_W_ORG_DS_FULL.

13. Edit the Interface ORG_D to display the Interface: *Name* dialog.
 - a. Display the Diagram tab, and select the **X_ACCOUNT_LOG** field in the Target Datastore area.
 - b. Click the **Launch Expression Editor** icon to display the Expression Editor dialog, and then select **Sq_W_ORG_DS.X_ACCOUNT_LOG** as the value in the **Implementation** field.
 - c. Select the **X_LAST_LOGIN** field in the Target Datastore area.
 - d. Click the **Launch Expression Editor** icon to display the Expression Editor dialog, and then select **Sq_W_ORG_DS.X_LAST_LOGIN** as the value in the **Implementation** field.
 - e. Click **OK** to save the details.
14. Repeat Step 13 for the Interface ORG_D_FULL and ORG_D_UNSPC.
15. Right-click the scenario **SILOS\SIL_OrganizationDimension** and select **Regenerate**.

Tips for Modifying the SQ_BCI_ Interface

Follow these tips to modify the SQ_BCI_Interface:

- Make sure that you define a new source table on the right side of a LEFT OUTER join syntax with the existing source tables. Using an INNER join or a RIGHT OUTER join might result in loss of records.
- Make sure that you define joins to match on a unique set of values. If you do not define a join that ensures a unique relationship, you might get a Cartesian product, which changes the granularity and results in duplicate errors.

If it is not possible to define a unique join, then do the following:

1. Create an inline view interface sourcing from the new table, outputting necessary columns plus a column LKP_ACTIVE.

For example, you might specify the expression for LKP_ACTIVE as:

```
IS_FIRST(  
ARG_GROUP(columns to be partitioned by),  
ARG_GROUP(columns to be ordered by))
```

Note: In the given example (specify the expression for LKP_ACTIVE), the IS_FIRST command and the matching filter are only needed if multiple records might be returned.

2. Bring the inline view interface into an existing interface with a filter LKP_ACTIVE=1, which guarantees that at most one record is returned.

As a best practice, you can comment custom code that you introduce. Comments should include the developer's name and the date indicating when the code is added.

Other Types of Customizations Requiring Special Handling

These types of customization require special handling.

Topics:

- [Modifying Category 2 SCD Behavior](#)
- [Adding a Dimension to an Existing Fact](#)
- [Adding a DATE_WID Column to a Fact](#)

Modifying Category 2 SCD Behavior

The Oracle 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.

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 selecting **Property Inspector** under **View**.
 - 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.

Enabling or Disabling Type II Behavior Associated with a Dimension

You can 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 have no effect as all columns are treated as Type I.

To enable or disable Type II behavior:

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, from the **Slowly Changing Dimensions Behavior** 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

Adding a Dimension to an Existing Fact

You can 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.

Customizing Fact Datastores and Tasks

You must extend the Fact related datastores, *W_Fact Name_FS* and *W_Fact Name_F* and tasks to reflect the new dimension.

To extend the Fact related 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).

The Oracle BI Applications Model should already be versioned.

- a. Navigate to **Models**, Oracle BI Applications (Folder), **Oracle BI Applications** (Model), **Fact Stage** (Submodel) and double-click the **Fact Staging** Table.
 - b. In the Columns subtab, select the **X_CUSTOM** column.
 - c. 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).

Repeat Step 1 to add a column to the fact datastore.

3. Add a foreign key constraint to the fact table that refers to the created custom dimension table.

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**.

The naming convention is *FK_Fact Table_Dimension Table*.

- c. Enumerate each WID column with a numeric suffix, for example, *FK_WC_CUSTOM_F_WC_CUSTOM_D1*. Type must be User Reference if there are multiple WID columns that need to reference the same dimension table.
 - d. Select **Custom Dimension** from the **Table** 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.
 - g. 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**.

The naming convention is *Fact Table_Fn*.

- c. Enumerate each of these indexes with a numeric suffix, for example, *WC_CUSTOM_F1*.
- d. Select the **Not Unique Index** option.
- e. In the Columns subtab, add the WID column using the shuttle button.
- f. In the Control subtab, select the **Defined in the Database** and **Active** options.
- 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)`

Adding 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. You must not perform 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 use this column in the ETL and various date-related expressions in the repository also use this column. All packaged fact tables have a single canonical date already identified. When creating custom fact tables, nominate one Date WID column as the canonical date and use consistently.

