Oracle Utilities Analytics

Administration Guide

Release 2.5.1

E48998-03

October 2014

Oracle Utilities Analytics, Release 2.5.1

E48998-03

Contents

Contents

Preface

This guide provides instructions for configuring and administering Oracle Utilities Analytics (OUA), including:

- **Audience**
- **Prerequisite Knowledge**
- **How This Guide is Organized**
- **Related Documents**
- **Conventions**
- **Acronyms**

Audience

This administration guide is intended for anyone interested in the process of configuring and administering Oracle Utilities Analytics.

Prerequisite Knowledge

Oracle Utilities Extractors and Schema and Oracle Utilities Analytics Dashboards use several technologies. You should have knowledge of the following before configuring and administering Oracle Utilities Analytics:

- Oracle Data Warehouse concepts:

  http://docs.oracle.com/cd/E11882_01/server.112/e25554/toc.htm
- Oracle Warehouse Builder:

  http://docs.oracle.com/cd/E11882_01/owb.112/e10581/toc.htm
- Oracle Data Integrator:

  http://docs.oracle.com/cd/E21764_01/integrate.1111/e12641/overview.htm
- Oracle GoldenGate:

  http://docs.oracle.com/cd/E35209_01/doc.1121/e29397.pdf
- Oracle WebLogic Server:

  http://docs.oracle.com/cd/E15051_01/wls/docs103/pdf.html
- Oracle Business Intelligence Enterprise Edition:

  http://docs.oracle.com/cd/E28280_01/bi.1111/e10544/toc.htm

How This Guide is Organized

This guide helps you configure and administer Oracle Utilities Extractors and Schema v2.5.1 and Oracle Utilities Analytics Dashboards v2.5.1. This guide refers to these two products together as Oracle Utilities Analytics. If any topic is specific to only one of the products, then it is specifically mentioned.

Chapter 1 describes the new features and functionalities and new changes added in this release.

Chapter 2 provides a brief overview of the various components of a data warehouse.

Chapter 3 provides an overview of Oracle Utilities Analytics and its components, such as data warehouse, Extract, Transform and Load (ETL) and analytics. Make sure that you are familiar with these concepts before you proceed to configure Oracle Utilities Analytics.

Chapter 4 explains how to configure Oracle Utilities Analytics once it is installed. This configuration is mandatory for proper and expected functioning of Oracle Utilities Analytics. This is a must read before you start the implementation.

Chapter 5 explains how to extend Oracle Utilities Analytics. Oracle Utilities Analytics provides the out of the box star schemas, extract- transform - load programs and analytics. This chapter explains how you can extend Oracle Utilities Extractors and Schema by utilizing user-defined fields. This is the only supported means of extending Oracle Utilities Extractors and Schema.

Chapter 6 explains how to add new requirements and how to add a new star schema to Oracle Utilities Analytics. Please note that any new requirements are maintained by you.

Chapter 7 describes methods to maintain the environment. The methods explained in this chapter help users in maintaining their environments easily. It is recommended that you refer to the related product documentation first, and then read this chapter for a complete understanding.

Chapter 8 describes various licensing restrictions while using some of the Oracle products, such as Oracle database, Oracle Warehouse Builder, Oracle Data Integrator and Oracle GoldenGate. Out of the box functionalities are covered under the license agreement described as part of this product licensing. If your business practices require that you create additional functionality using these products, verify that modification are covered under the licensing agreement, or that you already have the additional licenses.

Related Documents

The following documentation is included with this release:

- *Oracle Utilities Analytics User's Guide*
- *Oracle Utilities Analytics Installation Guide*
- *Oracle Utilities Analytics Quick Install Guide*
- *Oracle Utilities Analytics Release Notes*
- *Oracle Utilities Analytics Dashboards for Oracle Utilities Meter Data Analytics Metric Reference Guide*
- *Oracle Utilities Analytics Dashboards for Oracle Utilities Customer Analytics, Revenue Analytics and Credit & Collections Analytics Metric Reference Guide*
- *Oracle Utilities Analytics Dashboards for Oracle Utilities Exception Analytics Metric Reference Guide*
- *Oracle Utilities Analytics Dashboards for Oracle Utilities Mobile Workforce Analytics Metric Reference Guide*
- *Oracle Utilities Analytics Dashboards for Oracle Utilities Distribution Analytics and Outage Analytics Metric Reference Guide*
- *Oracle Utilities Analytics Dashboards for Oracle Utilities Work and Asset Analytics Metric Reference Guide*

- *Oracle Utilities Analytics Dashboards for Oracle Utilities Operational Device Analytics Metric Reference Guide*
- *Oracle Utilities Extractors and Schema for Oracle Utilities Customer Care and Billing Data Mapping Guide*
- *Oracle Utilities Extractors and Schema for Oracle Utilities Meter Data Management Data Mapping Guide*
- *Oracle Utilities Extractors and Schema for Oracle Utilities Mobile Workforce Management Data Mapping Guide*
- *Oracle Utilities Extractors and Schema for Oracle Utilities Network Management System Data Mapping Guide*
- *Oracle Utilities Extractors and Schema for Oracle Utilities Operational Device Management Data Mapping Guide*
- *Oracle Utilities Extractors and Schema for Oracle Utilities Work & Asset Management Data Mapping Guide*

Conventions

The following text conventions are used in this document:

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

Acronyms

The list of acronyms used in this guide is as explained below:

- **APEX**: Oracle Application Express
- **CC&B**: Oracle Utilities Customer Care and Billing
- **CDC**: Changed Data Capture
- **ELT**: Extraction, Loading and Transformation
- **ETL**: Extraction, Transformation and Loading
- **MDM**: Oracle Utilities Meter Data Management
- **MWM**: Oracle Utilities Mobile Workforce Management
- **NMS**: Oracle Utilities Network Management System
- **OBIEE**: Oracle Business Intelligence Enterprise Edition
- **ODI**: Oracle Data Integrator
- **ODM**: Oracle Utilities Operational Device Management
- **OGG**: Oracle GoldenGate

- **OUA**: Oracle Utilities Analytics
- **OWB**: Oracle Warehouse Builder
- **WAM**: Oracle Utilities Work and Asset Management

Chapter 1

What's New?

Oracle Utilities Analytics is a set of star schemas, graphic templates, and data processing programs that allows you to build a Business Intelligence (BI) solution to meet your organization's analytic requirements.

The product has started the process of migrating from Oracle Warehouse Builder (OWB) based ETL to Oracle Data Integrator (ODI) based ELT in the last few releases. This migration is planned in a phased manner, spanning across several releases over a period of time. During this migration cycle, ETL for some of the source applications will continue to use Oracle Warehouse Builder, while others will use Oracle Data Integrator.

In the release 2.5.1, Oracle Utilities Network Management System (NMS) source application will start using Oracle Data Integrator (ODI) based ELT. Earlier only Oracle Utilities Operational Device Management and Oracle Utilities Customer Care and Billing source applications used to support Oracle Data Integrator based ELT component.

> **Note**: For information about the new and enhanced products, see *Oracle Utilities Analytics Release Notes.*
>
> Visit My Oracle Support (http://support.oracle.com) for the most recent service packs and patches for this release to ensure you have the most current version of this product.

Chapter 2

Overview of Data Warehouse

The data warehouse is a database primarily used for reporting and analysis purpose. It is a central repository for the current as well as the historical data from various operational applications. Data from the transactional or the operational applications is extracted, transformed, and loaded into the star schema in the data warehouses.

The components of a typical data warehouse environment are:

- **Source Application** - The source application from where the data is transferred to the data warehouse.
- **ETL/ELT** - The Extract, Transform and Load (ETL) tools move the data from the source application to the target data warehouse.
- **Staging Area** - The staging area is a database that stores the raw data extracted from the source application.
- **Data Presentation Area** - The data storage architecture in the data warehouse such as the star schema.

Source Application

The day to day transaction of the system is captured in the source application. The data in the source application is stored in a way that allows for the fast reads and updates. Queries are not expected to be in the bulk form, but typically arranged, row by row. To achieve this, the data is stored in a normalized form in the source application. Generally, the source system does not store the historical data.

ETL/ELT

The process of identifying data to be transferred from the source to the data presentation area, extracting the data, and transforming and loading into the data warehouse is called Extraction, Transform and Load (ETL). There are many tools available in the industry. Based on the complexity of the ETL, you need to choose the tools or write the ETL programs.

Staging Area

The data warehouse stores data from various operational sources. The staging area is a database that stores the raw data extracted from the source application. The replication area is similar to the source application in the structure. This area can be used for cleansing the data, resolving domain conflict, merging or combining data from the multiple source applications, data duplication, etc.

Data Presentation Area

Once data is cleansed in the staging area, it is ready to be moved to presentation area. The data is organized in such a way that it is easily available for the presentation. The data is stored in dimensional model, typically in the star schema.

Chapter 3

Overview of Oracle Utilities Analytics

This chapter provides overview of Oracle Utilities Analytics (OUA), including:

- **What is Oracle Utilities Analytics**
- **Overview of Analytics**
- **Data Processing**
- **Oracle Warehouse Builder Based Extract, Transform and Load (ETL)**
- **Oracle Data Integrator Based Extract, Load and Transform (ELT)**

What is Oracle Utilities Analytics

Oracle Utilities Analytics consists of the pre-built analytics applications as well as a collection of the Extractors and Schema products. Oracle Utilities Analytics helps you extract, transform, load, and analyze the data generated in Oracle Utilities Applications, such as Oracle Utilities Customer Care and Billing (CC&B), Oracle Utilities Meter Data Management (MDM), Oracle Utilities Network Management System (NMS), Oracle Utilities Work and Asset Management (WAM), Oracle Utilities Mobile Workforce Management (MWM) and Oracle Utilities Operational Device Management (ODM).

Built on world class Oracle Business Intelligence Enterprise Edition (OBIEE) 11g platform with integrated spatial features, Oracle Utilities Analytics supports end to end analytic workflows including the ability to drill back to the source applications. The extractors and schema are designed with the pre-built mapping between the source and the target. They support schema extensibility with built-in user-defined fields, dimensions and measures.

