Oracle® Business Intelligence Applications

Administrator's Guide

11*g* Release 1 (11.1.1.10)

E53675-02

December 2015

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

ORACLE

Oracle Business Intelligence Applications Administrator's Guide, 11g Release 1 (11.1.1.10)

E53675-02

Primary Authors: Emily Kapner, Padma Rao

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

Contents

Preface

Oracle Business Intelligence Applications is comprehensive suite of prebuilt solutions that deliver pervasive intelligence across an organization, empowering users at all levels — from front line operational users to senior management - with the key information they need to maximize effectiveness.

Intuitive and role-based, these solutions transform and integrate data from a range of enterprise sources and corporate data warehouses into actionable insight that enables more effective actions, decisions, and processes.

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

Audience

This information is intended for system administrators and ETL team members who are responsible for managing Oracle BI Applications.

It contains information about ETL customization, domains and localization, Oracle Business Analytics Warehouse naming conventions, and system administration tasks, including setting up and using Oracle GoldenGate and Source-Dependent Schemas to support ETL performance.

Related Documentation

The Oracle Business Intelligence Applications documentation library contains several related documents.

See http://docs.oracle.com/cd/E51479_01/index.htm for a list of related documents.

Conventions

This document uses these text conventions.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New for Oracle Business Intelligence Applications Administration

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

New Features for Oracle BI Applications System Administrators

Oracle Business Intelligence Applications contains these new features for system administrators.

New Features

You can reconcile ledger information available in your Oracle E-Business Suite and Oracle Fusion Applications using Oracle GoldenGate. See Setting up Ledger Correlation using GoldenGate

Documentation Changes

The chapter, *Customizing the Oracle Business Analytics Warehouse*, has moved to the *Oracle Business Intelligence Applications ETL Guide*.

1

About Multi-Language Support

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

Configuring Base and Installed Data Warehouse Languages

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

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

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

Translation Tables

There are two types of translation tables: the Domains translation table and Dimension translation tables. There is a single Domain translation table which holds a translated value in each supported language for a domain. Dimension translation tables are extension tables associated with a given dimension. Depending on certain characteristics of a translatable attribute, it will be found in either the domain or a dimension translation table.

The user's session language is captured in an Oracle BI Enterprise Edition session variable named USER_LANGUAGE_CODE. This is set when users log in from Answers, where they select their preferred language. If users decide to change their preferred language in the middle of a session by using the Administration option to change the current language, this session variable will detect this change. Records

returned from a translation table are filtered to those records with a LANGUAGE_CODE value that matches this session variable.

About Pseudo-Translations

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

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

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

INTEGRATION_ID	NAME	LANGUAGE_CODE	SRC_LANGUAGE_CODE
ABC	Executive	US	US
ABC	Executive	IT	IT
ABC	Executive	SP	US

About Oracle BI Applications Domains

A domain refers to the possible, unique values of a table column in a relational database. In transactional systems, domains are often referred to as list of values (LOVs), which present attribute selections in the user's session language.

The storage of the transaction is independent of the user's language; and, therefore, the field is stored using a language independent identifier. This identifier is typically a character code but can also be a numeric ID. The LOV or domain is then based on an ID-value pair, referred to as a member, and the LOV presents the values in the user's session language. At run time, the IDs are resolved to the value for the user's session language.

In the Oracle Business Analytics Warehouse, the number of unique values in any particular domain is relatively small and can have a low cardinality relative to the dimension it is associated with. For example, the Person dimension may have the domain 'Gender' associated with. The dimension may have millions of records, but the domain will generally have two or three members (M, F and possibly U). In the Oracle Business Analytics Warehouse, the Gender Code is stored in the Person dimension

which acts as a foreign key to the Domains Translation table which stores the translated values. When a query is run, the user-friendly text associated with the code value is returned in the user's session language.

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

Data Model

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

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

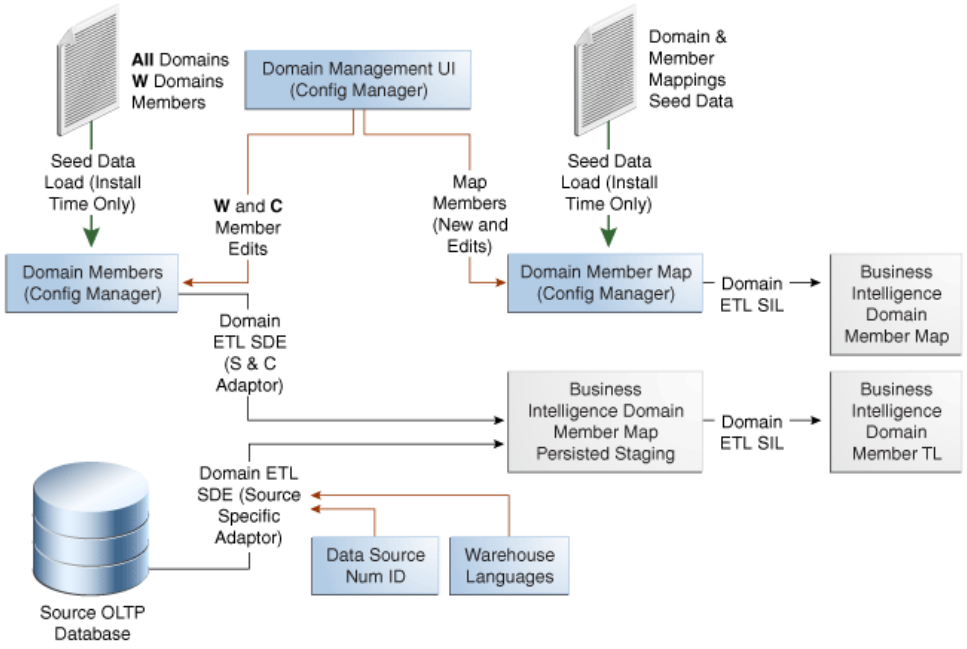
ETL Process

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

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

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

The figure shows an overview of the Oracle BI Applications domain ETL process.



About Oracle BI Applications Domains and Oracle BI Enterprise Edition

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

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

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

About Dimension Translation Tables

Domains are dimensional attributes that have a relatively small number of distinct members, have a low cardinality relative to the number of records in the dimension, and are often used in aggregations. Dimensions have other attributes that require translation that may not fit one or more of these criteria. Dimensions may have translatable attributes that have a high cardinality relative to the dimension or may have a large number of members, and, thus, are not likely candidates for aggregation.

If the domains ETL process was implemented in such cases, performance would be very poor. As a result, these particular attributes are implemented using dimension translation tables.

Data Model

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

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

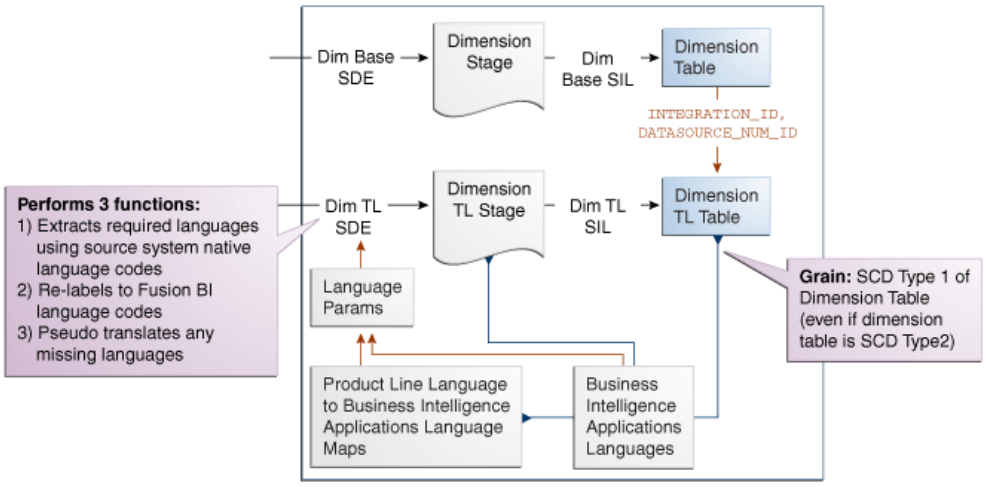
ETL Process

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

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

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

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



Oracle BI Enterprise Edition

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

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

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

Localizing New Domain Members and Oracle BI Repository Metadata

If you added new domain members that require localization or want to add string localizations in the Oracle BI Repository metadata , see About Setting Up Domain Member Mappings.

2

Localizing Oracle Business Intelligence Deployments

Oracle Business Intelligence is designed to allow users to dynamically change their preferred language and locale preferences. You can configure Oracle BI Applications for deployment in one or more language environments other than English.

Topics:

Maintaining Translation Tables Workflow for Oracle BI EE

The Oracle Business Intelligence Presentation layer supports multiple translations for any column name. When working with Oracle BI Answers or rendering a dashboard, users see their local language strings in their reports.

For example, English-speaking and French-speaking users would see their local language strings in their reports. There are two kinds of application strings requiring translation:

- Metadata

  Metadata strings are analytics-created objects in the repository such as subject areas, metrics, and dimensions.

- Presentation Services

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

Topics:

Adding String Localizations for Oracle BI Repository Metadata

If Oracle Business Intelligence data in your deployment is to be viewed in a language other than English, add string localizations in the Oracle BI Repository metadata.

1. Stop the OPMN services.

```
opmnctl stopall
```

2. Open a database administration tool, and connect to the Oracle Business Analytics Warehouse schema.

3. Identify the strings for these presentation objects.

   - Subject area
   - Presentation table
   - Presentation hierarchy
   - Presentation level
   - Presentation column

   For example, for the subject area Payables Invoices - Prepayment Invoice Distributions Real Time, you would enter the following strings:

String	Presentation Object
Payables Invoices - Prepayment Invoice Distributions Real Time	Subject area
Time	Presentation table
Date - Year	Presentation hierarchy
Total	Presentation level
Year	Presentation level
Calendar Year	Presentation column

4. For each subject area, externalize the strings for localization and generate custom names for the presentation objects.

   a. In the Oracle BI Administration Tool, right-click the subject area and select **Externalize Display Names**, and then select **Generate Custom Names**.

   b. Save your work.

      For more information about localizing strings, see Localizing Metadata Names in the Repository, *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

5. Check the consistency of the repository, and remove any inconsistencies.

   For instructions, see Checking the Consistency of a Repository or a Business Model.

6. Enter the custom name of one of the presentation objects into the table C_RPD_MSG.

```
INSERT INTO C_RPD_MSGS(MSG_ID, CREATED_BY, CREATION_DATE)
VALUES('<CUSTOM NAME OF PRESENTATION OBJECT>', 'CUSTOM', SYSTIMESTAMP);
COMMIT;
```

Note: To view the values for custom names and logical columns in the Administration Tool, right-click the presentation object and select **Properties**. The data in the **Custom display name** field appears in the format VALUEOF(NQ_SESSION.VALUE, where VALUE is the custom name for a presentation object, or the logical value for a presentation column. This value is the value that you need to enter in the VALUES section of the SQL statement.

7. Enter the localized string for the presentation object in the previous step into the table C_RPD_MSGS_TL.

```
INSERT INTO C_RPD_MSGS_TL(MSG_ID, MSG_TEXT, LANGUAGE_CODE, CREATED_BY,
CREATION_DATE)
VALUES('<CUSTOM NAME OF PRESENTATION OBJECT>', '<LOCALIZATION OF THE STRING'>,
'<LANGUAGE CODE FOR TRANSLATED LANGUAGE>', 'CUSTOM', SYSTIMESTAMP);
COMMIT;
```

   To identify the language code for a particular language, use the following SQL:

```
SELECT LANGUAGE_CODE, NLS_LANGUAGE, NLS_TERRITORY
FROM FND_LANGUAGES_B
WHERE INSTALLED_FLAG IN ('B', 'I');
```

8. Enter additional details about the presentation object into the table C_RPD_MSGS_REL.

```
INSERT INTO C_RPD_MSGS_REL(MSG_ID, MSG_NUM, MESSAGE_TYPE, CREATED_BY,
CREATION_DATE)
VALUES('<CUSTOM NAME OF PRESENTATION OBJECT>', '<TRANSLATION OF THE STRING'>,
'<LANGUAGE CODE FOR TRANSLATED LANGUAGE>', 'METADATA','CUSTOM', SYSTIMESTAMP);
COMMIT;
```

9. Repeat steps 6 through 8 for each presentation object requiring localization.

10. Validate that the physical connection of the session initialization block INIT_USER_LANGUAGE_CODE is operable

    a. In the Oracle BI Administration Tool, select **Manage**, **Variables**, then **Session Initialization Block**.

    b. Right-click **INIT_USER_LANGUAGE_CODE**.

    c. In the Properties dialog, click **Edit Data Source**.

    d. Click **Test**, and input the value for the language code, then click **OK**.

       For example, for Arabic enter 'AR'.

       The value USER_LANGUAGE_CODE = '<language code>' should be returned.

       If this value is not returned, the TNS entry for the data source is not properly configured.

11. Restart the OPMN services.

12. Verify the localized strings in Oracle BI Answers. On the login page, specify the appropriate language.

About Translating Presentation Services Strings

The translations for such Presentation Services objects as report and page names are copied to this location during the BI Applications installation process: `ORACLE_HOME/bifoundation/OracleBIPresentationServicesComponent/coreapplication_obips1/msgdb/l_`*`language_abbreviation`*`/Captions`. In multiple language deployment mode, if you add any additional Presentation Services objects, such as reports and new dashboard pages, you also need to add the appropriate translations.

Add these translations using the Catalog Manager tool. For more information on using this utility, see Configuring and Managing the Oracle BI Presentation Catalog, *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

Changing the Default Currency in Analytics Applications

In Oracle Business Intelligence Applications, you might see a dollar sign used as the default symbol when amounts of money are displayed.

To change this behavior, you must edit the `currencies.xml` file. The `currencies.xml` file is located in the following directories:

- Windows: `ORACLE_HOME\bifoundation\web\display\currencies.xml`
- UNIX: `ORACLE_HOME/bifoundation/web/display/currencies.xml`

To change the default currency in Analytics Applications:

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

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

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

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

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

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

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

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

3

Oracle Business Analytics Warehouse Naming Conventions

Oracle Business Analytics Warehouse contains these types of tables and columns. Be sure to follow the specified naming conventions.

Note:

This information does not apply to objects in the Oracle Business Intelligence repository.

Topics:

Naming Conventions for Oracle Business Analytics Warehouse Tables

Oracle Business Analytics Warehouse tables use a three-part naming convention: PREFIX_NAME_SUFFIX.

Part	Meaning	Table Type
PREFIX	Shows Oracle Business Analytics Warehouse-specific data warehouse application tables.	W_ = Warehouse
NAME	Unique table name.	All tables.

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

Table Types for Oracle Business Analytics Warehouse

This table lists the types of tables used in the Oracle Business Analytics Warehouse.