Follow the same steps as adding a dimension to a fact with the following changes. You must not create a custom SDE as you use the existing Date dimension.

You must extend the facts related datastores, *W_Fact Name_FS* and *W_Fact Name_F* and tasks to reflect the new dimension.

To extend the facts related datastores and tasks:

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 must 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)`.

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.

You can build entirely new tables to load the 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 the existing table structures. When you create a new table, you must define the table and indices in ODI Designer Models area first. The destination model for the Oracle Business Analytics Warehouse is Oracle BI Applications.

About the Main Required Columns

For custom staging tables, the required columns are:

- `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 plus these:

- `ROW_WID` — Stores 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.

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 are loaded in the same data warehouse tables from different sources if this column is given a different value for each data source.

Additional Information About Customizing

Learn about miscellaneous information about customization in Oracle BI Applications.

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 is designed to load only the changed data (new and modified). If the data is loaded without the incremental process, the loaded data is 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 to determine insert or update. Look at the similar mappings in the preconfigured folder for more details.

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 uses 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, register the indices in the ODI Model. See [Adding an Index to an Existing Fact or Dimension Table](#).

Adding New Fact Tables to 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.

Placeholder fact and dimension groups to which you can add your own custom content are provided out-of-the-box. These follow the X_CUSTOM naming convention. You might introduce your own custom fact and dimension groups. See [About Creating Custom Fact and Dimension Groups](#). The following steps assume content is being added to an out-of-the-box X_CUSTOM fact or dimension group. If you create custom groups, use those instead.

To add a new fact table:

1. Create the custom fact datastores and tasks.
 - a. Create a WC_<fact name>_F datastore under the **Oracle BI Applications – Fact** model.
 - b. 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.

The specific submodel that a table belongs to drives the table maintenance behavior. For example, tables in the Fact Stage submodel are always truncated

during each ETL run while tables in the Fact submodel are only truncated during a Full ETL run.

You can define a fact in ODI either manually, or 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 Reverse-engineering Knowledge Modules (RKM) of Oracle BI Applications. If using the RKM, the imported table is automatically placed in the Other submodel and you must move the imported table into the Fact Staging and Fact submodels as appropriate. Set the OLAP type also for the fact table to Fact Table.

2. Add a foreign key constraint to all dimension tables associated with this fact. If the custom fact table joins to any custom dimension tables, be sure to complete the steps to introduce the custom dimensions prior to creating the custom fact table.

The Dimension SDE task is 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** 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.
- e. 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.

Note: If you are using a custom fact group, in the following steps, replace references to X_CUSTOM with the name of your custom fact group.

- a. In Designer, navigate to **Load Plans and Scenarios, BIAPPS Load Plan**, then **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, set the Version as -1, and set the Step Name to 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.

Note: If you are using a custom fact group, in the following steps, replace references to X_CUSTOM with the name of your custom fact group.

- a. In Designer, navigate to **Load Plans and Scenarios, BIAPPS Load Plan**, then **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, set the Version as -1, and set the Step Name to match the Task name. Set the Restart Type to **Restart from failed step**.

Creating Fact Tables

You can create a new fact table using the WC_SAMPLE_FS and WC_SAMPLE_F datastores as templates. You should set the OLAP type to Fact Table and use the Generate DDL procedure to create the tables in the database.

To create a fact table manually:

1. In Designer, navigate to **Models, Oracle BI Applications (Folder), Oracle BI Applications (Model), Fact Stage (Submodel)**, then right-click the **WC_SAMPLE_FS datastore** and select **Duplicate Selection**.
2. Double-click the new datastore and rename it.

Name and Resource Name should match the actual table name. Alias is the same or a more user-friendly value.

3. In the Columns subtab, add all columns.
4. Repeat the same steps to create the Fact Table by copying the WC_SAMPLE_F datastore under the Facts submodel.
5. For the fact table, set the OLAP type to **Fact Table**.
6. Generate the DDL to create the table in the database.

Importing Fact Tables into ODI

You can import the existing fact tables into ODI using the RKM of Oracle BI Applications. The imported table is automatically placed in the Other submodel and you must move the imported table into the Fact Staging and Fact submodels as appropriate. You should set the OLAP type to Fact Table and use the Generate DDL procedure to generate the required DDL for creating the database trigger in the database.

To import fact tables into ODI:

1. In Designer, navigate to **Models, Oracle BI Applications (Folder)**, and then 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 **Reverse Engineer**.