Oracle Utilities Analytics data warehouse is a separate database from the operational database. The data warehousing involves large volumes of data used primarily for analysis. The data warehouse has the following features:

- The data structures are easily accessible by the end users for their reporting needs.
- The large volumes of data can be retrieved quickly. This in turn allows fast rendering of the graphics that showcase the Key Performance Indicators (KPIs).
- Oracle Utilities Analytics contains star schemas and graphics suited for data retrieved from the various Oracle Utilities edge applications.
- Oracle Utilities Analytics application also provides you with the ability to add additional star schemas and graphics as per your requirement using the required development tools.
- Oracle Utilities Analytics can be divided into two broad components, namely, the Extract, Transform, and Load (ETL) process and the Analytics. Oracle Utilities Analytics uses Oracle Business Intelligence Enterprise Edition as its Analytics tool. The ETL was done using Oracle Warehouse Builder until Oracle Utilities Analytics v2.4.0 release. The v2.5.1 release

has introduced Oracle Data Integrator based Extract, Load and Transform (ELT) for Oracle Utilities Operational Device Analytics, Oracle Utilities Customer Care and Billing Analytics, Oracle Utilities Outage Analytics and Oracle Utilities Distribution Analytics. In the upcoming future releases of Oracle Utilities Analytics, most of the existing ETL will be moved to Oracle Data Integrator ELT in a phased manner.

The following products use Oracle Warehouse Builder based Extract, Transform, and Load (ETL):

- Oracle Utilities Work and Assets Analytics
- Oracle Utilities Meter Data Analytics
- Oracle Utilities Mobile Workforce Analytics

The following product uses Oracle Data Integrator based Extract, Load and Transform (ELT):

- Oracle Utilities Operational Device Analytics
- Oracle Utilities Customer Care and Billing Analytics
- Oracle Utilities Outage Analytics
- Oracle Utilities Distribution Analytics

Overview of Analytics

Oracle Utilities Analytics comes with a rich set of pre-built analytics created using Oracle Business Intelligence Enterprise Edition (OBIEE). These analytics offer a detailed insight into vital Key Performance Indicators (KPI) for the various source applications.

These analytics are built on top of various star schemas that are delivered in the data warehouse for different source application. For better performance, these analytics have been optimized to retrieve data from specially designed materialized views in the data warehouse. These materialized views are optimized to summarize data efficiently and make data retrieval much faster.

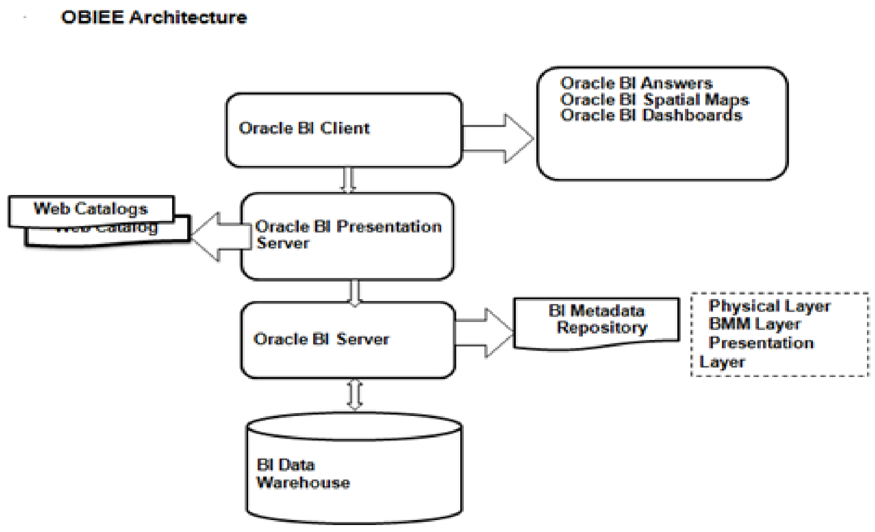
The pre-built analytics are created for each source application and grouped as separate catalog files as per the application. These product based catalogs are licensed individually.

Using the Oracle Business Intelligence Enterprise Edition Admin Tool, a metadata repository is first built using the star schemas available in the Oracle Business Intelligence Enterprise Edition data warehouse. This repository file consists of three distinct layers:

- **Physical Layer**: Contains information about physical data sources.
- **Business Layer**: Contains the business logic for the fields
- **Presentation Layer**: The tables grouped as subject areas for fact based reporting

Once the repository file is created and deployed on Oracle Utilities Analytics server, the dashboards are created based on the subject areas from the presentation layer. These dashboards are saved in the various web catalog files.

Below is a high-level architecture diagram for Oracle Business Intelligence Enterprise Edition components.

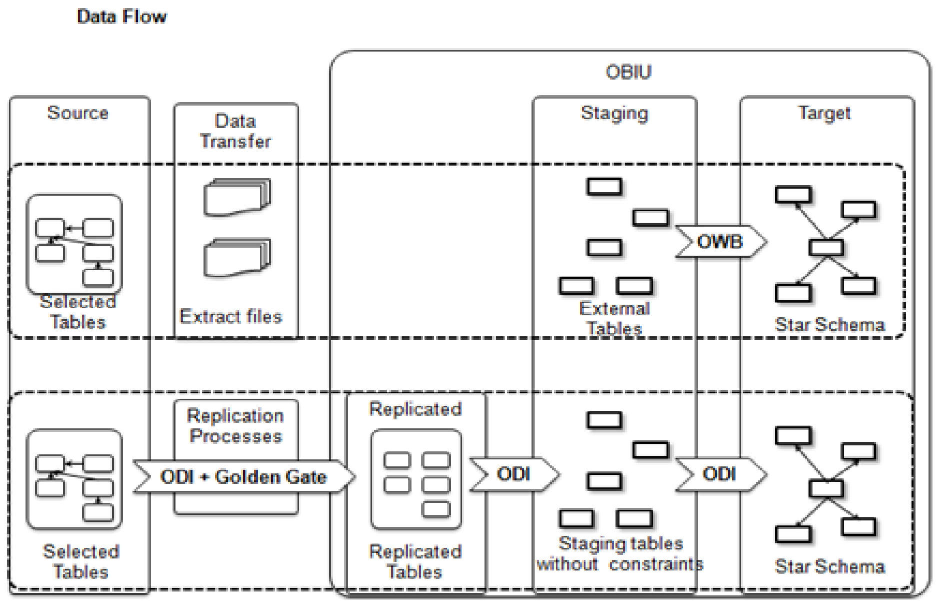


When Oracle Utilities Analytics is shipped for the analytics portion, the metadata repository and the source application specific catalog files are also delivered.

> **Note**: For details on Oracle Business Intelligence Enterprise Edition, refer to *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

Data Processing

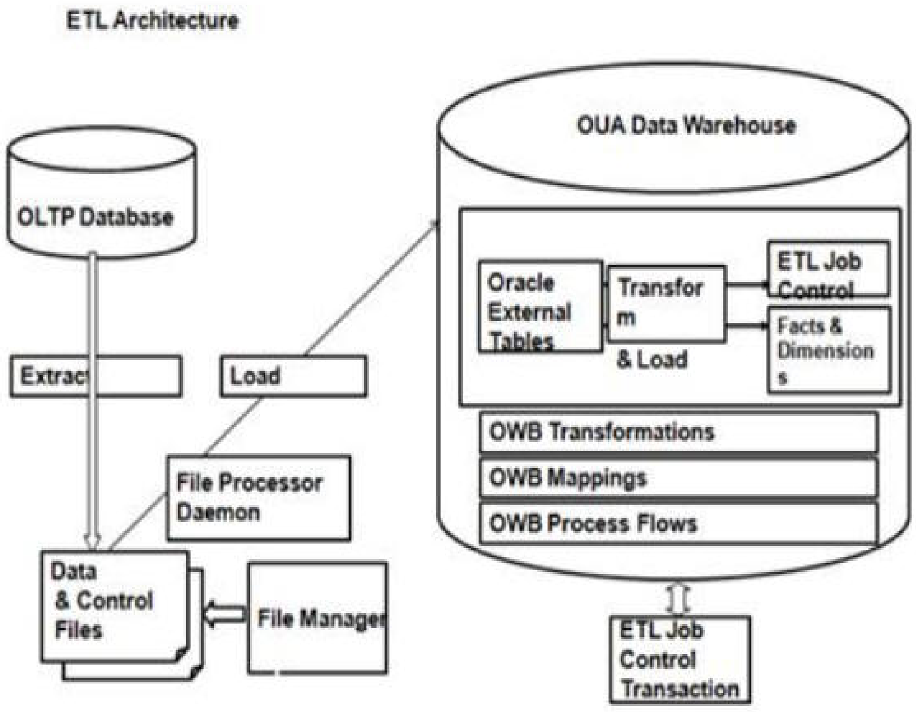
The architecture for Oracle Warehouse Builder based Extract, Transform, and Load (ETL) and Oracle Data Integrator based Extract, Load and Transform (ELT) is different. The diagram below illustrates the overall data processing pipeline using these two different methodologies.



The following sections give an overview of both the architectures.

Oracle Warehouse Builder Based Extract, Transform and Load (ETL)

The Oracle Warehouse Builder based methodology uses a three phase approach where the source system extractors extract data into flat files. These files are transferred to the target system and the Oracle Warehouse Builder process flows load the data into the target star schema.



Oracle Utilities Analytics uses Oracle Warehouse Builder to store the following items:

- The table designs of the star schemas.
- The data mappings used to generate batch jobs that perform the extract, transform and load operations.
- The process flows that validate the extracted information, load the star schemas and perform exception handling.

The following processes are involved in this methodology:

- **Extraction**
- **Transfer**
- **Load**
- **Aggregation**

Extraction

The extraction process executes on the source system and performs the transformations to convert data into a format that is accepted by the target data warehouse.

The extractors are provided for the Oracle Utilities suite of products. The extract programs execute in the operational database as they are extracting operational data. Oracle Utilities Analytics uses flat files as the source to load data into the data warehouse. The flat files are generated through an extraction process in the edge applications. Every fact and dimension in the data warehouse schema has a corresponding extract batch process. These batch processes extract data from the source system and transfer it to flat files. Along with each data flat file containing the extracted data, a single-record control file containing the batch information about the extractor program is also generated. The data and the control flat files, in turn, are loaded into Oracle Utilities Business Intelligence Enterprise Edition Data Warehouse.

Note: The Extractor programs are packaged with the edge applications.

Transfer

The data files generated on the source can be copied over to the target using FTP/SFTP/SCP or utilize a shared file location.

Load

The external tables in the target database point to these files and Oracle Warehouse Builder process flow loads the target data warehouse. The load processes perform the data validation and the reference key transformations.

The flat files produced by the extract programs serve as input to the load programs. The load programs use this data to populate the star schemas in the data warehouse.