Table Type	Description
Aggregate tables (_A)	Contain summed (aggregated) data.
Dimension tables (_D)	Star analysis dimensions.
Delete tables (_DEL)	Tables that store IDs of the entities that were physically deleted from the source system and should be flagged as deleted from the data warehouse. Note that there are two types of delete tables: _DEL and _PE. For more information about the _PE table type, see the row for Primary extract tables (_PE) in this table.
Dimension Hierarchy tables (_DH)	Tables that store the dimension's hierarchical structure.

Table Type	Description
Dimension Helper tables (_DHL)	Tables that store many-to-many relationships between two joining dimension tables.
Staging tables for Dimension Helper (_DHLS)	Staging tables for storing many-to-many relationships between two joining dimension tables.
Dimension Hierarchy Staging table (_DHS)	Staging tables for storing the hierarchy structures of dimensions that have not been through the final extract-transform-load (ETL) transformations.
Dimension Staging tables (_DS)	Tables used to hold information about dimensions that have not been through the final ETL transformations.
Fact tables (_F)	Contain the metrics being analyzed by dimensions.
Fact Staging tables (_FS)	Staging tables used to hold the metrics being analyzed by dimensions that have not been through the final ETL transformations.
Internal tables (_G, _GS)	General tables used to support ETL processing.
Helper tables (_H)	Inserted between the fact and dimension tables to support a many-to-many relationship between fact and dimension records.
Helper Staging tables (_HS)	Tables used to hold information about helper tables that have not been through the final ETL transformations.
Mini dimension tables (_MD)	Include combinations of the most queried attributes of their parent dimensions. The database joins these small tables to the fact tables.
Primary extract tables (_PE)	Tables used to support the soft delete feature. The table includes all the primary key columns (integration ID column) from the source system. When a delete event happens, the full extract from the source compares the data previously extracted in the primary extract table to determine if a physical deletion was done in the Siebel application. The soft delete feature is disabled by default. Therefore, the primary extract tables are not populated until you enable the soft delete feature. Note that there are two types of delete tables: _DEL and _PE. For more information about the _DEL table type, see the row for Delete table (_DEL) in this table.
Persisted Staging table (_PS)	Tables that source multiple data extracts from the same source table. These tables perform some common transformations required by multiple target objects. They also simplify the source object to a form that is consumable by the warehouse needed for multiple target objects. These tables are never truncated during the life of the data warehouse. These are truncated only during full load, and therefore, persist the data throughout.

Table Type	Description
Row Flattened Hierarchy Table (_RH)	Tables that record a node in the hierarchy by a set of ancestor-child relationships (parent-child for all parent levels).
Translation Staging tables (_TL)	Tables store names and descriptions in the languages supported by Oracle BI Applications.
Pre-staging or post-staging Temporary table (_TMP)	Source-specific tables used as part of the ETL processes to conform the data to fit the universal staging tables (table types_DS and _FS). These tables contain intermediate results that are created as part of the conforming process.
Unbounded dimension (_UD)	Tables containing information that is not bounded in transactional database data but should be treated as bounded data in the Oracle Business Analytics Warehouse.
Staging tables for Usage Accelerator (_WS)	Tables containing the necessary columns for the ETL transformations.

Aggregate Tables in Oracle Business Analytics Warehouse

One of the main uses of a data warehouse is to sum up fact data with respect to a given dimension, for example, by date or by sales region. Performing this summation on-demand is resource-intensive, and slows down response time.

Oracle Business Analytics Warehouse precalculates some of these sums and stores the information in aggregate tables. In the Oracle Business Analytics Warehouse, the aggregate tables have been suffixed with _A.

Dimension Class Tables in Oracle Business Analytics Warehouse

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

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

Dimension Tables in Oracle Business Analytics Warehouse

The unique numeric key (ROW_WID) for each dimension table is generated during the load process. This key is used to join each dimension table with its corresponding fact table or tables. It is also used to join the dimension with any associated hierarchy table or extension table. The ROW_WID columns in the Oracle Business Analytics Warehouse tables are numeric.

In every dimension table, the ROW_WID value of zero is reserved for Unspecified. If one or more dimensions for a given record in a fact table is unspecified, the corresponding key fields in that record are set to zero.

Dimension Tables With Business Role-Based Flags

This design approach is used when the entity is logically the same but participates as different roles in the business process.

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

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

Fact Tables in Oracle Business Analytics Warehouse

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

Helper Tables in Oracle Business Analytics Warehouse

Helper tables are used to solve complex problems that cannot be resolved by simple dimensional schemas.

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

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

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

Hierarchy Tables in Oracle Business Analytics Warehouse

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

There are two types of hierarchies: a structured hierarchy in which there are fixed levels, and a hierarchy with parent-child relationships. Structured hierarchies are simple to model, since each child has a fixed number of parents and a child cannot be

a parent. The second hierarchy, with unstructured parent-child relationships is difficult to model because each child record can potentially be a parent and the number of levels of parent-child relationships is not fixed. Hierarchy tables have a suffix of _DH.

Mini-Dimension Tables in Oracle Business Analytics Warehouse

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

The table lists the mini-dimension tables in the Oracle Business Analytics Warehouse.

Table Name	Parent Dimension
W_RESPONSE_MD	Parent W_RESPONSE_D
W_AGREE_MD	Parent W_AGREE_D
W_ASSET_MD	Parent W_ASSET_D
W_OPTY_MD	Parent W_OPTY_D
W_ORDER_MD	Parent W_ORDER_D
W_QUOTE_MD	Parent W_QUOTE_D
W_SRVREQ_MD	Parent W_SRVREQ_D

Staging Tables in Oracle Business Analytics Warehouse

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

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

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

Translation Tables in Oracle Business Analytics Warehouse

Translation tables provide multi-language support by storing names and descriptions in each language that Oracle Business Analytics Warehouse supports.

There are two types of translation tables:

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

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

Internal Tables in Oracle Business Analytics Warehouse

Internal tables are used primarily by ETL mappings for data transformation and for controlling ETL runs. These tables are not queried by end users.

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

Standard Column Prefixes in Oracle Business Analytics Warehouse

The Oracle Business Analytics Warehouse uses a standard prefix to indicate fields that must contain specific values.

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

Standard Column Suffixes in Oracle Business Analytics Warehouse

The Oracle Business Analytics Warehouse uses suffixes to indicate fields that must contain specific values.

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

System Columns in Oracle Business Analytics Warehouse Tables

Oracle Business Analytics Warehouse tables contain system fields. These system fields are populated automatically and should not be modified by the user.

The table lists the system columns used in data warehouse dimension tables.

System Column	Description
ROW_WID	Surrogate key to identify a record uniquely.
CREATED_BY_WID	Foreign key to the W_USER_D dimension that specifies the user who created the record in the source system.

System Column	Description
CHANGED_BY_WID	Foreign key to the W_USER_D dimension that specifies the user who last modified the record in the source system.
CREATED_ON_DT	The date and time when the record was initially created in the source system.
CHANGED_ON_DT	The date and time when the record was last modified in the source system.
AUX1_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX2_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX3_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX4_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
DELETE_FLG	This flag indicates the deletion status of the record in the source system. A value of Y indicates the record is deleted from the source system and logically deleted from the data warehouse. A value of N indicates that the record is active.
W_INSERT_DT	Stores the date on which the record was inserted in the data warehouse table.
W_UPDATE_DT	Stores the date on which the record was last updated in the data warehouse table.
DATASOURCE_NUM_ID	Unique identifier of the source system from which data was extracted. In order to be able to trace the data back to its source, it is recommended that you define separate unique source IDs for each of your different source instances.
ETL_PROC_WID	System field. This column is the unique identifier for the specific ETL process used to create or update this data.
INTEGRATION_ID	Unique identifier of a dimension or fact entity in its source system. In case of composite keys, the value in this column can consist of concatenated parts.
TENANT_ID	Unique identifier for a tenant in a multi-tenant environment. This column is typically be used in an Application Service Provider (ASP)/Software as a Service (SaaS) model.
X_CUSTOM	Column used as a generic field for customer extensions.

System Column	Description
CURRENT_FLG	This is a flag for marking dimension records as "Y" in order to represent the current state of a dimension entity. This flag is typically critical for Type II slowly changing dimensions, as records in a Type II situation tend to be numerous.
EFFECTIVE_FROM_DT	This column stores the date from which the dimension record is effective. A value is either assigned by Oracle BI Applications or extracted from the source.
EFFECTIVE_TO_DT	This column stores the date up to which the dimension record is effective. A value is either assigned by Oracle BI Applications or extracted from the source.
SRC_EFF_FROM_DT	This column stores the date from which the source record (in the Source system) is effective. The value is extracted from the source (whenever available).
STC_EFF_TO_DT	This column stores the date up to which the source record (in the Source system) is effective. The value is extracted from the source (whenever available).

Multi-Currency Support for System Columns

The table lists the currency codes and rates for related system columns.

System Column	Description
DOC_CURR_CODE	Code for the currency in which the document was created in the source system.
LOC_CURR_CODE	Usually the reporting currency code for the financial company in which the document was created.
GRP_CURR_CODE	The primary group reporting currency code for the group of companies or organizations in which the document was created.
LOC_EXCHANGE_RATE	Currency conversion rate from the document currency code to the local currency code.
GLOBAL1_EXCHANGE_RATE	Currency conversion rate from the document currency code to the Global1 currency code.
GLOBAL2_EXCHANGE_RATE	Currency conversion rate from the document currency code to the GLOBAL2 currency code.
GLOBAL3_EXCHANGE_RATE	Currency conversion rate from document currency code to the GLOBAL3 currency code.
PROJ_CURR_CODE	Code used in Project Analytics that corresponds to the project currency in the OLTP system.

Oracle Business Analytics Data Warehouse Primary Data Values

It is possible for various dimensions to have one-to-many and many-to-many relationships with each other. These kinds of relationships can introduce problems in analyses.

For example, an Opportunity can be associated with many Sales Representatives and a Sales Representative can be associated with many Opportunities. If your analysis includes both Opportunities and Sales Representatives, a count of Opportunities would not be accurate because the same Opportunity would be counted for each Sales Representative with which it is associated.

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

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

About Multi-Language Support in Oracle Business Analytics Warehouse

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

For more information about multi-language support, see About Multi-Language Support .

Oracle Business Analytics Warehouse Currency Preferences

Configure global currencies in Functional Setup Manager.

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

The Oracle Business Analytics Warehouse supports the following currency preferences.

- Contract currency — The currency used to define the contract amount. This currency is used only in Project Analytics.
- CRM currency — The CRM corporate currency as defined in the Fusion CRM application. This currency is used only in CRM Analytics applications.
- Document currency — The currency in which the transaction was done and the related document created.
- Global currency — The Oracle Business Analytics Warehouse stores up to three group currencies. These need to be pre-configured so as to allow global reporting

by the different currencies. The exchange rates are stored in the table W_EXCH_RATE_G.

- Local currency — The accounting currency of the legal entity in which the transaction occurred.
- Project currency — The currency in which the project is managed. This may be different from the functional currency. This applies only to Project Analytics.

4

Oracle Business Intelligence Endeca Discovery Option

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

Topics:

Setting Up Endeca Integration with Oracle BI Applications

Set up Oracle Endeca with Oracle BI Applications.

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

1. Install Oracle Endeca Server 7.6.0.

   For information about requirements and installation of the Endeca Server, refer to the *Oracle Endeca Server Installation Guide*. Install all available patches.

2. Install Endeca 3.1 Studio.

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

3. In Oracle Data Integrator (ODI) Studio, set up the ODI topology to include Oracle Business Intelligence Enterprise Edition as a source, the Endeca Server as a target, and access to required, as described in Setup Step: Set Up ODI Connectivity for Oracle BI and Endeca.

4. Load data by running ODI scenarios individually or by running the ODI load plan, as described in Setup Step: Load Data.

5. Create a new Endeca Studio Application and deploy the Sample Application, as described in Setup Step: Create a New Endeca Application.

Tasks for Setting Up Oracle Endeca with Oracle BI Applications

Perform these detailed tasks for setting up Oracle Endeca with Oracle BI Applications.

Note: You must perform the tasks in this section in the sequence described in Setting Up Endeca Integration with Oracle BI Applications.

Topics:

Setup Step: Install Oracle Endeca Server and Endeca Studio

Install all available patches. Optionally, you can take some steps to optimize the Endeca Server.

Recommended Data Size

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

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

Creating Endeca Data Domain Profile to Optimize CPU Core Count

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

After the new Data Domain Profile is created, set the new profile name to the ODI parameter ENDECA_DATADOMAIN_PROFILE when running the Endeca Load Plan or scenarios. For more information about running the Endeca Load Plan and scenarion, refer to Setup Step: Load Data.

Disabling, Enabling, or Deleting Data Domains

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

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

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

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

Install Endeca 3.1 Studio

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

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

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

Creating an Oracle BI Enterprise Edition Source

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

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

   Note:

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

6. In the BI Applications data server, create a new Physical Schema.

   In its Definition tab, specify Core for the Catalog (Catalog) and for Catalog (Work Catalog).

7. In ODI Designer's Topology tab, expand the Logical Architecture and assign the OBI_BIAPPS11G logical schema to the physical schema you created.

Creating the Endeca Server Target

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

1. In ODI Designer's Topology tab, expand the Endeca Server Technology in the Physical Architecture.
2. In the Endeca Server Technology, create a new Endeca data server.
3. In the Definition tab of the new server, enter a name.

   In the Connection details, leave the User and Password fields empty.
4. In the JDBC tab, leave the JDBC Driver empty.
5. Enter `http://IP address or the Endeca Server hostname:7001/endeca-server` as the JDBC URL.

   Note:

   Ensure that there are no leading or trailing spaces in the JDBC URL.
6. In the Endeca data server, create a new Physical Schema.

   In its Definition tab, you can leave the Endeca Datastore schema and Endeca Datastore work schema as undefined.
7. In ODI Designer's Topology tab, expand the Logical Architecture and assign the OEID_BIAPPS11G logical schema to the physical schema you created.

Creating the Data Server for Endeca Files

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

1. In ODI Designer's Topology tab, expand the File Technology in the Physical Architecture.
2. In the File Technology, create a new file data server.
3. In the Definition tab of the new server, enter a name.

   In the Connection details, leave the User and Password fields empty.
4. In the JDBC tab, enter `com.sunopsis.jdbc.driver.file.FileDriver` as the JDBC Driver.
5. Enter `jdbc:snps:dbfile?OPT=TRUE` as the JDBC URL.

   Note:

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

6. In the file data server, create a new Physical Schema.

   Use the directory containing the view XML files (for example, `C:\Temp`) for Directory (Schema) and Directory (Work Schema). This directory should be relative to the server where the ODI Agent is installed and can access the view XML files.

   Note:

   The Sample Apps View XMLs can be found in the BIAPPS installation under *MW_ Home*\Oracle_BI1\biapps\admin\provisioning\endeca \OracleBIApps_Endeca.zip.

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

Exporting Views Using Endeca Web Service Procedure

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

1. Create a new ODI package and add the Endeca Web Service Procedure.

   Give the package a name, for example Export Views.