A session is started that imports the table or tables into ODI. The Reverse Engineer process places all tables in the Other submodel.

4. Drag and drop W_%_FS tables into the Fact Stage submodel and the W_%_F table into the Fact submodel.
5. For the fact table, set the OLAP type to **Fact Table**.
6. Generate the DDL to create the table in the database.

Creating Custom Dimension Datastore and Tasks

You can 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 are always 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 is not implemented properly for tables in such a submodel.

You can define a dimension 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 RKM of Oracle BI Applications. If you use the RKM, the imported table is automatically placed in the Other submodel and you must move the imported table into the Dimension Staging and Dimension submodels as appropriate. Also, set the OLAP type for the dimension to Dimension or Slowly Changing Dimension as appropriate.

The X_CUSTOM placeholder load plan components are provided as templates, but do not use for new custom content. Create a new dimension group in Oracle BI Applications Configuration Manager. For information about how to do this, see [About Creating Custom Fact and Dimension Groups](#).

Creating Dimension Tables in ODI

You can create a new dimension table using the WC_SAMPLE_DS and WC_SAMPLE_D datastores as templates. You should set the OLAP type to Dimension or Slowly Changing Dimension as appropriate.

To create a dimension table in ODI:

1. In Designer, navigate to **Models**, Oracle BI Applications (Folder), **Oracle BI Applications** (Model), and then **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 is 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.

Importing Custom Dimension Tables into ODI

You can import the existing dimension tables into ODI using the RKM of Oracle BI Applications. The imported table is automatically placed in the Other submodel and you must move the imported table into the Dimension Staging and Dimension submodels as appropriate. And, you should set the OLAP type for the dimension to Dimension or Slowly Changing Dimension as appropriate.

To import a dimension table 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.

The Reverse Engineer process places all tables in the Other submodel.

4. 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.

Creating ODI Sequences for Custom Dimensions

You can create an ODI sequence for a custom dimension.

You can use a database sequence to populate the ROW_WID column of the dimension. You can use the Generate DDL procedure to generate the required DDL for creating the database trigger in the database. Use WC_SAMPLE_D_SEQ as a template.

To create an ODI sequence for the custom dimension:

1. In Designer, navigate to **Projects, BI Apps Project**, then **Sequences**.
2. Right-click the Sequence folder and select **New Sequence**.
3. Set name to *Dimension Name_SEQ*.
4. Select **Native** sequence.
5. Set the Schema to **DW_BIAPPS11G**.

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 database trigger populates the ROW_WID column.

6. Generate the DDL to create the table in the database.

Note: If you manually create the dimension in ODI, this generates the DDL to create both the table and sequence. If you import the dimension into ODI, this generates the DDL to create the sequence only.

Creating SDE and SIL Tasks

You can 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.

About Creating Custom Fact and Dimension Groups

Once you have completed customization steps to include custom tables and ETL tasks, create custom fact and dimension groups in Configuration Manager and ODI to incorporate your changes into the Load Plans that orchestrate ETL.

The Data Warehouse tables are populated by ETL tasks which are orchestrated by a Load Plan. When creating custom content, new data warehouse tables and associated ETL tasks are created and you must incorporate them into a Load Plan to populate the custom tables along with the standard tables.

You can build a Load Plan by assembling several Load Plan Components into an integrated load plan. You can organize Load Plan Components into level 0, 1, 2, and 3 components. Level 3 Load Plan Components execute the actual ETL tasks, while the other Load Plan Components are system Load Plan Components, and you can use the other load plan components in building up and organizing the Load Plan. The Load Plan Generator assembles these Load Plan Components into a combined Load Plan.

You must create custom Load Plan Components to have custom content such as custom dimensions or custom facts incorporated into a Load Plan that includes out-of-the-box content. The Load Plan Generator can then include these components when building a load plan.

To create custom tables and ETL tasks, see [Customizing the Oracle Business Analytics Warehouse](#). The following topics describe how to create new Load Plan Components to execute these ETL tasks. Use Pre-existing 3 Dims X_CUSTOM_DIM and 3 Dims

X_CUSTOM_FG Load Plan Components as templates and the steps in the following topics describe copying these X_CUSTOM components and updating the copies to reflect custom content. The steps are essentially the same regardless of which group you are adding; only the template you start with changes.