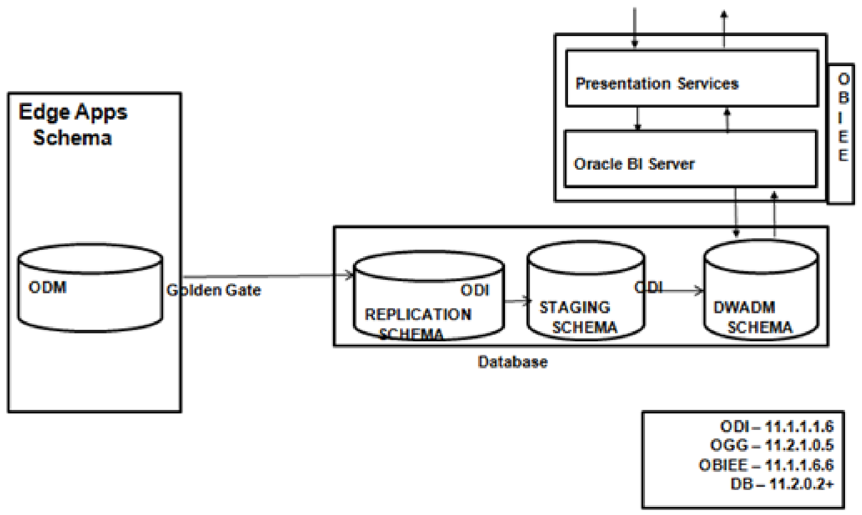
Note: Refer to the **Appendix I** for complete information on Oracle Warehouse Builder based ETL and its components.

Aggregation

The materialized views are used for storing aggregated data. However, the dependency on Oracle database's Query Rewrite feature has been removed, and Oracle Business Intelligence Enterprise Edition RPD is upgraded by providing a mapping between the subject areas and the different aggregations that can be utilized by fetching the data directly from the materialized views.

Oracle Data Integrator Based Extract, Load and Transform (ELT)

Oracle Data Integrator (ODI) is a comprehensive data integration platform that covers all data integration requirements ranging from high-volume, high-performance batch loads to event-driven and trickle-feed integration processes to Service-Oriented Architecture (SOA) enabled data services.



This methodology utilizes three key technologies from Oracle:

- Oracle Data Integrator
- Oracle GoldenGate
- Oracle WebLogic

The following concepts are involved in this methodology:

- **Replication**
- **Staging**
- **Target**
- **Data Cleansing**
- **Data Lineage**
- **Aggregation**

Replication

Oracle GoldenGate is utilized to replicate the selected tables from the source system into a replicated schema. Each source system instance has a dedicated schema associated for its replicated data. The replicated tables are similar in the structure to the source tables with a few additional columns added for tracking history and audit.

Oracle GoldenGate processes capture any changes on the source tables and transfer them to the replicated tables. The changes are available in the replication layer within few seconds, and the data can be used to load the target data warehouse. Also, later on, to reset and reload data in the data warehouse, the data in replication layer is available, provided the data were not purged.

A replication area is present on the target data warehouse to store data with history. The replication tables are classified into three categories:

- **Historized**: These are master data tables where a change in any column of the source is stored as a new row. If a previous row exists with the same natural key and effective end date as the last date, then that the row is end dated.
- **Overwrite**: These tables do not retain historical changes. Any change in the source is considered as an update and existing rows are updated. Deletions are also treated as an update and only the operation type is marked with a 'D' to indicate that the record has been deleted at the source. The transactional tables fall into this category.
- **Effective Dated**: The tables, which retain history in the source system. All changes are treated as updates. However, as source system does not store effective end dates; this value is calculated during sync.

The data area is also used to flatten out CLOB xml attributes into individual columns. The flattening out of CLOB into individual columns is done via views. These views also reside on the same physical schema on the database, but have a separate model in Oracle Data Integrator. It combines data from multiple tables together into a unified view that copies the target fact or dimension structure.

Staging

The staging tables are structurally similar to the target table and contain a few additional columns for the data transformations. There are no constraints on the staging tables. The foreign key mapping and other transformations are performed in the staging area. The data retention in the staging area is controlled by configuration stored in the metadata.

Target

The final target is the data warehouse entities, namely, dimensions and facts.

Note: For additional information, refer to the respective Data Mapping Guides for the source applications' specific star schema mapping.

Data Cleansing

The current release of Oracle Utilities Analytics paves the way to enable custom data cleansing routines to be implemented. To support this feature, a column "**cleansed_flg**" is provided on the replicated tables. The value is set to '**Y**' as default. The future release versions of Oracle Utilities Analytics will provide a framework to utilize the cleansing feature for implementing custom cleansing rules. At present, the data load process does not check this flag while loading the data.

Data Lineage

When data is loaded from the source system into the data warehouse using Oracle Data Integrator (ODI), it is possible to use Oracle Data Integrator Lineage for Oracle Business Intelligence feature to consolidate Oracle Data Integrator metadata with Oracle Business Intelligence Enterprise Edition and expose this metadata in a report to the Source Data Lineage Dashboard in Oracle Business Intelligence Enterprise Edition.

Note: For configuration of Oracle Data Integrator Lineage, refer to the **Oracle Business Intelligence Enterprise Edition Data Lineage** section in the *Oracle Data Integrator 11g New Features Overview* white paper as this feature is not provided out of the box.

Aggregation

In general, most facts have high volume of data, and multiple dimensions for slicing and dicing data. This can potentially lead to performance issues in the summarized dashboards. To mitigate risks associated with performance, Oracle Utilities Analytics utilizes Oracle Materialized Views to pre-aggregate the data. Oracle Data Integrator interfaces are used to build the materialized views.

For refreshing the materialized views, the following options are supported:

- **Scheduled**: The Materialized Views refresh is scheduled on a periodic basis to refresh the data in the materialized views based on the Oracle Integrator Scheduler using the configuration stored in metadata.
- **Dependency**: The Oracle Data Integrator based scheduling mechanism is utilized to refresh materialized view after any of the dependent object been loaded.

There is no dependency on Oracle database's Query Rewrite feature. A mapping between the subject areas and the different aggregations is present to upgrade Oracle Business Intelligence Enterprise Edition RPD for fetching the data.

Chapter 4

Configuring Oracle Utilities Analytics

Before you start using Oracle Utilities Analytics, it needs to be configured.

Follow the below-mentioned sections to configure the source system, ETL (Extract, Transform and Load) and analytics. It is extremely critical that you configure your system correctly for the product to behave as expected.

- **Configuring the Source System**
- **Configuring the ETL/ELT**
- **Configuring the Analytics**
- **Configuring the Database**

Configuring the Source System

Based on the product and its associated Extract, Transform and Load (ETL)/Extract, Load and Transform (ELT), you must have few configurations in place in the source system:

- **Oracle Warehouse Builder Based Extraction**
- **Oracle Data Integrator Based Extraction**

Oracle Warehouse Builder Based Extraction

Oracle Utilities Work and Assets Management (WAM), Oracle Utilities Meter Data Management (MDM), Oracle Utilities Mobile Workforce Management (MWM) use Oracle Warehouse Builder based extraction. For each dimension and fact in the data warehouse, there is a batch program in the source system that extracts and transforms the required data based on the type of extract and parameters passed to the batch program. These batch programs extract data and place them in the flat files. Make sure that these batch programs are configured on the source system.

A list of all the batch programs for the target entities is provided in the Appendices provided with this guide (starting from the **Appendix D** to the **Appendix F**). Other than the batch configuration, there may be additional configuration required on the source application.

> **Note**: For details on how to setup each of the source applications for data extraction, refer to the **Configuration** chapter in Data Mapping Guides for the respective source application.

Oracle Data Integrator Based Extraction

Oracle Utilities Network Management System (NMS), Oracle Utilities Operational Device Management (ODM) and Oracle Utilities Customer Care and Billing (CC&B), which use Oracle Data Integrator based extraction, need certain parameters to be configured in the source system. Make sure that those are setup correctly.

Note: For more details on the parameters, refer to the **Configuration** chapter in the Data Mapping Guides of the respective source applications.

Configuring the ETL/ELT

Depending on the product you use, you need to configure Oracle Warehouse Builder (OWB) based ETL (Extract, Transform and Load) or Oracle Data Integrator (ODI) based ELT (Extract, Load and Transform).

This section describes the following:

- **Oracle Warehouse Builder Based ETL**
- **Oracle Data Integrator Based ELT**

Oracle Warehouse Builder Based ETL

The ETL for the following source applications are based on Oracle Warehouse Builder:

- Oracle Utilities Work and Assets Management
- Oracle Utilities Meter Data Management
- Oracle Utilities Mobile Workforce Management

Once the extractor programs have generated the flat files for facts and dimensions, Oracle Warehouse Builder jobs pick these files and load the data into the data warehouse.

This section describes the following:

- **Scheduling Jobs Using the File Processor Daemon**
- **Reviewing the Log File**
- **Monitoring the Jobs**
- **Handling the Load Errors**
- **Capturing the Fact Load Errors**
- **Fixing the Load Errors**
- **Resubmitting a Failed Job**
- **Handling the Custom Errors**
- **Configuring the Multiple Data Source Indicator**

Scheduling Jobs Using the File Processor Daemon

The File Processor daemon is a simple java based utility that has the capabilities of a job scheduler. This persistent process runs continuously in the background and periodically monitors the extract folder. When the new data files arrive, it processes them and triggers the appropriate Oracle Warehouse Builder process flows for loading the data.

It determines the fact /dimension dependency. When a fact data file arrives in the extract file directory, it scans the extract directory to see if there are any data files present for any of the dimensions associated with this particular fact. If so, the loading of this particular fact file is skipped to let the dimension data load first into the data warehouse.

The File Processor daemon also scans the error folder to verify whether any of the dimension load jobs have failed. If there are failures, all the related fact data files are skipped from processing until the related dimensions are loaded successfully. The fact / dimension dependency is determined through the database constraints table present in the data warehouse.

The mappings in the parameter file control how the File Processor daemon can determine which Oracle Warehouse Builder process to trigger for which data file and which table name to check while querying the constraints table for fact/dimension dependencies. For more details, refer to the detailed description of the parameter **extract.file.mappingN** mentioned below.

The installation of the standard setup for the File Processor daemon is documented in *Oracle Utilities Analytics Installation Guide.* All the standard process flows are configured to run with the base installation.