2. Set the following Procedure options.

   - ENDECA_WS_RELATIVE_PATH: /ws/sconfig/<*Data Domain Name*>
   - REQUEST_FILE_NAME: Export_Views.xml
   - RESPONSE_FILE_NAME: Name of the response message file views export. The naming convention is <*Data Domain Name*>_Views.xml.

3. Place or save the E`xport_Views.xml` file into the directory entered in the BIAPPS_OEID_FILE physical schema.

   This is also the directory where the procedure outputs the response file (RESPONSE_FILE_NAME).

4. Run the ODI Package. If successful, the views are exported into the response file.

Importing Views Using Endeca Web Service Procedure

Exported views XML requires formatting before it can be imported or re-imported.

1. Open the XML in a text editor and search for `validatedSemanticEntity` and replace it with `semanticEntity`.

2. Search for `listEntitiesResponse` and replace it with `putEntities`.

3. Use a new or existing ODI package to load the Endeca Data Domain.

   In the package, add the Endeca Web Service Procedure as the final step.

4. Set the following Procedure options.
   - ENDECA_WS_RELATIVE_PATH: `/ws/sconfig/Data Domain Name`
   - REQUEST_FILE_NAME: Name of the response message file from the views export. The naming convention is `Data Domain Name_Views.xml`.
5. Place or save the `Export_Views.xml` file into the directory entered in the BIAPPS_OEID_FILE physical schema.

   This is also the directory where the procedure outputs the response file (RESPONSE_FILE_NAME).
6. Run the ODI Package. If successful, the views are imported into the target data domain.

Verifying Connection to the Data Warehouse

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

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

Configuring the ODI Agent

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

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

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

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

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

Disabling Schemas in the Load Target Schema Interface

Endeca Server schema metadata can be modified using Endeca Studio rather than editing ODI flexfields after the schema is created during the initial Endeca ETL. To

prevent overwriting a customized schema, you set the APPLY_SCHEMA option in the Load Target Schema Interface to FALSE.

1. In ODI Designer, under Projects, open the Package of the subject area and select the Diagram tab.
2. Right-click the **TLP_OEID_*Subject Area Name*Load_Tgt_Schema** step and select **Edit Linked Object**.

   The Load Target Schema Interface opens in a new tab.
3. In the Flow tab, click the Target (**BIAPPS_OEID**).

   Change the APPLY_SCHEMA option in the Properties Inspector options area to **false**.
4. Save and close the interface, then the package.
5. Generate the scenario.

   In Projects, right-click the Package and select **Generate Scenario**, and click **OK** to accept the defaults.
6. Note the version of the new scenario.

You can now run the scenario directly using the noted version number, selecting the Context, Logical Agent, and Log Level. Alternately, you can run the scenario from the Endeca load plan. For each of the scenario steps, the load plan sets the version -1, which uses the latest generated scenario. Scenarios are run directly in ODI Designer.

Setup Step: Load Data

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

Setting and Declaring Variables

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

- ENDECA_DATADOMAIN_PROFILE: This variable is the name of the Data Domain Profile if one was created. For information about creating a Data Domain Profile to maximize performance, refer to Creating Endeca Data Domain Profile to Optimize CPU Core Count.
- ENDECA_OBIEE_SQL_PREFIX: This variable is used to override the default preferred currency, which is GLOBAL1.
- ETL_PREDICATE_EXTRACT: This variable is used to enter filters.

The following ODI variables require modifications to the ODI package.

- ENDECA_DATADOMAIN
- ENDECA_DATALOAD_LOG_FILE_PATH

To change the ENDECA_DATADOMAIN and/or ENDECA_DATALOAD_LOG_FILE_PATH values:

1. In the Designer under Projects, open the package of the subject area and select the Diagram tab.
2. Select the ENDECA_DATADOMAIN or ENDECA_DATALOAD_LOG_FILE_PATH variable..

   In the Properties pane, in the General tab, enter the desired value
3. Save and close the Package.
4. Save and close the interface, then the package.
5. Generate the scenario.

   In Projects, right-click the **Package** and select **Generate Scenario**, and click **OK** to accept the defaults. Select Startup Parameters **Use All** when prompted, then click **OK**.

Run Scenarios Directly

Run the ODI scenarios directly.

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

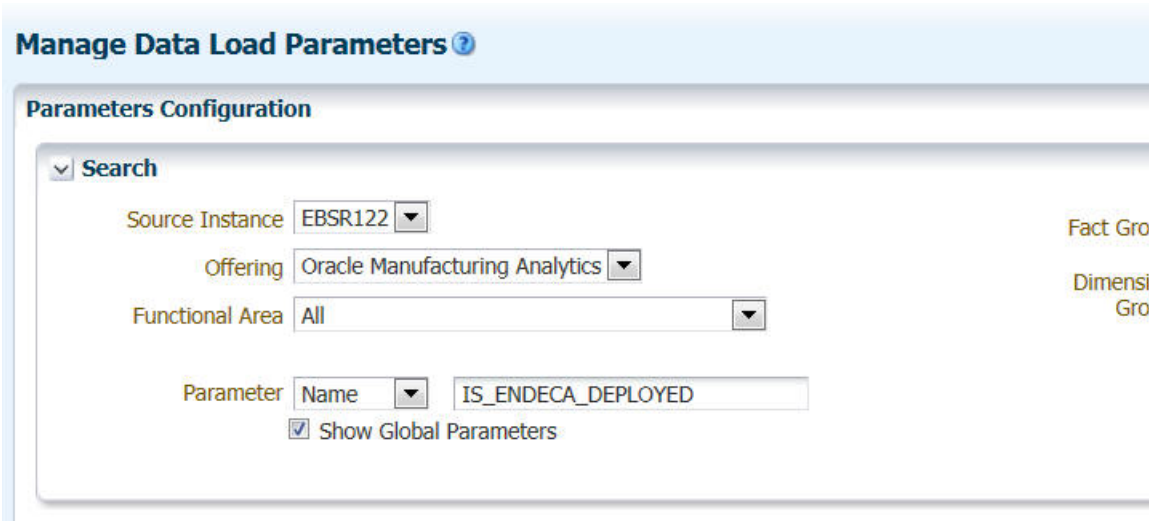
Run the Endeca Load Plan

A predefined load plan is provided to run the Endeca packages. It contains steps to evaluate if the IS_ENDECA_DEPLOYED Configuration Manager parameter is assigned to each Subject Area, for example, derived Fact Groups created by development teams. To select which scenarios are run, set the IS_ENDECA_DEPLOYED variable to Yes, then run the load plan.

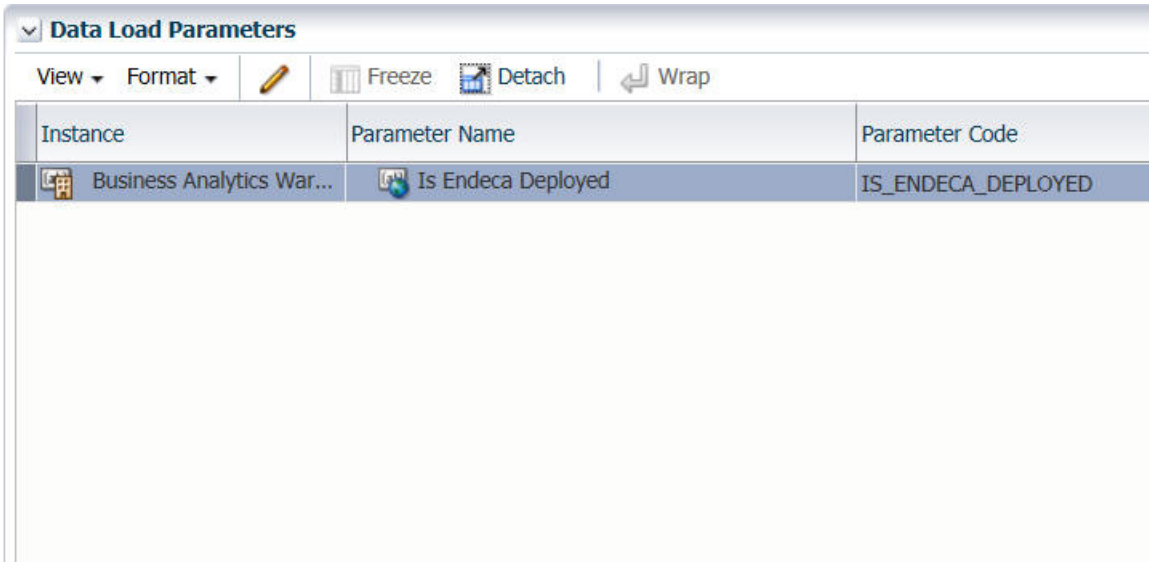
Complete the system setup (register sources, enable offerings, and so on) before performing any domain/parameter configuration, as described in Performing Post-Installation System Setup Tasks.

The frequency with which you run the load plan for data refresh is dependent on your use. Depending on the cost of a typical load, nightly execution may be appropriate, but if you do not require real-time data, you may run it less frequently. In the case that you are performing only historical analysis, the first load may be the only required.

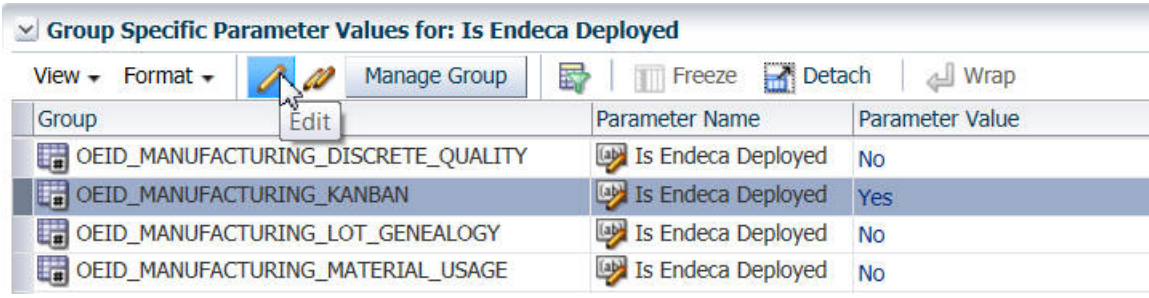
1. Log in to Configuration Manager.
2. Select **Data Load Parameters Administration**, then **Manage Data Load Parameters**.
3. Under the Search panel, select Source Instance, Offering, the Parameter `IS_ENDECA_DEPLOYED`, then click **Search**.



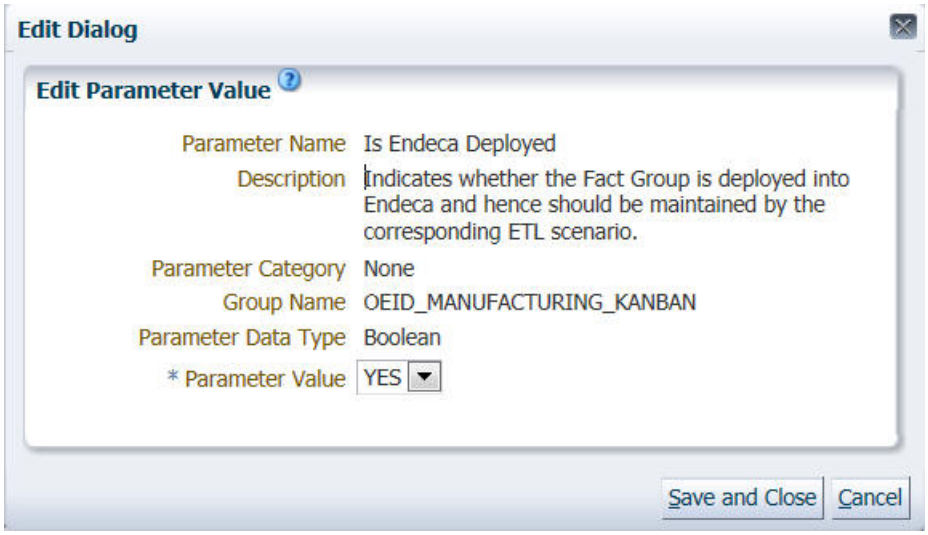
4. Under the Data Load Parameter panel, select the **Is Endeca Deployed** parameter.



5. Under the Group Specific Parameter Values panel, select the Endeca Fact Group (prefixed with OEID) and click **Edit**.



6. Change Parameter Value to **Yes**, then **Save and Close**.



Setting Filters and Currency Setting

Within the predefined Endeca load plan, variables such as ENDECA_OBIEE_SQL_PREFIX and ETL_PREDICATE_EXTRACT may be overridden and saved in the load plan.

Filter and currency modifications can be done directly in the load plan under Scenario Variables by selecting the scenario step, and checking and providing a new value.

Setup Step: Create a New Endeca Application

Create a new Endeca application.

1. Log in to Endeca Studio as an Administrator.
2. Click the gear icon and, in the Control Panel, select **Endeca Servers**.
3. Click **New Connection** and enter the required parameters.
4. Validate the connection and, optionally, test it.
5. Click **Back to Home** to return to the home page.
6. Click **New Application**.
7. Enter an Application name and Application description.
8. Select a pre-built Endeca Server under **Select a Data Source**.
9. Select a managed data connection.
10. Click **Done**.

A default template is created.

ODI Package, Interface, and Procedure Design Overview

Learn about the Endeca ODI package, interface, and procedures. Source and target data stores delivered with the BI Applications installation include all Oracle BI

Applications repository (RPD) logical columns. These may be customized to add or remove mappings to meet business requirements.

New Flexfields

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

Limitations

Some complex metadata from the RPD BMM layer cannot be extracted and loaded into Endeca Server, which includes:

- The dimension
- Aggregation rules
- Complex expressions
- Multiple Logical Table source
- Content of Logical source

The reports with same logical columns in OBIEE and Endeca Studio have different query results because of the missing metadata. You can use Endeca EQL to imitate the features of the OBIEE model.

Packages

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

- ENDECA_DATADOMAIN variable: Name of SA Data Domain, for example, OEID_Project_Cost.
- ENDECA_DATADOMAIN_PROFILE variable: Data Domain profile. If blank, the default Data Domain profile is used.
- ENDECA_DATALOAD_LOG_FILE_PATH variable: Path/name for Log File, for example OEID_Project_Cost_ENDECA_DATALOAD.LOG.
- ENDECA_OBIEE_SQL_PREFIX variable: Optional Oracle BI EE prefix, usually set for currency, for example, set variable PREFERRED_CURRENCY='Global Currency 4'.
- ETL_PREDICATE_EXTRACT variable: Optional Oracle BI EE where clause (filter) condition. The default is 1=1.
- TLP_OEID_*SA*_Load_Tgt_Schema interface, for example, TLP_OEID_Project_Cost_Load_Tgt_Schema
  - Interface does not load data (has filter set to 1=2).
  - Interface creates the Data Domain if it does not already exist.
  - Mapping contains all of the columns for the Subject Area and loads only the schema.