Creating Custom Fact and Dimension Groups in ODI

The first step in creating custom fact and dimension groups is performed in ODI Designer.

To create custom fact and dimension groups in ODI:

1. Add the custom load plan component.
 - a. In ODI Designer, navigate to the Designer tab and under **Load Plans and Scenarios**, expand the **BIAPPS Load Plan**, and then **Load Plan Dev Components**.
 - b. Expand the appropriate subfolder for the custom Load Plan Component you are adding.

If you are adding a SIL Load Plan Component, expand the SIL subfolder. If you are creating a SDE Load Plan Component, expand the SDE subfolder, then expand the appropriate product line subfolder.

2. Duplicate an existing X_CUSTOM Load Plan Component.

The following Load Plan Components are generally preconfigured with all required variables and other settings, so copying these components saves you the trouble of adding variables and configuring required settings:

- 3_SDE Dims X_CUSTOM_DIM <PLV_CODE>
- 3_SDE Fact X_CUSTOM_FG <PLV_CODE>
- 3_SIL Dims X_CUSTOM_DIM
- 3_SIL Fact X_CUSTOM_FG

3. Rename the Load Plan Component.

In the properties for the copied component, delete the Copy of prefix. For an SDE Load Plan Component, it is very important to retain the Product Line Version code (for example, EBS_11_5_10 and so on) as the suffix, because the suffix is used by Load Plan Generator to determine whether a Load Plan Component is included in a load plan that extracts from a particular data source. Rename the X_CUSTOM_DIM or X_CUSTOM_FG portion of the Load Plan Component name to match the code of the Dimension Group or Fact Group that is registered in Configuration Manager. Load Plan Generator uses this portion of the component's name to coordinate with Configuration Manager to include Load Plan Components for those Groups specified in Configuration Manager.

4. In the Steps subtab, rename the first, or Root, step to match the custom Group Code. Variables are assigned values at the Fact and Dimension Group level, and are refreshed at the root level. ETL refresh logic takes the name of the root step and passes it to Configuration Manager to get the value for the variable.
5. Add the Custom Level 3 Load Plan Component to the Level 2 System Load Plan Component.

- a. Navigate to the Designer tab and under **Load Plans and Scenarios**, expand the **Load Plan System Components**.
 - b. If you are adding a SDE Load Plan Component, expand the SDE subfolder. If you are adding a SIL Load Plan Component, expand the SIL subfolder.
 - c. If you are adding a Dimension Group, double-click the **Level 2 Dimension Group Load Plan Component**. If you are adding a Fact Group, double-click the **Level 2 Fact Group Load Plan Component**.
 - d. Navigate to the Steps subtab of the **Load Plan Component**, right-click the `$$$X_CUSTOM_DIM` Load Plan Step, and select **Duplicate Selection**.
 - e. Rename the Load Plan Step, replacing the `X_CUSTOM_DIM` or `X_CUSTOM_FG` suffix with the Group Code value used in the custom Load Plan Component and root step names. The portion within the brackets must match the custom Load Plan Component name exactly (ignoring the Product Line Version Code suffix in the case of SDE load plan components). Load Plan Generator uses this value to incorporate the Level 3 Load Plan Component into the generated load plan.
6. Add Load Plan steps to the 3 SDE Dims `X_CUSTOM_DIM <Product Line Version Code>` and 3 SIL Dims `X_CUSTOM_DIM` Load Plan Components you created.

The following steps use `X_CUSTOM` in describing the changes to the Load Plan Components. Replace this with the actual dimension name used when you created your custom Load Plan Components.

- a. 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.
- b. In the Steps subtab, select the `X_CUSTOM_DIM` step.
- c. Click the green +symbol near the top right and select **Run Scenario Step**.
- d. 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**.
- e. In Designer, navigate to **Load Plans and Scenarios, BIAPPS Load Plan, Load Plan Dev Components, SIL** and double-click the **3 SIL Dims X_CUSTOM_DIM** Load Plan Component.
- f. In the Steps subtab, select the `X_CUSTOM_DIM` step.
- g. Click the green +symbol near the top-right and select **Run Scenario Step**.
- h. 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**.

To associate your custom dimension with a fact table, see [Adding a Dimension to an Existing Fact](#).

Creating Custom Fact and Dimension Groups in Configuration Manager

The last step in creating custom fact and dimension groups is performed in Configuration Manager.