> **Note**: For information on how to install and run the File Processor daemon, refer to *Oracle Utilities Analytics Installation Guide.*

The File Processor daemon reads a parameter file "**SchedulerParm.properties**", existing in the directory where the File Processor daemon is installed. The released version of this properties file includes entries for all the standard process flows. Hence, if any of the base extract files are present in the extract load directory, these are processed automatically. No extra configuration needs to be done if only base extracts are being implemented.

The following required parameter entries are present in the delivered **SchedulerParm.properties** file, and can be modified if needed for an implementation by using the **configureEnv.sh/cmd** command as documented in *Oracle Utilities Analytics Installation Guide:*

Parameter Name	Description
execution.switch	Determines if the File Processor daemon is active, or inactive. As long as this parameter is set to 1, the File Processor daemon continues to run. To stop the File Processor daemon without killing the process, modify this file and set the execution switch parameter to 0.
scheduler.poll.duration	Directs the File Processor daemon the number of seconds it should wait before polling the extract directory again to read the next set of the flat files to be loaded. By default, this is set to 60 seconds. However, based on the time it takes for the load jobs to complete. You can configure this value to a higher value (if load jobs take longer time), or a lower value (if load jobs complete real quick).
extract.dir.path	Directs the File Processor daemon where to look for new extract files. This path must match the path that the Oracle Warehouse Builder process flows have been configured to check for the extract files.

Parameter Name	Description
extract.max. parallel.threads	Directs the File Processor daemon how many types of the extract files (for fact or dimension tables) to load at a single time. If there are more files in the extract directory than the number specified by this parameter, the first set of files found are loaded with the current run of the File Processor daemon. This parameter can be modified based on the size of the machine/system and on the number of files that can be handled at once by Oracle Warehouse Builder. **Note**: For a single fact / dimension table, several flat files can be generated. However, multiple files for the same fact or dimension table are not loaded in parallel. The load jobs have been configured to pick one file at a time for a fact or dimension table. The order in which they are picked, are the same order in which they were extracted from the source.
extract.file.mapping.count	Provides the number of process flows listed in the parameter file. This count must match the largest **extract.file.mapping(N)** present in the parameter file. If this count is set to a lower number than the number of mappings, then only the set number of mappings are processed by the File Processor daemon.
extract.file.mappingN	**N** is a number between **1** and the **extract.file.mapping.count** parameter, or **extract.file.mapping.override.count** parameter if this is specified. These parameters indicate the File Processor daemon, which extract files to look for, which process flow to run when an extract file is found and the actual table name in the data warehouse. The format of this parameter is **Extract File Name**, **Process Flow Name**, **Table Name**. The extract file name is just the base name without the data source indicator, batch number, thread number values and without the **.DAT**, or **.CTL** extensions. For example, this entry looks for the **Account Extract** files and runs the **SPLWF_D_ACCT** process flow when a new account extract file is found: `extract.file.mapping1= D_ACCT_EXT,SPLWF_D_ACCT,CD_ACCT` The table name is used for determining dependency between the facts and dimensions by querying the database constraints table. This dependency check is required to avoid processing of any fact data files if any of its dimension data files are required to be loaded first. It is important when entering values that the **N numbers** increase sequentially and that no numbers are skipped. The largest **N value** matches the **mapping.count** or **mapping.override.count** parameter specified in the parameter file.

Note: The following below-listed optional parameter is not present in the delivered **SchedulerParm.properties** file, but can be added if needed:

Parameter Name	Description
extract.file.mapping.override.count	This parameter provides a way to override the mapping count specified in the **extract.file.mapping.count** parameter. The default parameter file does not include this parameter. If this parameter is specified, then the **extract.file.mapping.count** value is not used by the File Processor daemon. **Note**: When an override count is specified using this parameter, the additional mappings should also be specified after this using the previously explained parameter extract.file.mappingN.

The important fields in the parameter file to look at when implementing new loads are the mapping.count and the **mappingN** parameters. The new loads can be added to the parameter file if these are implemented by a project. It is recommended that the new mapping **N** values to be entered at the end of the list so that during an upgrade process, it is easy to identify the added records when updating the newly released **SchedulerParm.properties** file.

After updating any of these parameters using the **configureEnv** program, you must run the **initialSetup.sh/cmd** script to create the **cm_schedulerParm.properties.exit_1.include** parameter file. You must place this file in the **<INSTALL_DIR>/templates** directory, and then restart the File Processor daemon. Refer to the following section for the steps to be done for the customization.

Steps to Customize the File Processor Mappings

For adding new mappings so that they can be processed by the File Processor, perform the following steps:

1. Create the "**cm_schedulerParm.properties.exit_1.include**" file under the templates folder.

   For UNIX:

   $SPLEBASE/templates

   For Windows:

   %SPLEBASE%\templates

2. Add the following entry in "**cm_schedulerParm.properties.exit_1.include**" in the file:

   ```
   extract.file.mapping.override.count = <COUNT_NUMBER>
   ```

   Where COUNT_NUMBER is scheduler parameter mapping count from the source parameter file (viz. extract.file.mapping.count) + user configurable mappings count.

   For example:

   ```
   extract.file.mapping.override.count = 251
   ```

3. Add the new user parameter mappings in "**cm_schedulerParm.properties.exit_1.include**" file.

   Example:

   extract.file.mapping250 = <MAPPING1>

   extract.file.mapping251 = <MAPPING2>

4. Run the **initialSetup command**.

   For UNIX:

   ```
   $SPLEBASE/initialSetup.sh
   ```

   For Windows:

   ```
   %SPLEBASE%\initialSetup.sh
   ```

5. Start the **File Processor**.

   For UNIX:

   ```
   cd $SPLEBASE/bin
   nohup ksh ./startFileprocessordaemon.sh > ../tmp/
   consoleFileprocessordaemon.log 2>&1 &
   ```

   For Windows:

   ```
   startFileprocessordaemon.cmd
   ```

Reviewing the Log File

The instructions on how to start the File Processor daemon are explained in *Oracle Utilities Analytics Installation Guide*. Once started, a log file called **FileProcessorDaemon.log** is written by the File Processor daemon. Messages from the File Processor daemon are written here for various normal activities. The error messages are also written to this file. If Oracle Warehouse Builder process flows are not triggered properly, review this log file to possibly identify the cause of the problem.

The log file is deleted once it reaches a size of 100 megabytes and a new file is started. Hence, if older errors are not showing up, then it is possible that the file may have been recently purged by the File Processor daemon.

Monitoring the Jobs

Note: You must have a full license of Oracle Utilities Analytics to use this feature.

The process flows that are run by the File Processor daemon are set up in two different ways depending on whether a fact or a dimension extract file is being loaded. For a dimension load, only the dimension file is loaded into the dimension table of the data warehouse. However, when a fact file is loaded, the fact table is updated, and then any associated materialized views are refreshed.

The load jobs are visible in the **ETL Job Control Administration** portal. This can be accessed by logging into the **Oracle Business Intelligence Enterprise Edition Dashboard** portal and navigating to **Dashboard** > **Administration** > **ETL Job Control**. The **Job Complete (JC)** status indicates that the data file is loaded successfully. If a fact is loaded, the **Job Complete** status does not indicate that the materialized views are successfully updated. An e-mail is sent to the configured recipient if the materialized view refresh fails, or if any of the load jobs fails.

The following attributes are available in the **ETL Job Control** portal:

- **Job No**: Indicates the unique number identifying the load process. A link is present on the value if the job status is 'ERROR'.
- **Batch Code**: The unique code identifying the extraction routine used to generate the extract files on the source system.
- **Description**: A descriptive name for the extract process.
- **Batch No**: The batch number indicates the batch in which the extract is run.

- **Batch Thread**: A batch can generate multiple files. Each file is identifiable by a thread number.
- **Start Date/Time**: Indicates the start time of the load process.
- **End Date/Time**: Indicates the end time of the load process.
- **ETL Job Status**: The current status of the job.
- **Data Source Indicator**: The unique identifier for each source system.
- **Load Audit ID**: A reference to the Oracle Warehouse Builder audit log tables.
- **ETL Map**: The Oracle Warehouse Builder mapping name that contains the code to load the data into the target.
- **Extract Record Count**: The number of records in the extract file.
- **Load Record Count**: The number of records loaded into the target.
- **Load Error Count**: The number of records that failed.
- **Load Error Message**: Indicates the summarized error message identifying the cause of the job failure.

In addition, the load jobs and any associated errors can be viewed in Oracle Warehouse Builder Control Center. In general, if an e-mail message indicating a load failed is not received, then it indicates that the load of a data file (and any materialized view refresh) was successful.

Handling the Load Errors

While loading data from the flat files to the target schema using Oracle Warehouse Builder, there might be various reasons of a load failure. This section describes attributes to check when trying to find out why an extract does not load or why an error is generated during a load.

Note: For details regarding resolving Oracle Warehouse Builder load problems, refer to *Oracle Warehouse Builder User's Guide*. You should turn off File Processor daemon before debugging.

- **Job Status**: Make sure that the jobs are not in an **In Progress (IP)**. The status of a job is stored in the **JOB_STATUS_FLG** field in the **B1_ETL_JOB_CTRL** metadata table available in the DWADM schema, or you can view the status on the **ETL Job Control Administration** portal in **Oracle Business Intelligence Enterprise Edition Dashboard** (**Dashboard** > **Administration** > **ETL Job Control)**.
- **Oracle Warehouse Builder Errors Details**: There are two views that you can query to see errors from a process flow: ALL_RT_AUDIT_EXEC_MESSAGES and WB_RT_ERRORS available in the B1REPOWN schema. These errors are present in the e-mail messages sent when a mapping fails. However, you can run the following SQL statements in the DWADM schema to view the errors stored from the last four hours if the email messages are lost or do not contain any error messages:

```
    begin
owbsys.wb_workspace_management.set_workspace('SPLBIREP','BIREPOWN');
    end;
    -- where workspace name is SPLBIREP and workspace owner is
    BIREPOWN;
    -- replace if necessary the OWB workspace Owner in the query.
    To find the Workspace Name and Owner, run:
    select * from owbsys.WORKSPACE_ASSIGNMENT;
    select to_char( created_on, 'dd-mon-yyyy hh24:mi:ss - ' ) ||
    message_text
    from all_rt_audit_exec_messages where created_on > sysdate - .2
order by message_audit_id;
```

- **Error RPE-02248**: If this error is encountered, then change the **Runtime.properties** file in the directory: $ORACLE_HOME/owb/bin/admin.

```
    -- Change the below settings from DISABLED to NATIVE_JAVA
    property.RuntimePlatform.0.NativeExecution.FTP.security_constraint
= NATIVE_JAVA

property.RuntimePlatform.0.NativeExecution.Shell.security_constrai nt
= NATIVE_JAVA

property.RuntimePlatform.0.NativeExecution.SQLPlus.security_constr
aint = NATIVE_JAVA
```

- **Strange Characters in Data Files**: After querying the external table if strange characters are viewed in the result set, then it shows that the character set may be specified incorrectly for the external file. The character set can be changed by running the **EditFFCS.TCL** file in Oracle Warehouse Builder. You can see the character set for a specific external file by running the following query:

```
select * from dba_external_tables
where table_name = 'STG_ACCT_CTL_EXT';
```

  Note: For viewing the character set for a specific external file, specify the appropriate external table instead of the one mentioned in the example given above.