Caution:

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

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

- TLP_OEID_*SA_Fact Table Name* interfaces, for example, TLP_OEID_Project_Cost_Fact_Project_Cost.
  - Interface or interfaces (some Subject Areas may have more than one) extracts data from Oracle BI EE and loads Endeca.
  - One interface exists per dataset collection name, for example, "Fact_Project_Cost"
  - Each interface loads a single denormalized fact star schema.
  - Each interface truncates the collection data before each load, so each load is a destructive load and not incremental.

Each interface uses the following:

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

Data Stores

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

Endeca Web Service Procedure

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

Deploying Sample Applications

You can deploy Endeca sample applications, which can be downloaded from the BI Applications 11.1.1.8.1 Media Pack from Oracle Software Delivery Cloud, and are used as a basis to create your own applications.

There are eight sample applications, with Data Domains and Studio Applications (.lar files) delivered in the Media Pack under Supporting Files for Endeca Sample Applications for BI Applications 11.1.1.8.1, and View XML files deployed by the BI Applications installer. Notice that some sample applications contain tag clouds based on enriched text, so if the Data Domain is reloaded, rules need to be re-run. If the Data Domain is reset, rule need to be recreated.

Importing Sample Data Domains

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

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

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

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

```
CALL endeca-cmd import-dd OEID_Sales_Order_Lines \--offline-name
OEID_Sales_Order_Lines
CALL endeca-cmd enable-dd OEID_Sales_Order_Lines
CALL endeca-cmd import-dd OEID_Project_Performance \--offline-name
OEID_Project_Performance
CALL endeca-cmd enable-dd OEID_Project_Performance
CALL endeca-cmd import-dd OEID_SIA_Admissions_and_Recruiting_Student_Response \--
offline-name OEID_SIA_Admissions_and_Recruiting_Student_Response
CALL endeca-cmd enable-dd OEID_SIA_Admissions_and_Recruiting_Student_Response
CALL endeca-cmd import-dd OEID_SIA_Admissions_and_Recruiting_Application_Evaluation
\--offline-name OEID_SIA_Admissions_and_Recruiting_Application_Evaluation
CALL endeca-cmd enable-dd OEID_SIA_Admissions_and_Recruiting_Application_Evaluation
CALL endeca-cmd import-dd OEID_Employee_Expenses_Overview \--offline-name
OEID_Employee_Expenses_Overview
CALL endeca-cmd enable-dd OEID_Employee_Expenses_Overview
CALL endeca-cmd import-dd OEID_Manufacturing_Work_Order_Performance \--offline-name
OEID_Manufacturing_Work_Order_Performance
CALL endeca-cmd enable-dd OEID_Manufacturing_Work_Order_Performance
CALL endeca-cmd import-dd OEID_Human_Resources_Recruitment \--offline-name
OEID_Human_Resources_Recruitment
CALL endeca-cmd enable-dd OEID_Human_Resources_Recruitment
CALL endeca-cmd import-dd OEID_Manufacturing_Actual_Production \--offline-name
OEID_Manufacturing_Actual_Production
CALL endeca-cmd enable-dd OEID_Manufacturing_Actual_Production
```

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

Refinement Rules

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

Importing Endeca Studio Application Sample Applications

Import Endeca Studio sample applications.

You are required to have created a new application prior to import.

To create a new application using the default template:

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

   Below is an example of typical parameters:

```
Connection ID: OEID_Employee_Expenses_Overview

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

   Note:

   If applying the custom OBIA Studio security manager for the sample apps, the Connection ID needs to match the Data Domain name to be in sync with the Connection ID specified in the delivered security columns configuration file. For more information about the security columns configuration file, refer to Applying the Custom BI Applications Security Manager.

4. Validate the connection, then click **Save**.

   Optionally, confirm connectivity using the **Test Connection** button.
5. Click the **Back to Home** button to return to the home page.
6. Click the **New Application** button.
7. Provide an Application name, for example, Employee Expenses, and an Application description.
8. Under **Select a Data** Source, select **Use a Pre-built Endeca Server**.
9. Select a managed data connection.

   For example, OEID_Employee_Expenses_Overview.
10. Click **Done** to create a default template.

Importing the Sample Application

Import a sample application (.lar file) provided in the Media Pack.

1. On the line of the application, select **Actions**, then **Manage** Pages.
2.
3. Under All Pages, in the Export/Import tab, select the **Import** tab.
4. Click **Browse** or **Choose File**, and select the appropriate .lar file.

5. Under the section titled What would you like to import, leave the defaults.

   To delete any pages in the destination environment that do not exist in the ,lar file, check the Delete Missing Pages checkbox. To ensure that the import includes all of the component configuration, check the User Preferences checkbox.

6. Click **Import**.

Recreating the Enrichment Pipelines

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

1. In **Application Settings**, **Data Sets**, then **Enrichments**, click **Add Enrichment**, extract terms, and select the attribute or attributes containing free form text (unstructured content) to be enriched.

2. Provide an output name, for example Attribute Name (terms).

   The rest of the settings can remain defaults.

3. Click the **Run** button to process the enrichment.

4. Edit the tag clouds settings to replace the invalid output attributes with the new ones.

Rerunning the Enrichment

Whenever data is replaced or reloaded into the Endeca Data Domain, the tag clouds and enrichment pipelines should still work and do not need to be recreated, but the enrichment pipeline needs to be run again, which recreates the enriched data. Otherwise, the enriched tag clouds do not show any data.

1. Navigate to Application Settings > Data Sets and select the datasets which have enrichment pipelines.

2. Select the **Enrichment** tab.

3. Click the **Run** button to process the enrichment.

Applying the Custom BI Applications Security Manager

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

Assigning BIImpersonator Role to an OBIEE user

The Oracle BI Applications Security Manager requires an Oracle BI Enterprise Edition user with administrator and impersonator roles, which is used to obtain information necessary to apply security filters in an Endeca Application.

If you don't have an impersonator user defined, you must create a user in the security realm associated with OBIEE and assign the user a name, such as `BIImpersonator`. In the Enterprise Manager tool launched on the WebLogic Admin server associated with OBIEE, give this user the permission of type `oracle.security.jps.permission` for the resource called `oracle.bi.server.impersonateUser`. Find detailed instructions in Credentials for Connecting to the Oracle BI Presentation Catalog.

To add the impersonator role to an existing Oracle BI EE administrator account:

1. Log in to Oracle Business Intelligence's Enterprise Manager with Administrator privileges.
2. Expand Business Intelligence in the left-hand pane.
3. Right-click **coreapplication** and select **Security**, then **Application Roles** to navigate to the Application Roles page.

   By default, the obi application stripe is selected and the default application roles are displayed.
4. Search for the Role Names with a prefix of **BI**.
5. In the Members list, click the **BIImpersonator** role, then click **Edit**.
6. Click **Add**.
7. In the Add Principal dialog box, search for **Type of** User and locate an administrative user.

Creating and Setting Up the Credential Store in Endeca Studio

The Oracle BI Enterprise Edition account credentials with administrator and impersonator roles is saved locally in the Endeca Studio Domain's Credential Store. The BI Applications Security Manager obtains the password information from the Credential Store to connect to Oracle BI Enterprise Edition using JDBC.

Endeca Information Discovery Studio Documentation

Complete doc set

To set up the credential store, follow the steps below, which assume WebLogic has already been installed and occur before creating an Endeca Studio domain. It is possible to extend an existing domain to include Enterprise Manager.

1. Verify that Oracle Application Development Framework was installed as part of the Endeca Server installation.

   ADF may already be available if Endeca Studio is installed on the same WebLogic Server.

2. When creating the Endeca Studio Domain for the first time, select **Oracle Enterprise Manager**.

   JRF is automatically included. If a Studio Domain has already been created, the existing domain can be extended to include Enterprise Manager.

3. Start WebLogic and create a Credential Store to save the OBIEE credential information using Enterprise Manager.

4. Log in to Enterprise Manager and, in the WebLogic Domain, right-click **endeca_studio_domain**, **Security**, then **Credentials**.

   Create a new map (for example, oracle.bi.enterprise) and key (for example, repository.OBIA), then save the OBIEE user name and password information.

   Note:

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

Including the Oracle BI Applications Security Manager Related Files

The Oracle BI Enterprise Edition account credentials with administrator and impersonator roles are saved locally in the Endeca Studio Domain's Credential Store. The BI Applications Security Manager obtains the password information from the Credential Store to connect to Oracle BI Enterprise Edition using JDBC.

To set up the credential store, follow the steps below, which assume WebLogic has already been installed and occur before creating an Endeca Studio domain. It is possible to extend an existing domain to include Enterprise Manager.

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

1. Save a copy of the Endeca Studio installation .ear file with a different file name, for example, `OBIA-endeca-portal-weblogic-3.1.13849.ear`.

2. Add `OBIAMDEXSecurityManager.jar` and `bijdbc.jar` to Endeca Studio's .ear file under `\APP-INF\lib\` using a compression utility.

3. Deploy the .ear into WebLogic following the installation instructions.

   For information about deploying the .ear, refer to Deploying Studio to the WebLogic domain. If there is an existing Endeca Studio deployment, be sure to undeploy it first. Note the name used for the deployment, as this value will be used when modifying the `system-jazn-data.xml`. A typical name for the deployment is OBIA-endeca-portal-weblogic-3.

4. Add directories on the WebLogic server named XML and User_Input under `$MW_HOME\user_projects\domains\endeca_studio_domain.`

5. Add or create the config.properties file in the User_Input directory.

   Set the parameters as follows. A sample file is available in *`MW_HOME`* `\Oracle_BI1\biapps\admin\provisioning\endeca \OracleBIApps_Endeca.zip (OBIAMDEXSecurityManager/User_Input/.`

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

6. Add or create the `securitycolumns.csv` file in the `User_Input` directory.

   A sample file is available in *`MW_Home`*`\Oracle_BI1\biapps\admin\provisioning\endeca\OracleBIApps_Endeca.zip(OBIAMDEXSecurityManager/User_Input/`. This .csv file has three columns, Endeca Server Connection ID, Collection Name, and Security Columns, used to relate the security columns.

Column Name	Description
Endeca Server Connection ID column	Indicate the Endeca Server Connection ID of the Endeca Studio Application that uses OBIEE security.
Collection Name	Indicate the Data Set name of the Endeca Studio Application that uses OBIEE security.
Security	Indicate the logical table name on which the row-level Security is applied. For example: `Endeca Server Connection ID,Collection Name,Security Columns OEID_Financials_AR_Balance,Fact_Fins_AR_Balance,""" Core""."Dim - Date Fiscal Calendar""."Fiscal Year"""`

7. Start and log in to Endeca Studio.

   From the Control Panel, select **Framework Settings** and set `df.mdexSecurityManager` from **com.endeca.portal.data.security.DefaultMDEXSecurityManager** to c**om.endeca.portal.extensions.OBIAMDEXSecurityManager**.

8. Click **Update Settings**.

9. Shut down the WebLogic Server.

10. Add the following entry to the system-jazn-data.xml file found in `$DOMAIN_HOME\config\fmwconfig`, under the <system-policy> and <jazn-policy> tags.

    This includes permissions for Studio to access the Credential Store.

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