To create custom fact and dimension groups:

Note: If you encounter an error while creating a fact or dimension group with CUSTOM_FG or CUSTOM_DIM code, then create with a different code and rename the code to CUSTOM_FG/DIM later.

1. In order to add custom Fact or Dimension Groups, the user, if not the Administrative User, must be granted the following roles:
 - BIA_CONFIGURE_CREATE_EDIT_DIMGROUP_PRIV
 - BIA_CONFIGURE_CREATE_EDIT_FACTGROUP_PRIV
2. In Configuration Manager, click **Manage Business Intelligence Applications**.
3. Select an offering where the custom Fact or Dimension Group most closely fits.
4. In the Associated Fact and Dimension Groups table, click **Actions** and select either **Create Fact Group** or **Create Dimension Group**.
5. Enter X_CUSTOM_FG if you are creating a fact group and X_CUSTOM_DIM if you are creating a dim group.

Note: The **Group Code** value must exactly match the value used when creating the corresponding ODI Load Plan Component. Dimension groups use _DIM as a suffix while Fact groups use _FG as a suffix. As a best practice, use X_ as a prefix to easily identify custom groups.

6. Add a meaningful group description.

The remaining steps for creating Fact Groups and Dimension Groups are different.

Creating Custom Dimension Groups

To create a Custom Dimension Group, you must associate it to an existing Fact Group, so if the custom Dimension Group is to be associated with a custom Fact Group, be sure to add the Fact Group first.

To create a Custom Dimension Group:

1. In the Associated Fact and Dimension Groups list for the offering you added a new group for, click **Actions** and select **Create Dimension Group**.
2. In the Create Dimension Group dialog, select a Fact Group from the Associate Fact Group list, and add it to the right-hand list of associated Fact Groups.

Note: The Create Dimension Group dialog doesn't filter the Fact Groups by the offering you selected when creating the new group. Be sure to only select those Fact Groups that are actually associated with the offering you are adding the custom Dimension Group to.

3. Click **OK**.

The custom Dimension Group is now added to the offering and associated with a Fact Group.

Category 3 Customizations: Adding New Data

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

Topics:

- [Adding New Data as a Whole Row Into a Standard Dimension Table](#)
- [Configuring Extracts](#)
- [Configuring Loads](#)

Adding 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 a standard dimension table:

1. Identify and understand the existing structure of staging tables.

Refer to Oracle Business Analytics Warehouse Data Model Reference Version 11.1.1.7.1 and 11.1.1.8.1 (My Oracle Support Doc ID 1638086.1) for the table structures. Non-system columns might 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 must populate 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.

Configuring Extracts

Each application has prepackaged logic to extract particular data from a particular source. You must 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.

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, modify the extract interface to include the new data.

Extracting Data from a New Source Table

Extract interfaces (which have the SQ_* naming convention) reside in source-specific folders within the repository. Use the Extract interfaces 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.

Extracting New Data Using an Existing Source Table

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 might need to set up the extract interface to include data from these tables.

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 must determine which type of extension column to map the data to in the staging table. After you modified the extract interface, you must also 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.

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 can 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 load this data without modification, ODI loads January as the Months value, and February as the Status value. The remaining data for the first record (that is, March, Active) is not 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 loads the data correctly. In this example, for the first record, ODI loads 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 select 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 datastores that are associated with the modified CSV files. The CSV file that you modified might be associated with one or more datastores.

5. In ODI Designer, change the properties for each of these datastores to use the enclosing character.
 - 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.

Configuring Loads

You can customize the way Oracle BI Applications loads data into the Oracle Business Analytics Warehouse.

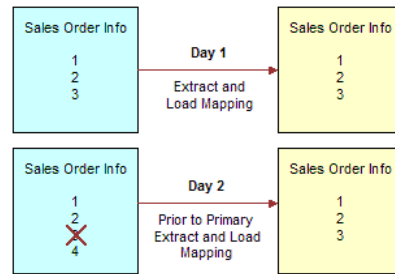
About Primary Extract and Delete Mappings Process

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 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

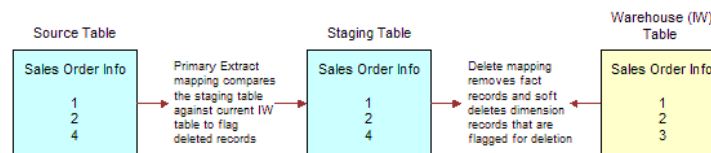
three is deleted and a new sales order is received, creating a disparity between the Sales Order information in the two tables.