  Note: Refer to *Oracle Utilities Analytics Installation Guide* for more information on **EditFFCS.TCL** file and changing the character set.

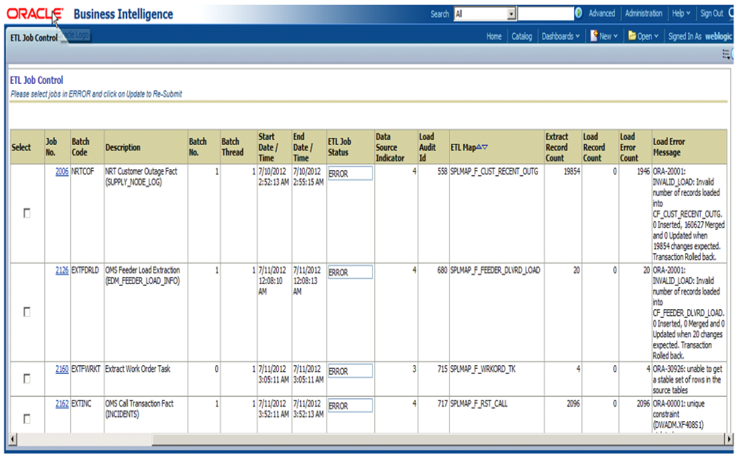
Capturing the Fact Load Errors

Apart from this out-of-the-box validation, Oracle Utilities Analytics also allows you to add additional validations to capture your business requirements.

In general, this process of capturing fact load errors is divided into two steps:

1. **Validation**: A validation function is invoked to execute the validation check. Oracle Utilities Analytics out-of-the-box validation validates the number of records loaded into the target file by comparing it with the number of records specified in the control file. You can write your own validation check if and when required. In case validation fails (as in case of out -of-the-box validation if number of records loaded into the target file is less than the number of records specified in the control file), the load is marked as an invalid load and a process to identify such records is executed.
2. **Error Identification**: This process executes certain queries to identify the records that failed to load and also the reason for the failure (missing reference records in a required dimension). All such records are inserted into a metadata table **B1_ETL_DATA_ERR** available in the DWADM schema. This process can be customized to add the logic for identifying such records and making an entry into the **B1_ETL_DATA_ERR** table. The out-of -the -box Oracle Utilities Analytics provides the error identification for foreign key reference only.

The following screenshot displays the reports, which are used to identify the failed load process:

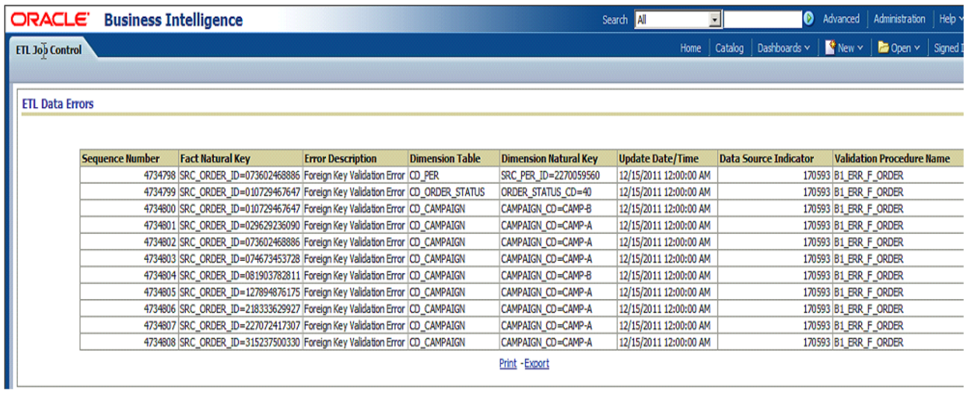

ORACLE Business Intelligence
Search All
Advanced | Administration | Help | Sign Out
ETL Job Control
Home | Catalog | Dashboards | New | Open | Signed In As weblogic

ETL Job Control

Please select jobs in ERROR and click on Update to Re-Submit

Select	Job No.	Batch Code	Description	Batch No.	Batch Thread	Start Date / Time	End Date / Time	ETL Job Status	Data Source Indicator	Load Audit Id	ETL Map	Extract Record Count	Load Record Count	Load Error Count	Load Error Message
☐	2006	NRTCOF	NRT Customer Outage Fact (SUPPLY_NODE_LOG)	1	1	7/10/2012 2:52:13 AM	7/10/2012 2:55:15 AM	ERROR	4	558	SPLMAP_F_CUST_RECENT_OUTG	19854	0	1946	ORA-20001: INVALID_LOAD: Invalid number of records loaded into CF_CUST_RECENT_OUTG. 0 Inserted, 160627 Merged and 0 Updated when 19854 changes expected. Transaction Rolled back.
☐	2126	EXTFDRLD	OMS Feeder Load Extraction (EDM_FEEDER_LOAD_INFO)	1	1	7/11/2012 12:08:10 AM	7/11/2012 12:08:13 AM	ERROR	4	680	SPLMAP_F_FEEDER_DLVRD_LOAD	20	0	20	ORA-20001: INVALID_LOAD: Invalid number of records loaded into CF_FEEDER_DLVRD_LOAD. 0 Inserted, 0 Merged and 0 Updated when 20 changes expected. Transaction Rolled back.
☐	2160	EXTFWRKT	Extract Work Order Task	0	1	7/11/2012 3:05:11 AM	7/11/2012 3:05:11 AM	ERROR	3	715	SPLMAP_F_WRKORD_TK	4	0	4	ORA-30926: unable to get a stable set of rows in the source tables
☐	2162	EXTINC	OMS Call Transaction Fact (INCIDENTS)	1	1	7/11/2012 3:52:11 AM	7/11/2012 3:52:13 AM	ERROR	4	717	SPLMAP_F_RST_CALL	2096	0	2096	ORA-00001: unique constraint (DWADM.XF408S1)

In the screenshot above, the jobs have been filtered by the status = 'ERROR'. The report shows the number of rows in the data file and the number of rows loaded. Click the **Job No**. field to navigate to that particular error details page. This error details page shows the data, which is not loaded due to the error as shown in the screenshot below:


ORACLE Business Intelligence
Search All
Advanced | Administration | Help
ETL Job Control
Home | Catalog | Dashboards | New | Open | Signed

ETL Data Errors

Sequence Number	Fact Natural Key	Error Description	Dimension Table	Dimension Natural Key	Update Date/Time	Data Source Indicator	Validation Procedure Name
4734798	SRC_ORDER_ID=073602468886	Foreign Key Validation Error	CD_PER	SRC_PER_ID=2270059560	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734799	SRC_ORDER_ID=010729467647	Foreign Key Validation Error	CD_ORDER_STATUS	ORDER_STATUS_CD=40	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734800	SRC_ORDER_ID=010729467647	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-B	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734801	SRC_ORDER_ID=029629236090	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-A	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734802	SRC_ORDER_ID=073602468886	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-A	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734803	SRC_ORDER_ID=074673453728	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-A	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734804	SRC_ORDER_ID=081903782811	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-B	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734805	SRC_ORDER_ID=127894876175	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-A	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734806	SRC_ORDER_ID=218333629927	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-A	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734807	SRC_ORDER_ID=227072417307	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-A	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER
4734808	SRC_ORDER_ID=315237500330	Foreign Key Validation Error	CD_CAMPAIGN	CAMPAIGN_CD=CAMP-A	12/15/2011 12:00:00 AM	170593	B1_ERR_F_ORDER

Print - Export

The screenshot above shows individual record level details.

The report provides the following details:

- **Fact Natural Key**: Indicates the unique key combination in the staging table.
- **Error Description:** Indicates the error type description.
- **Dimension Table**: The dimension table for which a reference could not be resolved in case the error is foreign key reference issue.
- **Dimension Natural Key**: The natural key columns and the data of the dimension, which are used to fetch the reference key in case the error is foreign key reference issue.
- **Update Date/Time**: The update date time in the extract file.
- **Data Source Indicator**: The data source indicator in the extract file.
- **Validation Procedure Name**: The procedure name that is used to identify these issues.

Fixing the Load Errors

This section describes ways to fix the issues identified during error identification process. If a data file fails to load, it is moved to the error directory just below the load directory. The load directory can be accessed from the database server.

You can determine the directory path using the following query from the **DWADM** account and with the **Control Table** name that is included in the failed record:

```
SELECT directory_path

FROM user_external_tables a, all_directories b

WHERE a.table_name = UPPER( '&Control_Table_Name' ) AND

b.directory_name = a.default_directory_name AND

b.owner = a.default_directory_owner;
```

The easiest way to examine the data file that has errors is to copy it into the Load directory so that Oracle queries can be run against it. The extract file replaces the **.DAT** file in that directory with the file name.

For example, the <DIRECTORY PATH>/error/ D_ACCT_EXT017933000000001001.DAT file replaces the <DIRECTORY PATH>/ D_ACCT_EXT.DAT file. The **.CTL** file in the processed directory can also replace the **.CTL** file in the Load directory.

Once you find the missing records, you can then determine if a required dimension key is missing from the dimension table.

> **Note**: It is possible that a later dimension load could have added the dimension records after the fact table was loaded, so if no records are returned by any of these queries, then reloading the fact records may solve the problem.