```
</permissions>
</grant>
```

Examples of values include:

- appName=OBIA-endeca-portal-weblogic-3
- mapName=oracle.bi.enterprise
- keyName=repository.OBIA

11. Restart the WebLogic Endeca Studio Server with the new custom Security Manager applied and in use by Endeca Studio.

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

Update the setDomainEnv script file, which is named `setDomainEnv.cmd` in Windows environments and `setDomainEnv.sh` in Linux.

The file is located in the bin subdirectory of the domain directory, *`MiddlewareHomeDirectory`*`/user_projects/domains/endeca_studio_domain/bin/`.

1. Search for the following in the setDomainEnv.cmd file:

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

2. Before the above IF statement, add the following, which sets a higher `-Xmx` or maximum memory heap size:

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

Optional: Enabling Verbose Debugger Logging

You can enable logging of debugging messages in the log file located in `$MW_HOME\user_projects\domains\endeca_studio_domain\eid-studio.log`.

1. Select **Server Administration** in the Control Panel, then select the **Log Levels** tab.
2. Select the **Add** Category tab and enter:
   - `com.endeca.portal.extensions.OBIAMDEXSecurityManager (DEBUG)`
   - `com.endeca.portal.extensions.OBIAMDEXSecurityManager.BIHandlers (DEBUG)`
3. Log out, then log back in to enable the changes.

   This step has to be repeated if the Endeca Studio server is restarted.

Overriding Screen Name Validator in Endeca Studio

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

1. Shut down Endeca Studio.
2. Open the portal-ext.properties file under *%WLS_HOME*\user_projects\domains\endeca_studio_domain\eid\studio.

   Back up the existing portal-ext.properties file.
3. Add the following at the bottom of the file and save it:

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

Defining New Users in Endeca Studio and Adding Users to Studio Applications

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

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

Important: The BI EE user must belong to a BI Applications Application role, for example, AR Analyst. The data filter must be defined for that role. If it's defined for this BI EE user, it doesn't work in the Endeca Studio.

Optionally, you can also change the default behavior, which is to have Endeca users log in using their email addresses, so that logins are consistent with Oracle BI Enterprise Edition. For information on administration tasks for Endeca Information Discovery Studio, see Endeca Information Discovery Studio Administration and Customization Guide.

Adding Users and Managing Studio Applications Permissions in Endeca Studio

Users must be added to a Studio Application to enable them to view it.

Documentation Resources

Endeca Information Discovery Studio Administration and Customization Guide

- For information about how add users to Studio Applications in Endeca Studio, refer to *Adding and removing application members*.
- By default, users are able to create new applications. To prevent this ability for a user, remove the Power User role. For information about removing this role, refer to the section titled, *Preventing a user from creating applications*.

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

Troubleshooting Endeca Integration

Follow these instructions to troubleshoot common issues for Endeca integration.

Troubleshooting ETL

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

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

5

Researching Data Lineage

Data lineage dashboards provide reporting on out-of-the-box Business Intelligence Application module metrics and data, allowing data analysis from the transactional source application through the Business Intelligence repository and the underlying ETL mappings.

Using data lineage dashboards, business analysts and those customizing ETL and repository objects can gain insight into where the data in their applications' reports and dashboards is being sourced from, how it is modeled in the Oracle BI Applications repository, and how it is loaded during ETL. Data lineage dashboards are useful in performing business analysis and in understanding source-to-target ETL mappings and data transformation logic when planning or implementing Oracle Business Analytics Warehouse customizations.

Topics:

Setting Up Data Lineage

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

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

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

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

1. Configure the environment to support data lineage, as described in Setup Step: Configure ODI Topology and Load Plan.
2. Configure the WebLogic Server heap size, as described in Setup Step: Configure the WebLogic Server Heap Size.

3. Optionally, disable the Fusion step if you are using other BI Applications sources, as described in Setup Step: Disable Fusion Step (Optional).

4. Create data lineage warehouse tables, as described in Setup Step: Create the Data Lineage Warehouse Tables.

5. Extract metadata using Windows OBIEE clients, as described in Extracting Oracle Business Intelligence Metadata Using Catalog Manager and Administration Tool.

6. Execute the data lineage load plan in ODI to load the data lineage warehouse tables, as described in Executing and Monitoring the Data Lineage Load Plan. This step loads the data lineage warehouse tables.

Setup Step: Configure ODI Topology and Load Plan

The ODI Work Repository comes preconfigured with required topology and environment settings to support data lineage data extraction from the dashboards' five sources.

During initial setup, you configure prebuilt data lineage sources in the ODI topology so that ODI, Configuration Manager, and Oracle Business Intelligence Enterprise (OBIEE) data can be sourced during data lineage ETL. You then configure a prebuilt data lineage home directory variable which provides access to extracted metadata files and configure an adaptor list variable.

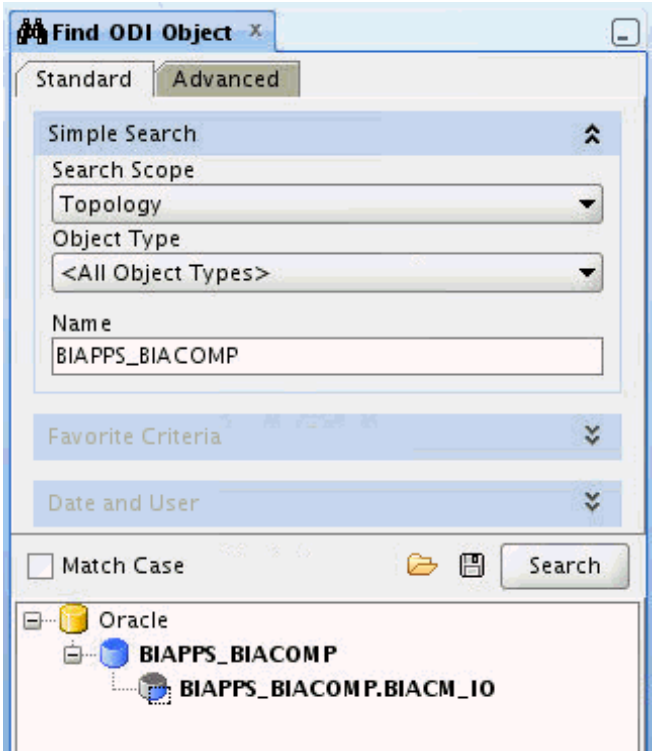
The data lineage ETL uses the following connections to extract metadata from its sources and load it into the data warehouse:

- BIAPPS_ODIREP: Used to extract ODI Metadata from ODI schema where SNP_*** tables are located.
- BIAPPS_BIACOMP: Used to extract configuration metadata from the Configuration Manager schema where C_*** tables are located.
- BIAPPS_DW: Used to load data lineage tables located in the OBAW.
- BIAPPS_DW_FILE: Used to extract Oracle Business Intelligence Enterprise (OBIEE) metadata from files. This connection should point to the BIA_11 location (`<source file home>/biapps/etl/data_files/src_files/BIA_11`)

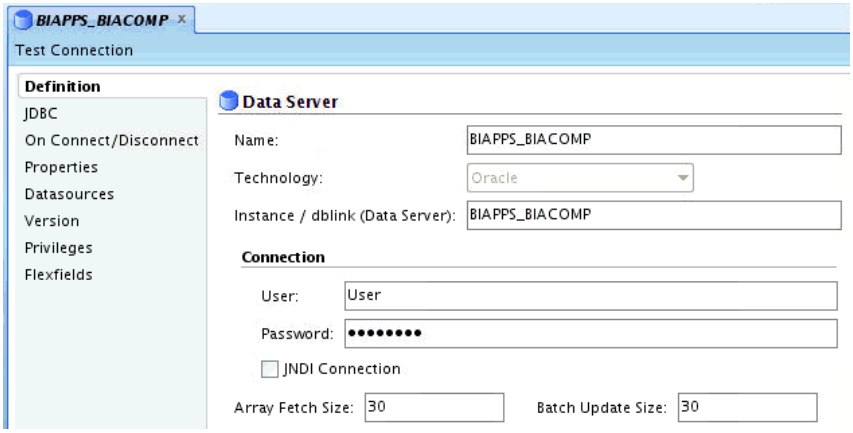
To configure the data lineage source and target connections:

1. In ODI Studio, navigate to the Physical Architecture view of the Topology Navigator's tree view, and search for the prebuilt BIAPPS_BIACOMP Data Server.

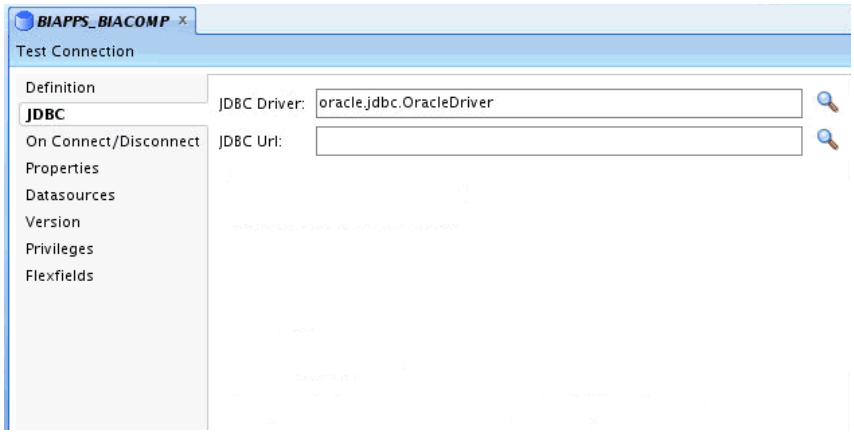
   The BIAPPS_BIACOMP connection to the Configuration Manager database may already have been configured during BI Applications installation and configuration.



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

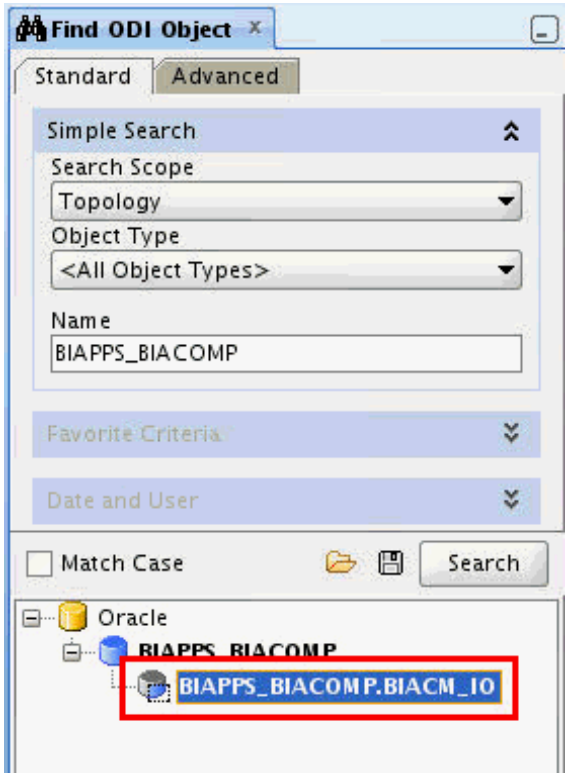


3. In the JDBC tab of the editor, verify the JDBC Driver and verify or specify a valid JDBC URL for the connection.

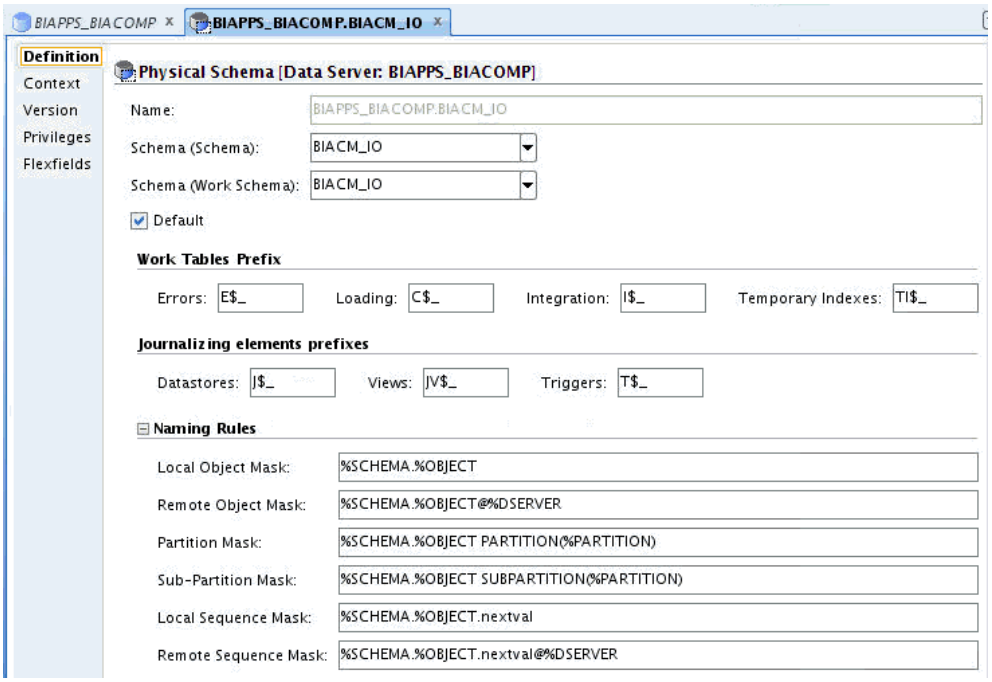


4. Click **Test Connection** to verify that the connection details you have entered are correct.

5. In the Search pane, double-click the physical schema to open it in the editor.



6. Configure or confirm the Schema and Work Schema.



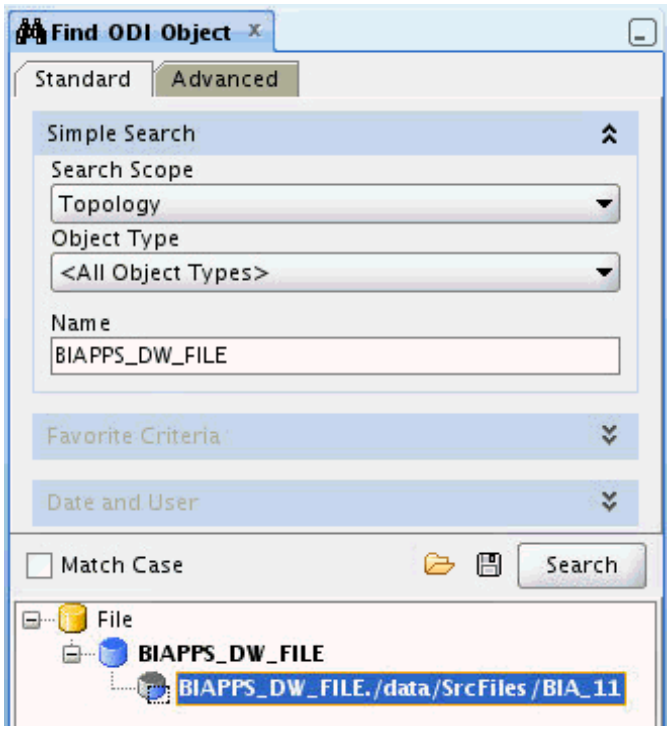
7. Repeat the above steps to configure the other connections.

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

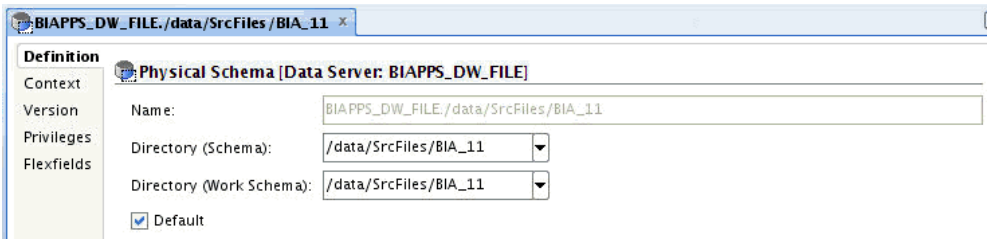
Configuring Data Lineage File Connection

Configure the data lineage file connection.

1. Search for the prebuilt **BIAPPS_DW_FILE** and double-click it to open it in the editor.

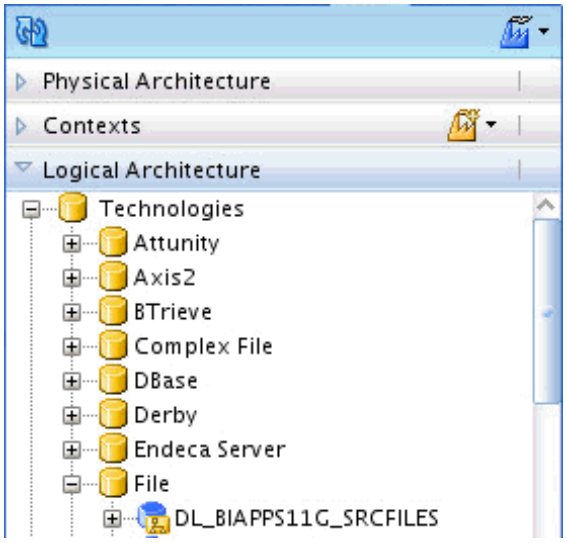


2. Configure Schema and Work Schema to point to the BIA_11 location, *source file home*/biapps/etl/data_files/src_files/BIA_11.

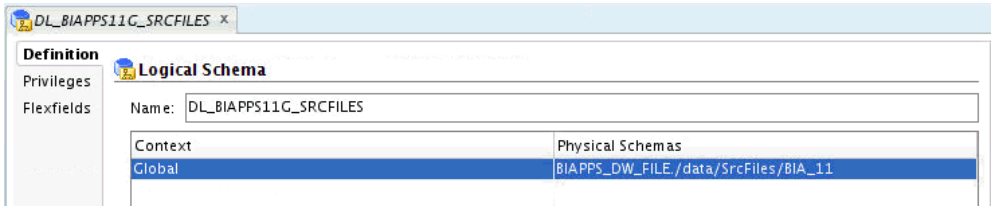


Open Logical File Connection DL_BIAPPS11G_SRCFILES. It can be located in Topology - Logical Architecture under Technologies -> File.

3. Navigate to the Logical Architecture view of the Topology Navigator's tree view, and locate the Logical File Connection DL_BIAPPS11G_SRCFILES under **Technologies**, then **File**.



4. Double-click the connection and, in the editor, associate DL_BIAPPS11G_SRCFILES to the BIAPPS_DW_FILE - Physical Schema.



Configuring the Load Plan Variable

Configure the load plan variable in the Data Lineage Extract and Load Loadplan.

- DL_HOME — Location of data lineage source files. Configure to *`source file home`*`/biapps/etl/data_files/src_files/BIA_11`, the same as the BIAPPS_DW_FILE configured in a previous step.
- ADAPTOR_LIST — List of required Source Adaptor Names.

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

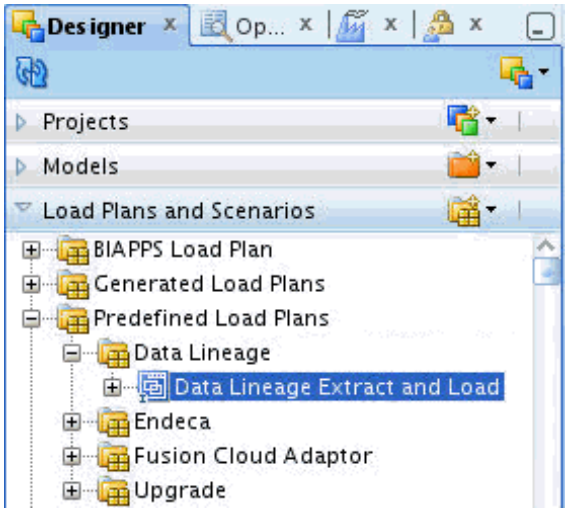
  'SDE_ORA11510_Adaptor','SDE_ORAR1212_Adaptor','SDE_PSFT_90_Adaptor','SDE_OP_V1_Adaptor','SDE_ORAR1211_Adaptor','SDE_PSFT_91_Adaptor','SDE_SBL_822_Adaptor','SDE_SBL_811_Adaptor','SDE_ORAR12_Adaptor', 'SDE_ORAR1213_Adaptor','SDE_JDEE1_91_Adaptor','SDE_JDEE1_90_Adaptor','SDE_Universal_Adaptor','SDE_FUSION_V1_Adaptor'

  The Adaptor Name has to be enclosed in single quotes. If you want to enter multiple adaptors, use commas as separators between multiple adaptor names.
- ADAPTOR_LIST — This is the same as ADAPTOR_LIST. If you want to configure multiple adaptors and ADAPTOR_LIST value has reached limits, use ADAPTOR_LIST1. Otherwise, copy the ADAPTOR_LIST value to ADAPTOR_LIST1.

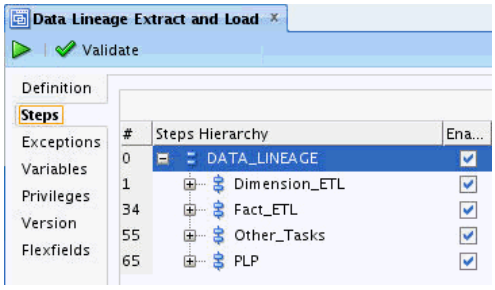
Note:

Do not leave ADAPTOR_LIST1 value empty. If the variable is empty, it will cause ETL failure.

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



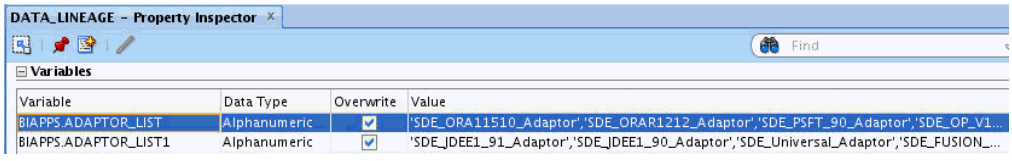
2. In the Steps tab, click the root step, **DATA_LINEAGE**.



The Property Inspector window lists all variables.

3. Enable the Overwrite option and type a value for DL_HOME, ADAPTOR_LIST and ADAPTOR_LIST1.

   Enclose Adaptor Names in single quotes and, if you want to enter multiple adaptors, use commas as separators between multiple adaptor names.



Setup Step: Configure the WebLogic Server Heap Size

Configure the WebLogic Server heap size.

Setting the WebLogic Server Heap Size in Windows

Set the heap size in Windows.

1. Open the *DOMAIN_HOME*/bin/setDomainEnv.cmd file for editing.
2. Locate the `if NOT "%USER_MEM_ARGS%"=="" (` line, and add the following code before it:

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

Setting the WebLogic Server Heap Size in Linux

Set the heap size in Linux.

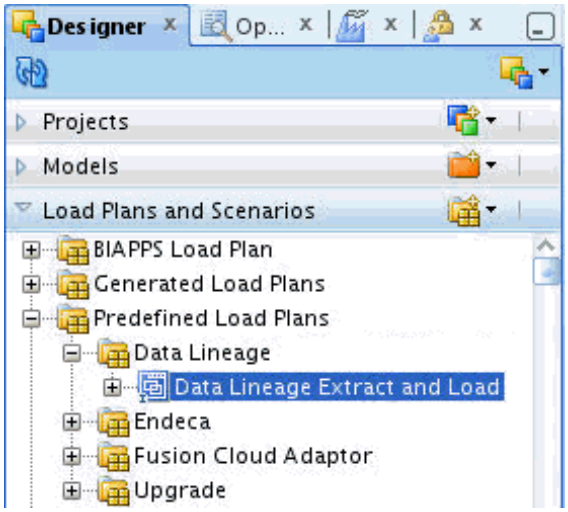
1. Open the *DOMAIN_HOME*/bin/setDomainEnv.sh file for editing.
2. Locate the `if [ "${USER_MEM_ARGS}" != "" ] ; then` line, and add the following code before it:

```
if [ "${SERVER_NAME}" = "odi_server1" ] ; then
      USER_MEM_ARGS="-Xms512m -Xmx12288m -XX:PermSize=256m -XX:MaxPermSize=512m"
      export USER_MEM_ARG
fi
```

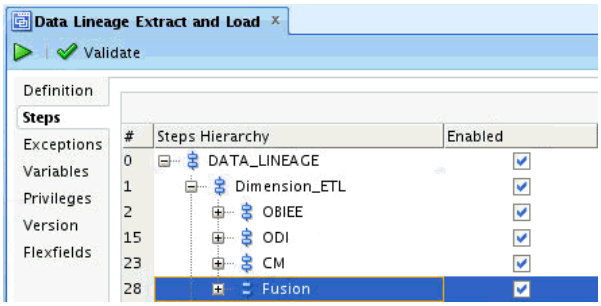
Setup Step: Disable Fusion Step (Optional)

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

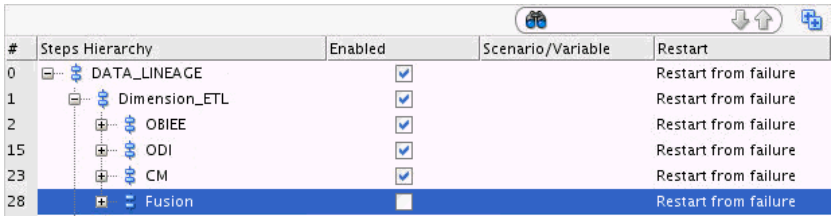
1. In ODI Designer tree view, navigate to the Load Plans and Scenarios view and open the Data Lineage Extract and Load load plan under **Predefined Load Plans**, then **Data Lineage**.



2. In the Steps tab, in the Steps Hierarchy, expand **DATA LINEAGE**, then **Dimension_ETL**.

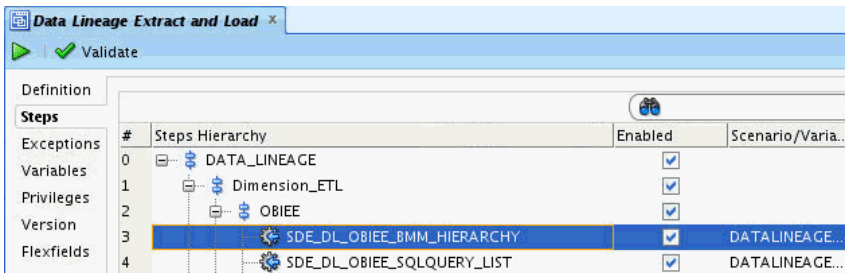


3. Deselect the **Enabled** check box for the Fusion step (step 28) to disable the step.



4. Expand **DATA LINEAGE**, then **Fact_ETL** in the hierarchy and disable the Fusion step (step 52).

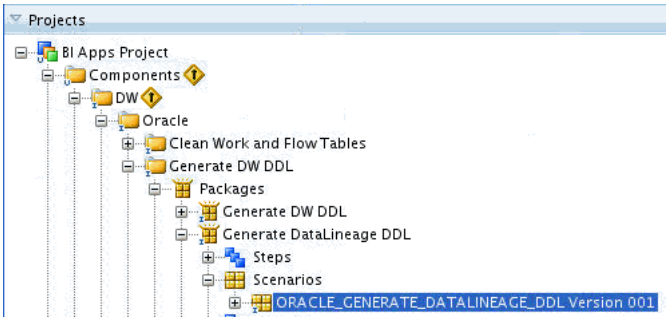
5. Expand **DATA_LINEAGE**, **Dimension_ETL**, then **OBIEE**, and enable the SDE_DL_OBIEE_BMM_HIERARCHY step (step 3).



Setup Step: Create the Data Lineage Warehouse Tables

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

1. In the ODI Studio Designer Navigator, expand **BI Apps Project**, **Components**, **DW**, **Oracle**, **Generate DW DDL**, **Generate DataLineageDDL**, **Scenarios**, then **ORACLE_GENERATE_DATALINEAGE_DDL**.

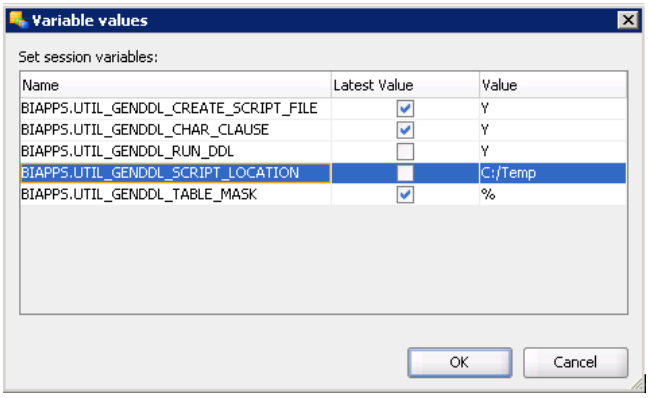


2. Select the **ORACLE_GENERATE_DATALINEAGE_DDL** and click the **Execute** (green arrow) button.

3. In the Execution dialog box, choose **Agent** and click **OK**.

4. In the Variable Values dialog box, change values for the third and fourth variables. Set BIAPPS.UTIL_GENDDL_RUN_DDL to 'Y' and set BIAPPS_UTIL_GENDDL_SCRIPT_LOCATION to a valid local folder location.

   Set BIAPPS.UTIL_GENDDL_RUN_DDL to **Y** and set BIAPPS_UTIL_GENDDL_SCRIPT_LOCATION to a valid local folder location.



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

Tasks for Loading and Refreshing Data Lineage Dashboards

Perform these tasks for initial and ongoing loading and refreshing of data lineage dashboards and their required metadata.

> **Note:** You must perform the tasks in this section in the sequence described in Setting Up Data Lineage.

Topics:

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

Extracting Oracle Business Intelligence Metadata Using Catalog Manager and Administration Tool

Extract Presentation Catalog (webcat) and repository (RPD) metadata into files sourced during load of the Data Lineage warehouse tables.

Prerequisities:

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

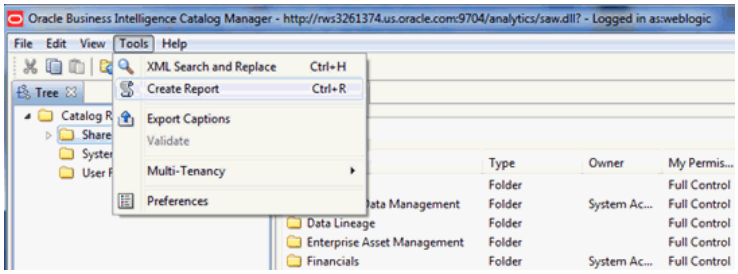
You use Windows-based OBIEE clients from release 11.1.1.6.0 or 11.1.1.7.0 to generate the following files:

- `webCat_dashboard_text.txt`
- `webCat_ text.txt`
- `rpd_text.txt`

Extracting the Dashboard Metadata

Extract the dashboard metadata.

1. Open Oracle Business Intelligence Catalog Manager 11.1.1.6.0 and connect using a connection type of online to the Oracle Business Intelligence Server, URL: `http://`*`Host`*`:`*`port`*`/analytics/saw.dll`.
2. Select **Tools**, then **Create Report**.

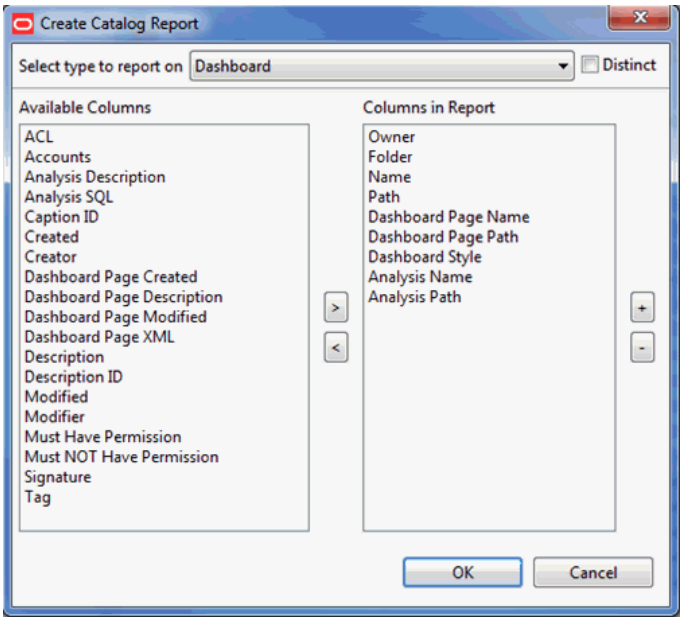


3. In the Create Catalog Report dialog box, select **Dashboard** in the Select type to report on drop-down list
4. In the Available Columns list, select the following columns and arrange them in order.
   - Owner
   - Folder
   - Name
   - Path
   - Dashboard Page Name
   - Dashboard Page Path
   - Dashboard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

- Analysis Name
- Analysis Path

Note:

The column order must be as above.



5. Click **OK**.

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

6. Save the file as `webCat_dashboard_text.txt` in a local folder.

Extracting the Report Metadata

Extract the report metadata.

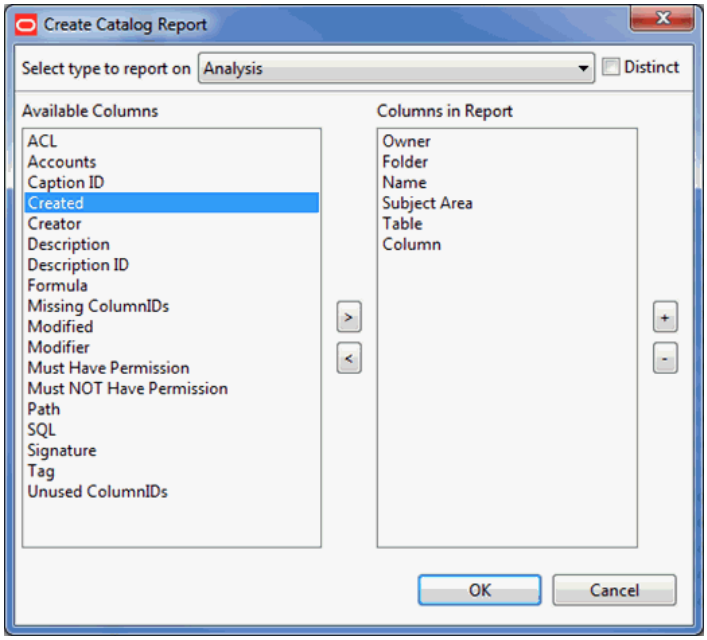
1. Select **Tools**, then **Create Report**.
2. In the Create Catalog Report dialog box, select **Analysis** in the Select type to report on drop-down list.
3. In the Available Columns list, select the following columns and arrange them in order.

   - Owner
   - Folder
   - Name
   - Subject Area
   - Table

- Column

Note:

The column order must be as above.

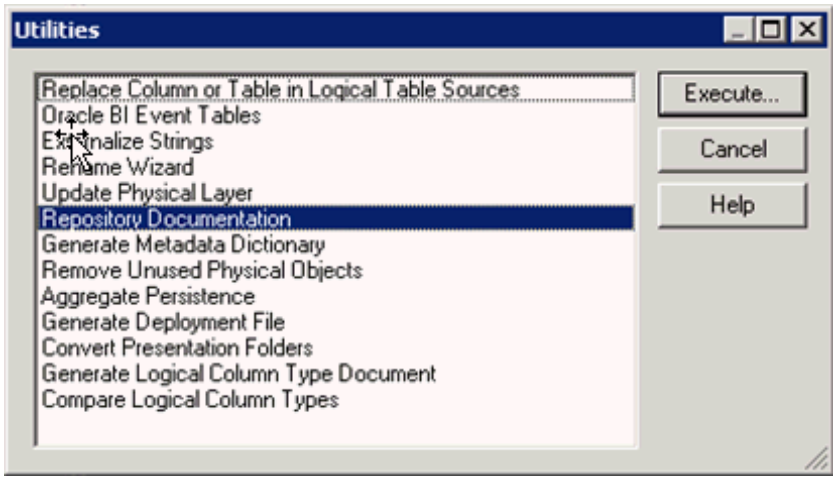


4. Click **OK**.
5. Save the file as `webCat_text.txt` in a local folder.

Extracting the RPD Metadata

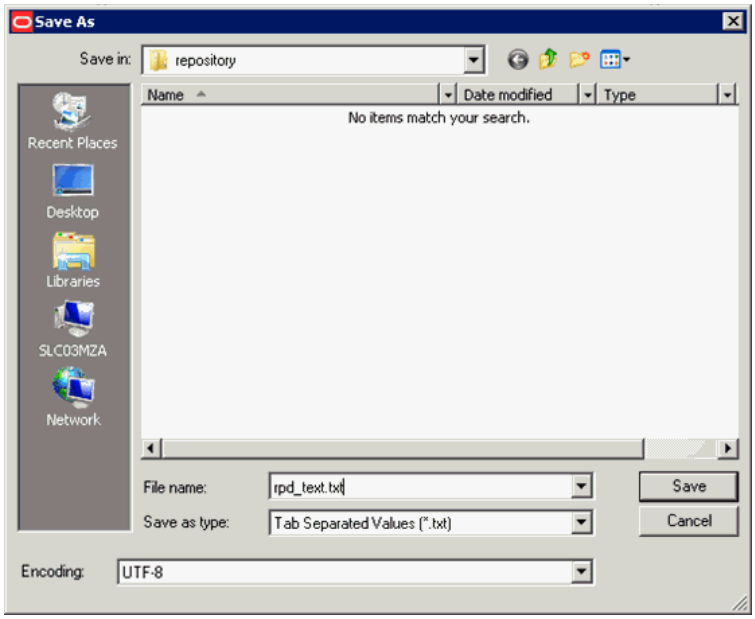
Extract the RPD metadata.

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



4. In the Save As dialog box, save the file as `rpd_text.txt`.

   Select **Tab Separated Values** (*.txt) in the Save as Type drop-down list



Copying to the $DL_HOME Directory

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

1. Copy the `webCat_dashboard_text.txt`, `webCat_ text.txt`, and `rpd_text.txt` from the local directory on the Windows machine to `$DL_HOME`.

2. Copy Fusion EAR files from `$ORACLE_BI_HOME/biapps/DataLineage` to `$DL_HOME`.

Executing and Monitoring the Data Lineage Load Plan

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

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

About Performing Analysis with Data Lineage Dashboards

Data lineage is analyzed across a variety of Oracle BI Applications entities, using dashboard prompts for the details on the DataLineage Summary page of the OBIA DataLineage dashboard.

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

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

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

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

reports describing the warehouse tables, and listing the ODI interfaces and adaptors that load both from the OLTP source into the staging tables and from the staging tables into the warehouse dimension and fact tables.

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

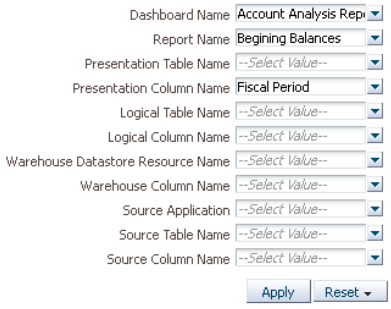
Analyzing Data Lineage for Oracle Business Intelligence Applications

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

To analyze data lineage for Oracle Business Applications:

1. Navigate to the OBIA Data Lineage dashboard.
2. In the Data Lineage Summary page, make a selection from one or more of the available prompts.

   You can make multiple selections in prompts, and available prompt selections are constrained by earlier selections in other prompts.



3. Click **Apply** to display the OBIA - LineageSummary report, which provides lineage details from the presentation dashboards and reports through BI Applications repository metadata, and finally to the warehouse table and column.
4. To drill to the detailed reports in the Dashboard Implementation page of the dashboard, click the **Dashboard Name** value in the summary report.

   This opens the Dashboard Implementation page with the Dashboard Name dashboard prompt pre-selected with the dashboard name. You could also click other entities in the report to navigate to other pages. For example, clicking a Warehouse Column Name and selecting Implementation would navigate to the Warehouse Implementation page, which focuses on ETL implementation.Examine the reports in the Dashboard Implementation page.

   Examine the reports in the Dashboard Implementation page:

- The first report in the page, DashboardImplementation- OBIA - LineageSummary, provides a similar lineage to the default LineageSummary report, tracking lineage from the dashboard through the Presentation catalog to the warehouse table and column.
- The second report, DashboardImplementation-DashboardDetails, focuses on the dashboard alone, detailing dashboard pages, reports, the Presentation Catalog, and the associated Presentation table and column names.
- The third report, DashboardImplementation-OBIA-RPDImplementation, focuses on the lineage of the data in the BI repository metadata, covering all three layers of the repository, starting from Presentation Catalog names through business model and logical table and column names, and down to the physical catalog, table, and column names. The logical layer details include the expression, which often is just the logical expression of the table and column resources.
- The fourth report, DashboardImplementation-ODIImplementation-SourceToStaging, focuses on the ETL interfaces used to load the warehouse staging tables, providing the warehouse table and associated adaptor and interface details.
- The fifth report, DashboardImplementation-ODIImplementation-StagingToWarehouse, focuses on the ETL interfaces used to load the warehouse target fact and dimension tables, providing the warehouse table and associated adaptor and interface details.

5. Navigate back to the DataLineageSummary page, click a **Report name** value, and select **Report Implementation** to open that report as the dashboard prompt value in the Report Implementation page.

   Scroll through the reports on this page and notice that they closely mirror those provided for dashboards.

6. Navigate to the Presentation Layer Implementation page, which includes reports detailing the logical and physical column derivation in the RPD for Presentation columns.

   There are also reports describing the warehouse tables, and listing the ODI interfaces and adaptors that load both from the OLTP source into the staging tables and from the staging tables into the warehouse dimension and fact tables.

7. Navigate to the Warehouse Implementation page, in which the WarehouseImplementation-OBIA-LineageSummary report summarizes the relationship between warehouse tables and columns and associated models and source transactional tables and columns.

6

Administering Oracle GoldenGate and Source Dependent Schemas

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

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

In addition to the ETL use case, you can reconcile ledger information available in your Oracle E-Business Suite and Oracle Fusion Applications using Oracle GoldenGate.

Source Dependent Schema Architecture

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

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

- Source Dependent Data Store (SDS): A separate schema on the OBAW database that is a replication of the source OLTP systems tables. Also stores deletes and additional optimizations for incremental ETL.

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

Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema

Perform these detailed tasks to set up Oracle GoldenGate and SDS.

Note: You must perform the tasks in this section in the listed sequence.

Topics:

Setup Step: Configure Source and Target Database

In this step, you create Oracle GoldenGate database users on source and target databases. Unlike other database schemas used by BI Applications, the SDS and OGG schemas are not automatically created during installation.

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

1. Create the OLTP Oracle GoldenGate database user.

Each OGG process requires a dedicated database user. On the source system, the OGG user needs to be able to query various metadata.

Secure database practice is to avoid granting privileges to tables not in use, so `SELECT ANY TABLE` is not granted to the OGG database user. Instead, as part of the SDS DDL, `SELECT` privileges are granted only to those tables in the OLTP schema being replicated.

The user creation scripts use the following parameters:

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

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

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

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


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

2. Prepare the OLTP database and redo logs.

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

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

3. Create the target Oracle GoldenGate database user.

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

   The user creation scripts use the following parameters:

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

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

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

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

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

4. Create the SDS database user.

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

Source Instance Name	Model Code	Data Source Number	SDS
Oracle EBS 11.5.10	EBS_11_5_10	310	BIAPPS_SDS_ EBS_11_5_10_310
Siebel CRM 8.1.1	SEBL_8_1_1	625	BIAPPS_SDS_ SEBL_8_1_1_625

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

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

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