The figure 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. When the extract and load mappings run, the new sales order is added to the warehouse.



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 are 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, 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

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 do not mark the archived records as deleted.

Customizing Stored Lookups and Adding Indexes

Learn about how this information applies to all categories of customization in Oracle BI Applications.

- [How Dimension Keys are Looked Up and Resolved](#)
- [Adding an Index to an Existing Fact or Dimension Table](#)

How Dimension Keys are Looked Up and Resolved

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.

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.

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 following table:

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).

Adding an Index to an Existing Fact or Dimension Table

Dimension and Fact Tables in the Oracle Business Analytics Warehouse use an ETL Index for Unique/Binary Tree index, and Query Index 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** option, 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.

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.

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.

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 following table:

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 table's foreign key).

Custom Domains

Domains are similar to list of values and lookups commonly found in transactional systems. These often have language independent code, and one or more language dependent values associated with them. In Oracle BI Applications, you can use domains to support multiple languages and aggregations.

For example, you might have the domain `GENDER` with the domain members `M` and `F`. To support multiple languages, you might have corresponding English values of `Man` and `Woman`, and French values of `Homme` and `Femme` – the appropriate value returned based on the user's preferred language.

Domains are often used as criteria in analyses. For example, a report such as `Total # of Employees by Gender` issues a SQL statement similar to `SELECT GENDER, COUNT(*) FROM Table GROUP BY GENDER`.

Oracle BI Applications ships a number of domains out-of-the-box to support multiple languages and analyses. You might need to extend Oracle BI Applications with additional domains to support your users' multi-language and analytical needs.

The two different types of domains are source-based and conformed.

- Source-based domains are values that are extracted directly from the source system and loaded into the data warehouse with minimal changes (other than to default for NULL values).
- Conformed domains are meant to provide a consistent set of analyses across multiple heterogeneous source systems that might use different codes to represent the same thing – the ETL process takes the source-based code and matches or conforms it to a Oracle BI Applications-specific code.

Oracle BI Applications supports only custom source domains in the data warehouse. Mapping a new custom source domain to an existing conformed domain is not supported.

See *About Multi-Language Support in Oracle Business Intelligence Applications Administrator's Guide*.

Extending Oracle BI Applications with Custom Source Domains

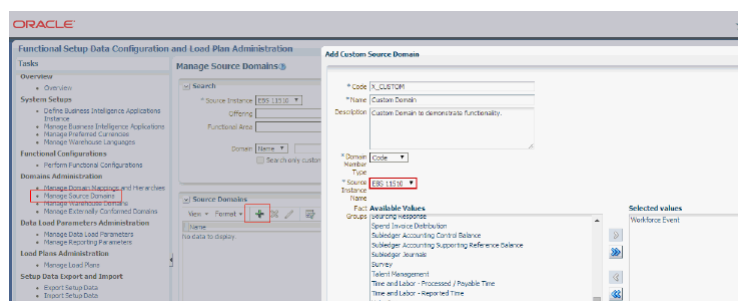
Typically, extending Oracle BI Applications involves defining a custom domain in Configuration Manager, extending data warehouse and staging tables with the new domain column, creating custom domain SDE tasks, extending regular SDE and SIL ETL tasks, and extending the repository's layers.

To extend Oracle BI Applications with custom source domains:

1. Define a custom domain in Configuration Manager.

To create a custom domain and associate it with a Fact or Dimension Group, see *About Working With Domains and Domain Mappings*. Note the Domain Code value used. If the Domain Code is not entered correctly at any point, records associated with the domain can cause errors.

While not required, it is a good practice to use a prefix with Domain Codes, such as `X_`, to easily distinguish custom codes from Oracle-supplied Domain Codes, for example, `X_CUSTOM`.



2. Extend data warehouse and staging tables with the new Domain column.

Oracle recommends that the Domain column reflects the Domain Code. However, Domain columns should always have `_CODE` as a suffix, for example, `X_CUSTOM_CODE`.

3. Create custom domain SDE tasks.

If the source domain is extracted from a source table, create an ETL task to extract the domain members and load them into the `W_DOMAIN_MEMBER_GS` table. Follow the regular customization steps for creating a custom SDE task with the following additions:

- a. Create custom interface and package for the target table, W_DOMAIN_MEMBER_GS. Follow these guidelines for the mapping expressions that populate the columns in W_DOMAIN_MEMBER_GS.