There are several different ways to fix the data so that it can be loaded. The method used to successfully load the data depends on what has caused the error:

- Make sure that the dimension files are loaded successfully. If there are errors during the dimension load or the number of records loaded did not match the number of records in the file, then you may need to figure out why the dimension records did not load. Fix those and reload the dimension files, before you can start working on the reloading fact records.
- Fix a fact data problem issue in the source system. This allows the data to be re-extracted the next time the fact data is extracted.
- Modify the fact records to have valid dimension keys. If the fact records are modified, then these must be added to a new extract file to get reloaded. A manual extract can be done to create a new extract file, and then the modified records can be loaded to this file. The modified records are then loaded when the next load process is run.

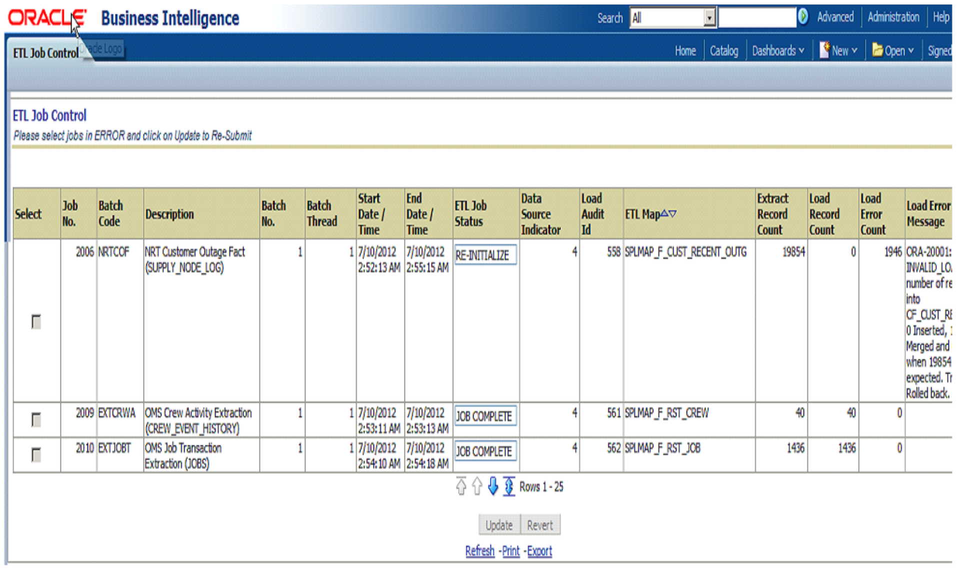
Resubmitting a Failed Job

> **Note**: You must have the full license of Oracle Utilities Analytics application to use this feature.

Resubmitting a Failed Job

Follow these steps to resubmit a failed job:

1. View the **ETL Job Control** page under the **Administration** menu of the dashboards.



ETL Job Control

Please select jobs in ERROR and click on Update to Re-Submit

Select	Job No.	Batch Code	Description	Batch No.	Batch Thread	Start Date / Time	End Date / Time	ETL Job Status	Data Source Indicator	Load Audit Id	ETL Map	Extract Record Count	Load Record Count	Load Error Count	Load Error Message
☐	2006	NRTCOF	NRT Customer Outage Fact (SUPPLY_NODE_LOG)	1	1	7/10/2012 2:52:13 AM	7/10/2012 2:55:15 AM	RE-INITIALIZE	4	558	SPLMAP_F_CUST_RECENT_OUTG	19854	0	1946	ORA-20001: INVALID_LO number of re into CF_CUST_RE 0 Inserted, Merged and when 19854 expected. Tr Rolled back.
☐	2009	EXTCRWA	OMS Crew Activity Extraction (CREW_EVENT_HISTORY)	1	1	7/10/2012 2:53:11 AM	7/10/2012 2:53:13 AM	JOB COMPLETE	4	561	SPLMAP_F_RST_CREW	40	40	0	
☐	2010	EXTJOBT	OMS Job Transaction Extraction (JOBS)	1	1	7/10/2012 2:54:10 AM	7/10/2012 2:54:18 AM	JOB COMPLETE	4	562	SPLMAP_F_RST_JOB	1436	1436	0	

Rows 1 - 25

Update Revert

Refresh - Print - Export

2. Select the job that needs to be restarted and click on **Update** button provided at bottom of the report. This changes the ETL job status on the screen to **RE_INITIALIZE**.

3. Move the data and control files that errored out from the Error folder to Load directory folder. This is very important as the file processor does not process the files that are in the error folder.

4. Place the corrected files in the data folder and switch on the File Processor daemon. The corresponding process flow is triggered during the next polling of the file.

Handling the Custom Errors

Note that the out of the box validation is always going to be executed during every fact load. In addition to this out of the box validation, a user exit hook has been provided to allow you to write your own Customer Modification (CM) validation functions and error identification procedures. These customer modified functions and procedure should be created in the DWADM account based on the naming convention. The out-of- the-box fact load procedure automatically executes these functions/procedures if these objects exist in the database.

The list of all the existing validation functions and error identification procedures is provided in the **Appendix H**. It also lists the custom function and procedure names to be used when extending the data load validation.

> **Note**: You must use the exact name provided in the **Appendix H** for your custom validations to be called while loading the fact.

In general, it is advisable to follow closely the out of the box routine for the custom code. The following guidelines have to be followed when creating custom validation functions:

- The validation function name is derived from the built in validation function by replacing '**B1**' with '**CM**' in the name.

  Note: Refer to the **Appendix H** for a complete list of validation functions for all facts. For example, for the fact **CF_CONSUMPTON**, the name of the out of the box validation function is **B1_VAL_F_CONSUMPTON**, and then the custom validation function name is **CM_VAL_F_CONSUMPTON**.

- The custom validation function has the same input parameters as those for the custom validation function.

Follow the below template to create customer validation function:

```
   CREATE OR REPLACE
   FUNCTION CM_VAL_F_ << Custom Validation Function Name >> (
p_num_inserts IN NUMBER, p_num_merged IN NUMBER, p_num_updates IN
NUMBER)
   RETURN NUMBER
   AS
   lv_ret_value NUMBER;
   BEGIN
   /* Your Logic Here */
   EXCEPTION
   WHEN OTHERS THEN
   lv_ret_value := 1 ;
   RETURN lv_ret_value;
   END;
   /*
```

The input parameters to the custom validation function are explained as below:

- **p_num_inserts**: Indicates the actual number of records inserted into target fact table by the mapping.
- **p_num_merged**: Indicates the actual number of records merged into target fact table by the mapping.
- **p_num_updates**: Indicates the actual number of records updated into target fact table by the mapping.

- The function returns **1** in case the validation fails and the load needs to be cancelled, and **0**, otherwise.
- The following guidelines have to be followed when creating custom error identification procedure. Always use the built-in procedure as a starting point:
- The procedure name is derived from the built in procedure by replacing '**B1**' with '**CM**' in the name.

  Note: Refer to the **Appendix H** for the complete list of error identification procedures.

  For example, for the fact **CF_CONSUMPTON**, the name of the out-of-the-box Error Identification procedure is **B1_ERR_F_CONSUMPTON**, and then the custom Error Identification procedure name is **CM_ERR_F_CONSUMPTON**.
- The custom procedures are the same input parameters as those of the out of the box procedures.
- Follow the below template to create the customer error identification procedure:

```
   CREATE OR REPLACE
   PROCEDURE CM_ERR_F<< Custom Error Handling Procedure Name >>
("IN_MAP_NAME" IN VARCHAR2, "IN_JOB_NBR" IN NUMBER)
   IS
   PRAGMA AUTONOMOUS_TRANSACTION;
   BEGIN
   /* Your Logic Here */
   EXCEPTION
   WHEN OTHERS THEN
   ROLLBACK;
   END;
```

The input parameters to the custom error identification procedure are explained as below:

- **IN_MAP_NAME:** Indicates the name of the mapping for which the procedure is getting executed for custom error record identification.

- **IN_JOB_NBR**: Indicates the job number of the particular mapping (as in **B1_ETL_JOB_CTRL** table) for which the procedure is getting executed for custom error record identification.
- The procedure must be executed as an autonomous transaction.
- The insert statement should be appropriately modified as per the custom validation implemented.

Configuring the Multiple Data Source Indicator

Oracle Utilities users can implement multiple utilities application leveraging the integration supported by these systems.

For example, Oracle Utilities Meter Data Management (MDM) and Oracle Utilities Customer Care and Billing (CC&B) joint implementations can leverage the MDM-CCB integration. Oracle Utilities Customer Care and Billing is used as the master repository for user related data (Account, Premise, Person, etc). So Oracle Utilities Meter Data Management instance does not store any user related information, but refer to the information in Oracle Utilities Customer Care and Billing instead. Similarly, Oracle Utilities Mobile Workforce Management (MWM) can be integrated with Oracle Utilities Meter Data Management and Oracle Utilities Customer Care and Billing. In this integration, Oracle Utilities Meter Data Management is used as the master repository for all the meter and measurements related data. In the Oracle Utilities Business Intelligence Enterprise Edition Data Warehouse, the data for these objects is loaded from the master repository only. For example, Account, Person, Premise dimensions are loaded only from Oracle Utilities Customer Care and Billing and Oracle Utilities Meter Data Management fact may have a reference to the dimensional data loaded from Oracle Utilities Customer Care and Billing.

In this case, while loading the Oracle Utilities Meter Data Management data, the load process looks up for the correct records in the dimension table loaded from Oracle Utilities Customer Care and Billing. Hence, there is a need to identify the Data Source Indicator (DSI) of Oracle Utilities Customer Care and Billing environment. The data source indicator is a unique value corresponding to each instance of an edge application feeding data into the data warehouse. The edge applications default the data source indicator from an environment ID. To support this requirement, Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management extract programs are enhanced to accept the additional parameters for Oracle Utilities Customer Care and Billing DSI and Oracle Utilities Meter Data Management DSI. These additional DSI values are appended into the control file of the extract files. During data load in the warehouse, the load process uses Oracle Utilities Customer Care and Billing DSI to lookup and joins to Oracle Utilities Customer Care and Billing specific dimensions, and Oracle Utilities Meter Data Management DSI to lookup and join to Oracle Utilities Meter Data Management specific dimensions.

> **Note**: Refer to the *Oracle Utilities Analytics ReadMe* document for the required Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management patches to support this feature.