```
DEFAULT COMPRESS FOR OLTP;

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

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

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

Setup Step: Install Oracle GoldenGate on Source and Target Systems

Download and install Oracle GoldenGate software first on the source and then on the target machines. The software is available from Oracle Technology Network.

For information about installation of Oracle Golden Gate, refer to the Oracle GoldenGate Installation and Setup Guide for your platform and database.

Installation Recommendations

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

- For each OLTP instance supported, install a separate Replicate process on the target machine. As each OLTP instance has its own separate SDS schema on the target database, the Replicate process is populating different targets so a separate Replicate process is required for each.
- Install a Data Pump process on the source machine.
- The name of the Extract, Data Pump and Replicat processes are limited to eight characters. The suggested naming convention is as follows:

Process	Naming Convention	Example
Extract	EXT_Datasource Num Id	EXT_310
Data Pump	DP_*Datasource Num Id*	DP_310
Replicate	REP_*Datasource Num Id*	REP_310

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

Example Steps to configure the Oracle GoldenGate processes

These example steps illustrate how to configure the OGG processes. Modify these steps as appropriate for your environment.

For the source system, configure Extract and Data Pump processes. The initial steps in the example below effectively remove an existing instance of both processes. If none already exist, start with the START MGR command.

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

STOP MGR

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

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

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

--Start Manager on primary database
START MGR

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

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

Example:

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

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

STOP DP_310
DELETE RMTTRAIL ./dirdat/*
DELETE DP_310

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

```
--Start Manager on primary database
START MGR

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

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

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

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

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

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

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

--Start Manager on target database
START MGR

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

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

Example:

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

--Stop GG processes
STOP REP_310
DELETE REP_310

--Delete CHECKPOINTTABLE
DELETE CHECKPOINTTABLE

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

--Start Manager on target database
START MGR
```

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

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

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

Configure BI Applications Configuration Manager and Oracle Data Integrator to support the Source Dependent Schema.

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

1. In Configuration Manager, select the **Source Instance**.
2. Click **Manage Data Load Parameters**.
3. Locate the IS_SDS_DEPLOYED parameter in the list.
4. In the Global Parameter Value, replace <Edit Value> with **Yes**.

   A warning is displayed indicating that the parameter is being updated globally for all Fact and Dimension Groups.
5. Click **Yes** to confirm or, if you prefer, set the global parameter to **No**, and then edit the parameter value at the Fact Group or Dimension Group level.

Adding SDS Physical Schemas in ODI

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

1. In ODI Studio's Topology Navigator, expand the Oracle technology in the Physical Architecture.
2. Right-click **Oracle_BI_Apps_DW** and select **New Physical Schema**.
3. In the Definition, set Schema (Schema) and Schema (Work Schema) both to the SDS schema owner.
4. Select **Flexfields**.
5. For the DATASOURCE_NUM_ID flex field, uncheck the **Default** checkbox and assign the DSN value associated with that source as defined in Configuration Manager.
6. Save the physical schema definition.

   Ignore the message about the physical server not being assigned a context.

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

Generate and run the Data Definition Language to create the SDS tables on the SDS schema in the target database.

1. Generate the SDS DDL.

   Procedures are provided to generate the required objects to enable the SDS. To generate the required DDL, in ODI Designer execute the 'Generate DDL - SDS' scenario found under **BI Apps Project**, **Components**, **SDS**, **Oracle**, then **Generate SDS DDL**. Provide an appropriate context when prompted.As the procedure can only accept a single source type, this process needs to be repeated for each different type of Source system to be enabled.

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

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

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

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

2. Grant privileges to OLTP tables.

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

   The SDS DDL generator procedure creates a script to grant `SELECT` privileges to the OGG user. Refer to the script `BIA_SDS_Schema_Source_Grants_DDL_`*`unique ID`*`.sql` and execute the contents in the OLTP database with a user with sufficient privileges to grant `SELECT` privileges on the OLTP tables.

3. Create the SDS tables.

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

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

4. Perform initial Load of the SDS tables: create database link to OLTP database.

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

   Note:

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

   Note:

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

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

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

Note:

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

5. Perform initial load of the SDS tables: execute procedure to generate DML script.

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

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

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

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

6. Perform initial load of the SDS tables: execute DML script on SDS database.

   The resulting DML script can be executed using the SDS schema owner or the BIAPPS DW schema owner. If executed by the BIAPPS DW schema owner, this

user must be granted the `SELECT ANY TABLE` and `INSERT ANY TABLE` privileges in order to populate data in another schema. If executed using the SDS schema owner, a private database link named "DW_TO_OLTP" must be created in the SDS schema (the SDS user must be granted the `CREATE DATABASE LINK` privilege to create this database link) or already created as a public database link.

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

7. Create SDS indexes and analyze the SDS schema.

   When the tables are populated, execute the `BIA_SDS_Schema_Index_DDL_`*`unique ID`*`.sql` script to create indexes and analyze the SDS tables.

Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Systems

Parameter files are used to control how Oracle GoldenGate operates. These files are deployed to the source system, where the Extract and Data Pump processes are executed, and the target system, where the Replicat process is executed.

An ODI procedure generates these parameter files based on the metadata defined in ODI. A scenario that executes this procedure is provided to generate the Oracle GoldenGate parameter files to populate the SDS.