Column	Expression	Notes
DOMAIN_CODE	Hard-code the value assigned in Configuration Manager when the domain is registered, for example, X_CUSTOM.	Not applicable.
DOMAIN_TYPE_CODE	'S'	Not applicable.
DOMAIN_MEMBER_CODE		Map to the language independent value that identifies the domain member.
DOMAIN_MEMBER_NAME		Map to the language dependent value that corresponds to the short text name of the domain member.
DOMAIN_MEMBER_DESCRIPTION		Map to the language dependent value that corresponds to the long text description of the domain member. If there is no description type field, map to the same field as Name.
DOMAIN_MEMBER_REFERENCE_CODE	'__NOT_APPLICABLE__'	Not applicable.
LANGUAGE_CODE	DOMAIN_MEMBER_MAPPING('LANGUAGE', <i>OLTP Language Column</i> , #DATASOURCE_NUM_ID, 'W_LANGUAGE')	LANGUAGE is an example of a conformed domain. Map the LANGUAGE column to the appropriate language column in the OLTP. If OLTP table does not support multiple languages, seed using 'BIAPPS.LANGUAGE_BASE'
INTEGRATION_ID	'Domain Code registered in Configuration Manager for this Domain' '~' <i>OLTP Column that identifies the Domain member</i>	Concatenate the domain code with the domain member code, for example, X_CUSTOM.
DATASOURCE_NUM_ID	#DATASOURCE_NUM_ID	Not applicable.

b. Implement the filter expression.

If the OLTP table supports multiple languages, implement the following filter:

```
<OLTP Language Column> IN (#LANGUAGE_LIST)
```

If the OLTP table does not support multiple languages, do not implement a filter on language.

c. Use the IKM BIAPPS Oracle Incremental Update IKM.

d. Add the Custom SDE to the Domain Load Plan Component.

Navigate to the appropriate 3 SDE General Domain Load Plan Component: **Load Plans and Scenarios, BIAPPS Load Plan, SDE, Adaptor Code_version number**, then **3 SDE General Domain version number**.

Add a new Step. Have the step execute the custom Domain SDE's scenario. Populate the Keywords property with DOMAIN=Value assigned in Configuration Manager when the domain is registered.

This step allows the Load Plan Generator to associate the Domain with a particular Fact or Dimension Group. When you generate a Load Plan, you specify the Fact Group to use to the Load Plan Generator. The Load Plan Generator includes any Domain ETL tasks that are stamped with a Domain Keyword that is associated with this Fact Group.

Note: Because the out-of-the-box Load Plan is modified, patches override the customization. You might need to reapply any customizations after load plan patches are applied.

4. Extend regular SDE and SIL ETL tasks to populate the new Domain column.

Follow the regular customization methodology for extending the SDE and SIL ETL tasks to populate the Fact or Dimension table with the following addition.

- In the SDE, set the mapping expression for the Domain column as:

```
DOMAIN_DEFAULT_UNASSIGNED(<OLTP Column>)
```

- In the SIL, set the mapping expression for the Domain column as:

```
- DOMAIN_DEFAULT_NOT_APPLICABLE(<Staging Table Column>)
```

```
- DOMAIN_DEFAULT_NOT_APPLICABLE(X_CUSTOM_CODE)
```

5. Extend the repository's Physical Layer.

Other than exposing the custom Domain column in the Fact or Dimension table, do not make other changes.

6. Extend the repository's Logical Layer.

- a.** Map the physical column to a logical column. Use in any filters or calculations as required.

- b.** Implement LOOKUP () function if MLS required.

- Create a new logical column. For Names, map to "Core"."Lookup - Domain Source"."Domain Member Name" while map Descriptions to "Core"."Lookup - Domain Source"."Domain Member Description"
- Set the Column Source as **Derived from existing columns using an expression.**
- Set the expression as follows (if you are exposing the Name):

```
Lookup(DENSE "Core"."Lookup - Domain Source"."Domain Member Name" ,  
'<Domain Code registered in BIACM>',  
<Logical Column mapped to physical column>,  
VALUEOF(NQ_SESSION."USER_LANGUAGE_CODE") ,  
<Logical column that maps to physical DATASOURCE_NUM_ID column>)
```

7. Extend the repository's Presentation Layer.

Expose the logical Name and Description columns in the Presentation Layer as required.