Multiple data source indicator changes are covered in the detail below:

- **Multiple Data Source Indicator Changes for Oracle Utilities Meter Data Management Application**
- **Multiple Data Source Indicator Changes for Oracle Utilities Mobile Workforce Management**

Multiple Data Source Indicator Changes for Oracle Utilities Meter Data Management Application

This section describes how to configure the Oracle Utilities Customer Care and Billing DSI on Oracle Utilities Meter Data Management application. To configure, follow these steps:

Configuring Oracle Utilities Customer Care and Billing DSI on Oracle Utilities Meter Data Management Application

Perform the following steps:

1. Add a new feature configuration for the feature type '**Business Intelligence Configuration**'.
2. Select the option type '**External Data Source Indicator 1**' and enter the Oracle Utilities Customer Care and Billing DSI value. The Oracle Utilities Customer Care and Billing DSI value can be retrieved by running the below query in the Oracle Utilities Customer Care and Billing database:

   ```
   select ENV_ID from F1_INSTALLATION;
   ```

During the ETL process, if Oracle Utilities Customer Care and Billing DSI is set in the extract staging control file, it is used to join with the matching DSI in the dimension table to match the dimension key. If the Customer DSI has not been extracted in the staging control file, then the default data source indicator are used.

For Oracle Utilities Meter Data Management source application, Oracle Utilities Customer Care and Billing DSI joins with Oracle Utilities Customer Care and Billing related dimension keys, such as Account key, Person key, Premise key, and Service Agreement (SA) key for the following facts:

- CF_CONSUMPTION
- CF_DEVICE_ACTIVITY
- CF_DEVICE_EVT
- CF_INSTALL_EVT
- CF_SP
- CF_SP_SNAP
- CF_SP_UT_AGE
- CF_VEE_EXCP

Multiple Data Source Indicator Changes for Oracle Utilities Mobile Workforce Management

This section describes how to configure Oracle Utilities Customer Care and Billing DSI and Oracle Utilities Meter Data Management DSI on Oracle Utilities Mobile Workforce Management application:

Configuring DSI on Oracle Mobile Workforce Management (MWM) Applications

Perform the following steps:

1. Add a new feature configuration for the feature type '**Business Intelligence Configuration**'.
2. Select the option type '**External Data Source Indicator 1**' and for the **Value**, enter the Oracle Utilities Customer Care and Billing DSI. The Oracle Utilities Customer Care and Billing DSI can be retrieved by running the below query in the Oracle Utilities Customer Care and Billing database.

   ```
   select ENV_ID from F1_INSTALLATION;
   ```

3. Add one more entry with the option type as '**External Data Source Indicator 2**' and for the value, enter the Oracle Utilities Meter Data Management DSI. The Oracle Utilities Meter

Data Management DSI can be retrieved by running the following query in the Oracle Utilities Meter Data Management database:

```
select ENV_ID from F1_INSTALLATION;
```

Oracle Utilities Mobile Workforce Management application can be integrated with Oracle Utilities Customer Care and Billing and Oracle Utilities Meter Data Management, and therefore two separate data source indicator columns have been added to Oracle Utilities Mobile Workforce Management fact staging control tables.

Oracle Utilities Customer Care and Billing DSI joins with Oracle Utilities Customer Care and Billing related dimension keys, such as Account key, Person key, Premise key, and Service Agreement (SA) key. Oracle Utilities Meter Data Management DSI joins with Oracle Utilities Meter Data Management related dimension keys, such as Contact key, Meter Device key, SP key, and US key for the following facts:

- CF_CREW_TASK
- CF_FLD_ACTIVITY

Note: The multiple data source indicators support is a new enhancement included in Oracle Utilities Advanced and Operational Spatial Analytics (OUASA) release 2.4.0 service pack 4. This feature is not supported in earlier releases of Oracle Utilities Analytics (OUA).

In previous releases, it was suggested to make Oracle Utilities Meter Data Management DSI same as Oracle Utilities Customer Care and Billing DSI to achieve the same functionality.

After this enhancement, it is not required to match Oracle Utilities Meter Data Management DSI with Oracle Utilities Customer Care and Billing DSI. Therefore, Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management DSI values are defaulted from the environment ID and are no more configurable. For users who are upgrading to this enhancement, a special upgrade script is provided.

For more details, refer to *Oracle Utilities Analytics Installation Guide*.

Oracle Data Integrator Based ELT

The ELT for the following source applications are based on Oracle GoldenGate/Oracle Data Integrator:

- Oracle Utilities Network Management System (NMS)
- Oracle Utilities Customer Care and Billing (CC&B)
- Oracle Utilities Operational Device Management (ODM)

Oracle Data Integrator (ODI) based ELT uses Oracle GoldenGate and Oracle Data Integrator combination to extract, load and transform data from the source database to the target database. Oracle GoldenGate captures the changes in the selected tables of the source schema, and transfers these data to the replication schema. The tables in the replication schema are similar in the structure to the source tables with a few additional columns added for tracking history and audit. Data retention in the replication schema is controlled by configuration stored in the metadata.

Based on the scheduler configuration, data from replication schema is transferred by Oracle Data Integrator to the staging schema. The tables in the staging schema are similar to the tables in the target schema. It contains a few additional columns for data transformations. There are no constraints on the staging tables. The foreign key mapping and other transformations are performed in the staging area. The data retention in the staging schema is controlled by configuration stored in the metadata.

From the staging schema, Oracle Data Integrator transfers data to the facts and dimensions in the target schema. Oracle Data Integrator also loads and refreshes the materialized views in the target schema.

The required configuration for Oracle GoldenGate and Oracle Data Integrator is done during the Oracle Utilities Analytics installation process. Oracle Utilities Analytics provides an Administration Tool to maintain these configurations. The following user interfaces are available as part of the Administration Tool for maintaining various configurations.

- Product instance
- Oracle GoldenGate configuration
- Object map
- Global configuration
- Job Configuration
- Dependency
- Check point
- Target entity
- Source table
- Job execution (Display only)

> **Note**: For further information on how to use the Administration Tool, please refer to the **Appendix B**.

This section describes the following:

- **Scheduling**
- **Monitoring the Jobs**
- **Debugging**
- **Handling the Errors**
- **Reloading the Data**

Scheduling

In any data warehouse, the basic challenge is to get the data loaded quickly and efficiently whether it is the initial load or the incremental load. Data volumes are high during initial load and during incremental loads the volumes are considerably smaller. The Oracle Data Integrator based ELT processes utilize a time based slicing mechanism to split the load volumes into more manageable slices to process the initial as well as incremental loads. The metadata configurations allow you to control the size of the slice, parallelism of the loads and much more. To ensure that the data loaded is always consistent, the process executions are governed by a set of rules. The following criteria are considered for the job execution:

- There should be no errors, which need reprocessing.
- The maximum retries limit for the day should not be exceeded or reached.
- Tasks wait for the configured retry interval before submitting a retry for the job.
- Number of parallel jobs is always being limited to the maximum parallel configured.
- Jobs are not executed beyond the time of the most recent Oracle GoldenGate sync, or to a specified cut off time whichever is less.
- If a job is dependent on tables, which are being synced by separate Oracle GoldenGate processes, then the common sync time of both processes is considered.
- If a scenario does not exist, then the jobs are not executed.

- An interface should be active.
- All dependencies should be run.
- The number of running/error instances of the job should be less than the maximum parallel executions allowed.
- If the instance is configured as run once, then it should not execute once it is successfully executed.
- If a job fails, it should be retried again until the maximum retries per day is reached. The interval between successive retries should be based on configuration.
- Oracle GoldenGate models comprising of the source tables used in the entity should have been synced up. In case, the sync timestamps vary across multiple models, then the maximum sync timestamp is used.
- The snapshot entities are executed on or after the snapshot end period.
- The schedule time can be used to stagger loads and distribute processing. A job is not being executed until the current time crosses the scheduled time.

Note: Refer to the **Configuring the Target Entities** section in the **Appendix B** for where to schedule jobs.

All jobs executions are internally managed by the scenario named **B1_RUN_ALL**. This is scheduled using Oracle Data Integrator to run every 1-10 minutes as per your requirement.

Monitoring the Jobs

As jobs keep running on a regular basis, these are monitored or tracked to ensure that the required performance parameters are met. Jobs are created so that these are capable of automatic re-execution and retries. However, reasons for failure should be looked into and appropriate actions should be taken to resolve those issues. The following views are provided for achieving this:

View	Purpose
b1_jobs_vw	Provides a list of jobs executed with details of the slice start and end, scheduled execution time, actual start and end times, status, and record counts.
b1_config_vw	The consolidated view of the current configuration for all entities in the data warehouse.
b1_wait_reasons_vw	There are possibilities of erroneous configuration or other reasons due to which entities may not be running. To figure out what may be preventing a job from executing, have a look at this view's results.

Debugging

Oracle Data Integrator jobs are designed to make it easy to look at the data that was processed to figure out issues in the processing pipeline. To do this, the staging tables are utilized that retain data for a configured duration. Each execution results in some data that has been processed from the replication layer into the staging area, and then further on to the target entity. Each such set is stored in the staging table with the session number to identify the executing session.

At any point of time if a job fails or to look at how data was before it was loaded into the target, you can query the associated staging tables to look at the data.

The staging tables' names are derived by prefixing the target entity name with "**STG_**".

Handling the Errors

There are three basic components utilized in the data processing pipeline:

- Oracle GoldenGate for replicating changes from the source to the target
- Oracle Data Integrator to load data from the replication layer to the target
- Oracle database to store and manage data in the different layers. Oracle data pump is utilized to perform the initial sync between the source database and the target database.

There is a possibility of error occurring in one or more stages of the process. This section covers some of the details to help you identify the root cause and provides pointers to take appropriate action to resolve the issues that may arise:

1. **Failure of Initial Sync Processes**: In case of an error, you can execute the initial sync process again and it starts up from where it was left off. If the Oracle GoldenGate replication process is running while the initial sync is performed it is possible that the process will fail due to the duplicate keys. Here are the steps to resolve the issue:

   i) Stop the relevant replicat process.

   ii) Delete the data from the tables, which are listed in the error.

   iii) Execute the initial sync process again.

   iv) After the initial sync has been successfully loaded start the replicat processes.

2. **Failure of Oracle GoldenGate Processes**: Oracle GoldenGate processes can fail if the source or the target database goes down or the network connectivity fails between the source and the target. To identify whether Oracle GoldenGate processes are running properly, you should login to the server and run the command info all on the Oracle GoldenGate command prompt. If you see that any jobs shows up as ABENDED or STOPPED, verify the reason for the error by looking at the Oracle GoldenGate logs and start the jobs. Oracle GoldenGate processes will restart from the point of failure and continue the replication processes.