Generate Oracle GoldenGate Parameter Files

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

Parameter	Description
PARAM_FILE_LOCATION	Location on machine where ODI client is running where parameter files will be created. Example: `C:\temp\`
DATASOURCE_NUM_ID	Datasource Num Id value associated with the particular source for which parameter files are to be generated. Example: 310
DATAPUMP_NAME	Name of the Datapump Process specified when installing OGG on the source machine. Limit is eight characters. Suggested naming convention is DP_*Datasource Num Id*, for example DP_310.

Parameter	Description
EXTRACT_NAME	Name of the Primary Extract Process specified when installing OGG on the source machine. Limit is eight characters. Suggested naming convention is EX_*Datasource Num Id*, for example EXT_310.
EXTRACT_TRAIL	Path and name of trail file on source system. Can be a relative or fully qualified path, though actual file name must be two characters. In the example below, 'tr' is the name of the trail file. Example: `./dirdat/tr`
DEFSFILE	The relative or fully qualified path on the source system where the DEFGEN definition file should be created and file name. This value is included in the DEFGEN.prm parameter file that is generated by this procedure. The DEFGEN utility is executed on the source database, so the path provided must be a path available on the system the source database runs on. Suggested naming convention is DEF_*Datasource Num Id*.def. Example: `./dirdef/DEF_310.def`
SOURCE_GG_USER_ID	Database user dedicated to the Oracle GoldenGate processes on the source database. Example: GG_USER
SOURCE_GG_PASSWORD	Password for the database user dedicated to the Oracle GoldenGate processes on the source database. By default, the password is stored as clear text in the generated parameter file. If an encrypted value is desired, use the ENCRYPT PASSWORD utility and edit the generated parameter files accordingly. Example: GG_PASSWORD
SOURCE_PORT	Port used by the OGG Manager Process on the source system. The default value when OGG is installed is 7809.
REPLICAT_NAME	Name of the Replicat Process specified when installing OGG on the target machine. Limit is eight characters. Suggested naming convention is REP_*Datasource Num Id*, for example REP_310

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

The procedure automatically creates subfolders under a folder you specify. The naming convention is DSN_*DATASOURCE_NUM_ID* where *DATASOURCE_NUM_ID* is the value you specify when executing this procedure. For example, if you specify 310 as the value for DATASOURCE_NUM_ID, there will be a folder named DSN_310. Under this folder are two more subfolders, 'source' and 'target'. The 'source' folder contains all of the files that need to be copied to the source system, while 'target' contains all of the files that need to be copied to the target system.

Tip:

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

About JD Edwards Support

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

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

Keep the following in mind when generating the parameter files.Execute the procedure for each database instance. The name of each process and trail file should be unique. The following example assumes all four components are on different databases:

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

About PeopleSoft Learning Management Support

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

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

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

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

Configure the Source System

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

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

- ADD_TRANDATA.txt

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

- DEFGEN.prm
- *EXTRACT_NAME*.prm where <EXTRACT_NAME> is the value specified when generating the parameter files.
- *DATAPUMP_NAME*.prm where <DATAPUMP_NAME> is the value specified when generating the parameter files.

Edit the Extract parameter file

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

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

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

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

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

Save and close the file.

Enable Table Level Logging

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

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

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

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

Note:

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

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

Generate Data Definition File on the Source System

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

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

```
GGSCI>EDIT PARAMS DEFGEN
```

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

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

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

Configure the Target System

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

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

- *REPLICAT_NAME*.prm where <REPLICAT_NAME> is the value specified when generating the parameter files.

Edit the Replicat Parameter File

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

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

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

Save and close the file, and exit GGSCI.

Create a Checkpoint Table (Optional)

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

Start the GGSCI command utility:

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

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

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

Setup Step: Start Oracle GoldenGate on Source and Target Systems

Start Oracle GoldenGate on source and target systems.

1. Start Oracle GoldenGate on the source system.

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

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

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

   Example:

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

   --Start pump on primary database
   START DP_310
   ```

2. Start Oracle GoldenGate on the target system.

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

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

   Example:

   ```
   START MGR

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

ETL Customization

Learn about SDS considerations for ETL customization.

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

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

Adding a Custom Source Column to an Existing ETL Task

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

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

Adding a Non-Custom Source Table to an Existing ETL Tas

In cases where an ETL task is customized to use an additional table that is included as part of the standard OLTP application, if the table is already used by another ETL task then that table should already exist in the ODI model and is already replicated in the SDS. No special steps are required.

If the table is not already used by any other ETL task, run the RKM to add the table to the ODI model and use the SDS DDL generator in incremental mode to add this table to the SDS schema. Use one of the initial load options with this table in the Table List to repopulate the table. Regenerate the SDS parameter files to ensure the table is included as part of the replication process.

Creating a Custom ETL Task

If a custom ETL task sources a table that is already extracted from, no special steps are required. However, if the custom task extracts from a new table that is not already included as part of the standard Oracle BI Applications source-specific model, run the RKM to add the table to the ODI model and use the SDS DDL generator in incremental mode to add this table to the SDS schema. Use one of the initial load options with this table in the Table List to repopulate the table. Regenerate the SDS parameter files to ensure the table is included as part of the replication process.

Patching

After releasing BI Applications, Oracle may release patches. This section discusses patches that impact SDS related content and considerations when deploying those patches.

Patch Applied to ODI Datastores or Interfaces

ODI datastores and interfaces are the only BI Applications content that impacts SDS related content. Applied patches that impact OLTP-specific datastores are relevant to the SDS.

It is possible that an applied patch could change the definition of an OLTP-specific datastore, for example adding a column or changing a column's size or datatype. A patch could also introduce a new table. In these cases, run the SDS DDL generator in incremental mode, providing the list of datastores that are patched. Execute the generated DDL against the SDS schema. In case of a new column or table being introduced, run the initial load, specifying just the new or changed table in the table list in the provided procedure.

A patch could impact an interface by adding a new OLTP table from which data must be extracted. In the previous step, you would have generated the DDL and DML to create and populate this table. Run the Oracle GoldenGate parameter generator procedure to recreate the required parameter files and redeploy to the source and target systems. For more information about creating and recreating parameter files, see Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines.

Patch Applied to SDS-Related Procedure

In the case an SDS-related procedure is replaced by a patch, depending on the nature of the reason for the patch, it may be necessary to re-execute the procedure and re-deploy its output. If the patch is related to the SDS DDL or SDS Copy procedures, the procedure can be run in incremental mode to make some changes to the SDS or in full mode to completely replace the SDS. The patch notes will describe exactly what must be done to implement the patched procedure.

Troubleshooting GoldenGate and SDS

Review these troubleshooting tips and resolutions for common errors encountered during setup of Oracle GoldenGate and SDS.

Topics:

Create the SDS Tables

If you encounter any issues with the script generated by the GENERATE_SDS_DDL procedure, verify the following have been correctly set.

- The model you are specifying is associated with a logical schema.
- The logical schema's DATASOURCE_NUM_ID flexfield is assigned a numeric value. A value is automatically assigned when a datasource is registered in Configuration Manager.
- The logical schema is mapped to a physical schema in the context (for example, Global) you are executing the procedure with. The physical schema is automatically mapped to the Global context when the datasource is registered in Configuration Manager.
- The physical schema's DATASOURCE_NUM_ID flexfield is assigned the same numeric value as the logical schema. A value is automatically assigned when a datasource is registered in Configuration Manager.
- Under the same context, a physical schema is mapped to the DW_BIAPPS11G logical schema, for example BIAPPS_DW.OLAP.
- A new SDS physical schema has been added to the same physical server, for example BIAPPS_DW.SDS_EBS_12_1_3_310. This physical schema is manually added.
- The physical schema's DATASOURCE_NUM_ID flexfield is assigned the same numeric value as used previously. This value is manually assigned.

The following are some common error messages and issues you may encounter.

- `com.sunopsis.tools.core.exception.SnpsSimpleMessageException: com.sunopsis.tools.core.exception.SnpsSimpleMessageException: Exception getSchemaName("[logical schema name]", "D") : SnpPschemaCont.getObjectByIdent : SnpPschemaCont does not exist`

  Verify the logical schema is mapped in the context you are executing the procedure with.

- `java.lang.Exception: The application script threw an exception: java.lang.Exception: Model with code '[logical schema]' does not exist`

Verify the logical schema associated with your model has been assigned a value for the DATASOURCE_NUM_ID flexfield.

- `java.lang.Exception: The application script threw an exception: java.lang.Exception: Can't find physical schema for connection for DW_BIAPPS11G with DSN 310 in context Global`

  Verify a physical schema is created under the Data Warehouse physical server and assigned the same DATASOURCE_NUM_ID value as assigned to the OLTP.

Using the DML Option to Perform an Initial Load

If you encounter any issues with the script generated by the `COPY_OLTP_TO_SDS` procedure, verify the following have been correctly set.

A database link with the name DW_TO_OLTP is created in the database user schema used by the data warehouse Data Server (BIAPPS_DW) that points to the OLTP database. The procedure is executed by this user so Oracle looks for this database link in the user's schema, not the SDS schema. You still need a database link with this name in the SDS schema for other reasons, so you have a total of two database links to the same source database.

The following are some common error messages and issues you may encounter.

- `ODI-1228: Task Copy SDS Data (Procedure) fails on the target ORACLE connection BI_APPLICATIONS_DEFAULT`

  PL/SQL: ORA-00942: table or view does not existPLS-00364: loop index variable 'COL_REC' use is invalid"

  Verify a database link named DW_TO_OLTP exists in the schema owned by the database user associated with the DW physical server.

- `Insert statement only populates the CDC$ columns`

  The script has statements such as the following where only the CDC$ columns are populated:

```
truncate table SDS_EBS11510_FULL.HR_LOCATIONS_ALL;
INSERT /*+ APPEND */ INTO SDS_EBS11510_FULL.HR_LOCATIONS_ALL (CDC
$_SRC_LAST_UPDATE_DATE, CDC$_RPL_LAST_UPDATE_DATE, CDC$_DML_CODE) SELECT SYSDATE,
SYSDATE, 'I'
FROM HR_LOCATIONS_ALL@DW_TO_OLTP;
```

  Verify the database link DW_TO_OLTP is pointing to the correct OLTP database. The procedure gets a column list from the data dictionary on the OLTP database for the tables that correspond to the SDS tables. If the database link points to the wrong database, a column list will not be retrieved.

Create SDS Indexes and Analyze the SDS Schema

When executing the script to create indexes and primary key constraints on the SDS tables, you may see some of the following error or warning messages.

- **"such column list is already indexed"**

  You may see this message when executing the script that creates the indexes. This message can be ignored.

Oracle GoldenGate works best when a primary key is defined on the target table in order to determine which record to update. If a primary key is not found, the replicat process searches for a unique index to determine which record to update. The definition of the tables in the SDS is based on the definition in the source system (leveraging both the application dictionary and data dictionary). If the table does not have a primary key in the source system but does have multiple unique indexes, a primary key may be added in the ODI definition to ensure Oracle GoldenGate can correctly identify the record to be updated. This primary key may be defined on the same column that a unique index is already defined on. The DDL script creates the primary key as an index and a constraint before creating the unique index. When creating the unique index, the database will report that the column is already indexed.

- **"column contains NULL values; cannot alter to NOT NULL"**

  This error can occur when a primary key constraint is being created. Primary key constraints are introduced in ODI when a primary key is defined in the OLTP system. A primary key constraint may also be introduced in ODI when there is no primary key in the OLTP system for the table but the table has multiple unique indexes; in this case, a primary key constraint is introduced to ensure Oracle GoldenGate does not use a unique index that may not correctly identify a record for updates. This error can occur for two reasons:

  – **OLTP table has Primary Key Constraint**

    Due to differences in patches and subversions, it is possible the OLTP instance used to originally import the datastores from had a primary key constraint that differs from the OLTP release you are using. If the OLTP table has a primary key constraint, ensure the definition in ODI matches this primary key. If there is a difference, you can modify the Index DDL script to use the proper definition to create the primary key constraint in the target database. You should also update the constraint in ODI to match this definition.

    If the OLTP and ODI definitions of the primary key constraint match, it is possible the initial load process did not populate one or more of the columns that make up the primary key. If the primary key includes a LOB or LONG datatype, data is not replicated in these columns, which would leave the column empty. In this case, no unique index or primary key can be created, and without data in this column the record cannot be uniquely identified. Any ETL task that extracts from this table needs to be modified to extract directly from the OLTP system. This is done by modifying the load plan step for this task, overwriting the IS_SDS_DEPLOYED parameter for that load plan step to a setting of 'N'.

    If the OLTP and ODI definitions of the primary key constraint match and the key does not include a column that has either the LOB or LONG datatype, review the initial load files and verify whether the column is populated or not. Refer to Using the DML Option to Perform an Initial Load for more details.

  – **OLTP table does not have Primary Key Constraint**

    Primary key constraints in the ODI model are introduced when a primary key may not exist in the original table. This primary key generally matches an existing unique index. Due to differences in patch and subversions for a given OLTP release, it is possible that the instance used when importing a unique index had a column that is not nullable but in another OLTP release, that column may be nullable. Unique indexes allow null values but primary keys do not. In this case, a unique index is created for the SDS table but the primary key constraint fails to be created. Oracle GoldenGate uses the first unique index it

finds (based on the index name) to identify a record for update; if the index that the primary key constraint is based on is not the first index, rename this index in ODI to ensure it will be the first. Generally, the first unique index is the correct index to use to identify a record, in which case this error can be ignored.

- **"cannot CREATE UNIQUE INDEX; duplicate keys found"**

  Due to differences in patch and subversions for a given OLTP release, it is possible that the instance used when importing a unique index uses a different combination of columns than are used in your particular release of the same OLTP. For example, the OLTP subversion used to import an index uses 3 columns to define the unique index but a later subversion uses 4 columns, and you are using this later subversion. Check the definition of the unique index in your OLTP instance and modify the index script and corresponding constraint definition in ODI to match.

Setting up Ledger Correlation using GoldenGate

Reconcile ledger information available in your Oracle E-Business Suite and Oracle Fusion Applications using Oracle GoldenGate.

If you are analyzing data sourced from Oracle E-Business Suite and Oracle Fusion Applications and are using the Fusion GL Accounting feature, then you can drill in analyses from Fusion GL balances to related detailed EBS ledger information. This ledger correlation is supported by GoldenGate replication, and requires GoldenGate configuration on your source systems. For information about setting up GoldenGate on EBS and Fusion Applications, see: *Configuring Oracle GoldenGate to Integrate the E-Business Suite General Ledger with Fusion Accounting Hub*, My Oracle Support Document 1324692.1.

After setting up the required GoldenGate configurations on the sources, you need to expose the following Presentation columns in the applicable reports to support drilling:

- **Target GL Account ID** for **GL Account** of subject area **Financials – GL Balance Sheet**.
- **Target Ledger ID** for Ledger
- **Target Fiscal Period ID** for Time
- **Dim – Ledger.Source ID** (Data Source Num ID)

For information about working with Presentation columns, see Creating and Maintaining the Presentation Layer, *Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition*.