3. **Failure of Oracle Data Integrator Processes**: The Oracle Data Integrator processes have been designed to automatically retry for a configurable number of attempts per day. If the issue is related to a table spaces not available or database connectivity issues, the jobs would automatically execute once the issue has been resolved.

If the issue resolution is anticipated to take few hours, it is advisable to disable the failing jobs for the duration of the fix.

Query the view B1_JOBS_VW in the metadata schema to identify if any jobs have failed. In case of a failure, the status_flg column will show 'E'. You can log into Oracle Data Integrator and filter by the session number (available in the jobs view) and look at the error encountered.

Any failures will rollback any changes to the target. A late arriving dimension will be tagged to a default key (-99) in the dimension and these will be reprocessed in subsequent loads. If the dimensional data has arrived into the warehouse, then the foreign key references will be corrected automatically.

Reloading the Data

Data warehouses are usually designed based on the assumption that data is added only once to the warehouse. It is rare, although, sometimes to reset specific entities, the data is reloaded again. Oracle Utilities Analytics provides the functionality to reset and reload individual entities, or all data associated with a specific instance of a source system:

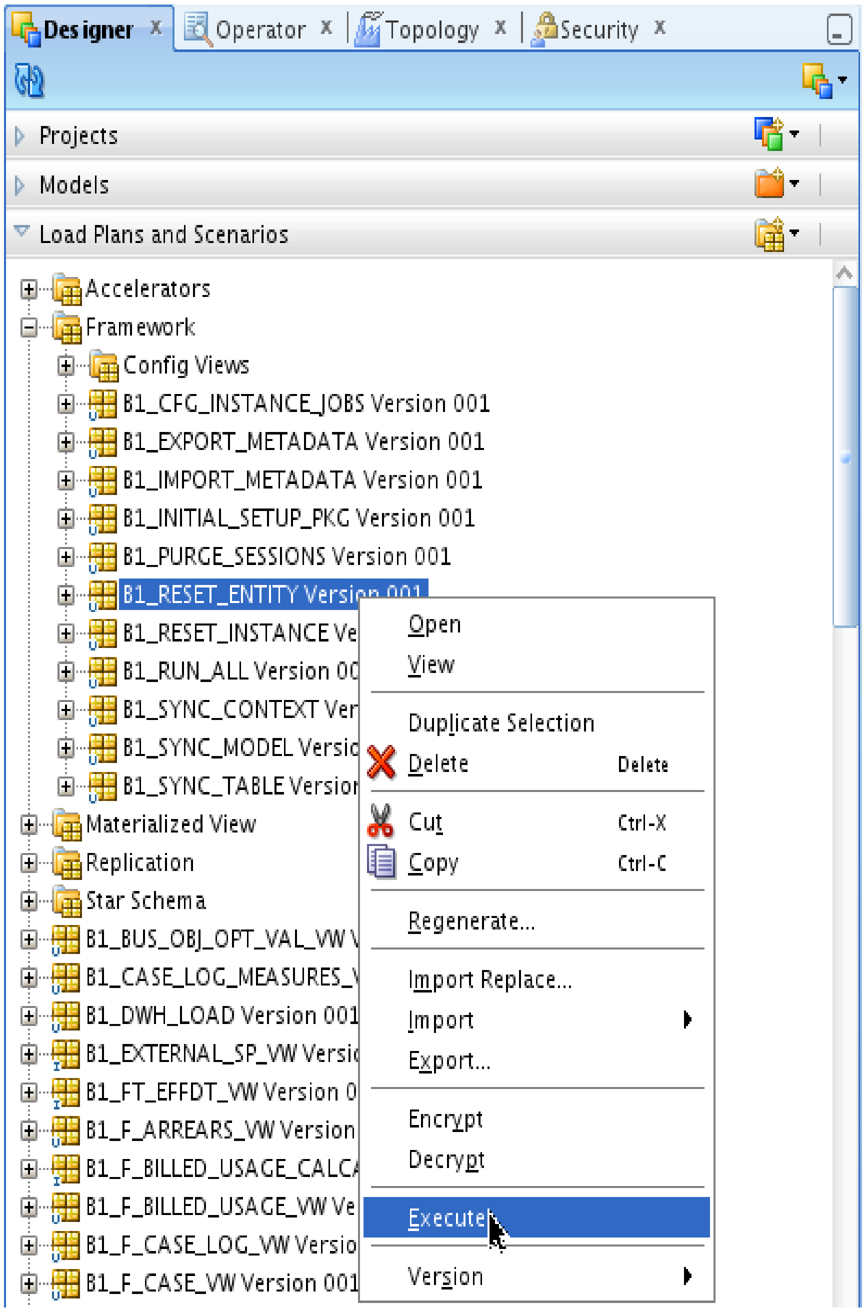
- **Resetting an Entity**
- **Resetting an Instance**
- **Resetting the Age Buckets**

- **Resetting Range Lookup for the Age Buckets**
- **Resetting the Extract Parameters**

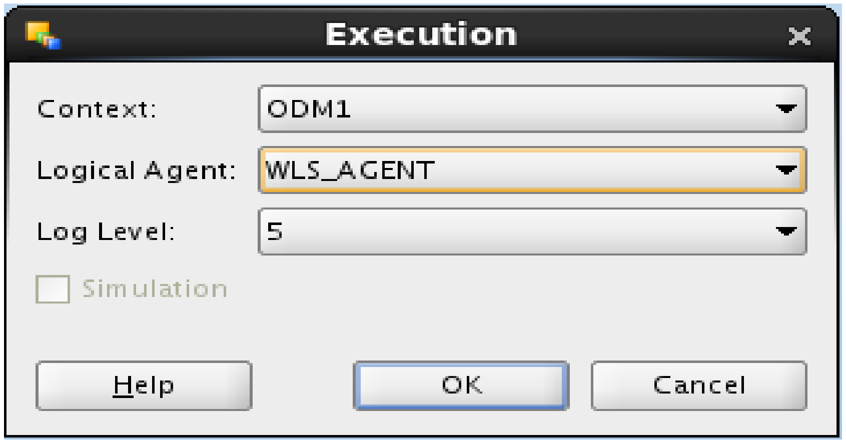
Resetting an Entity

Perform the following steps to reset an entity:

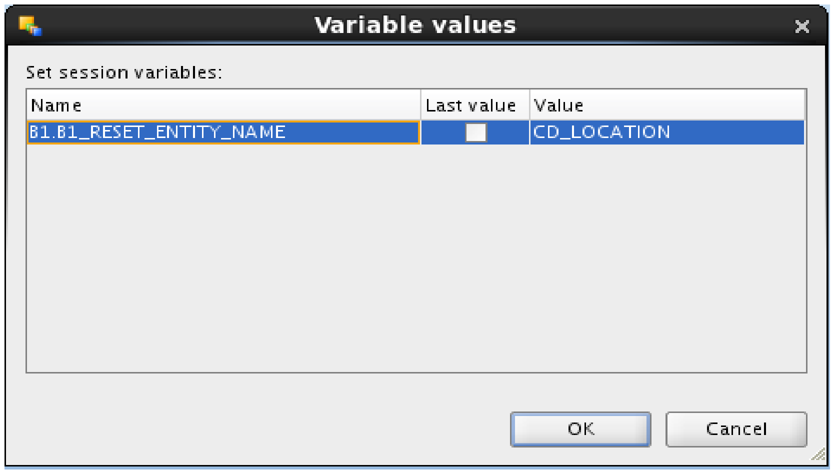
1. Login to **Oracle Data Integrator** client.
2. Navigate to the **Load Plans & Scenarios** section. Select the scenario named **B1_RESET_ENTITY**. Right-click and execute.



3. In the popup menu, select the context.
For example, to reset the fact for Oracle Utilities Device Management product instance 1, select the context **ODM1**.



4. Uncheck **Last Value** checkbox, specify the variable value, and click on the name field before clicking **OK**.



Name	Last value	Value
B1.B1_RESET_ENTITY_NAME	☐	CD_LOCATION

This cleans up the job executions and associated data. The entity is in disabled state after the job is completed.

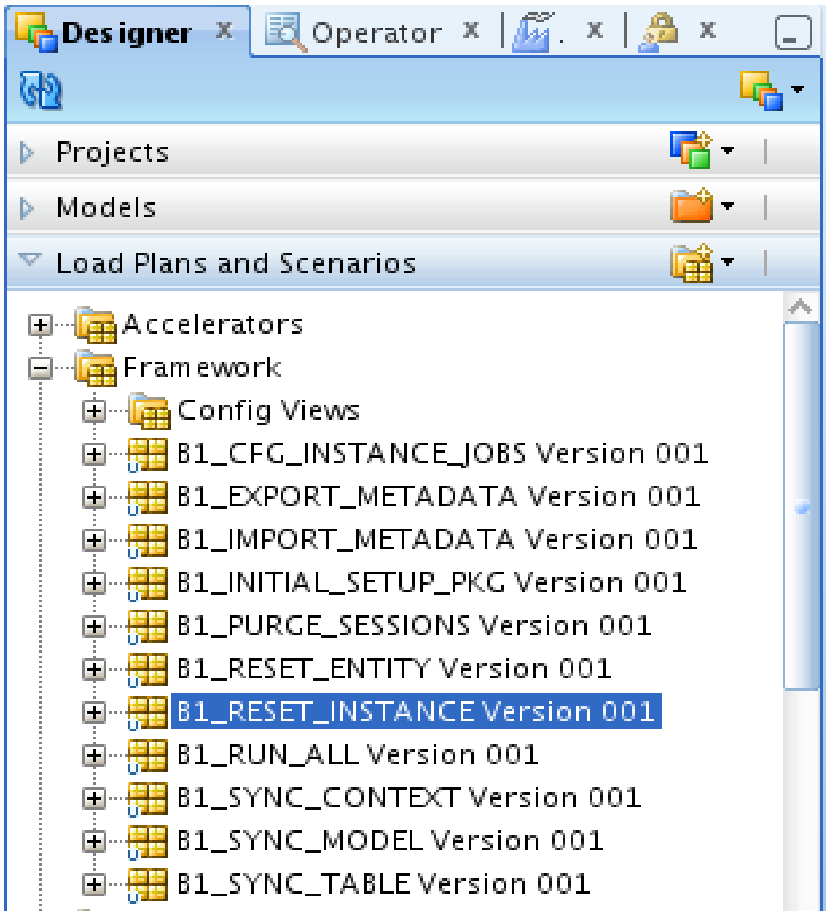
Resetting an Instance

Perform the following steps to reset an instance:

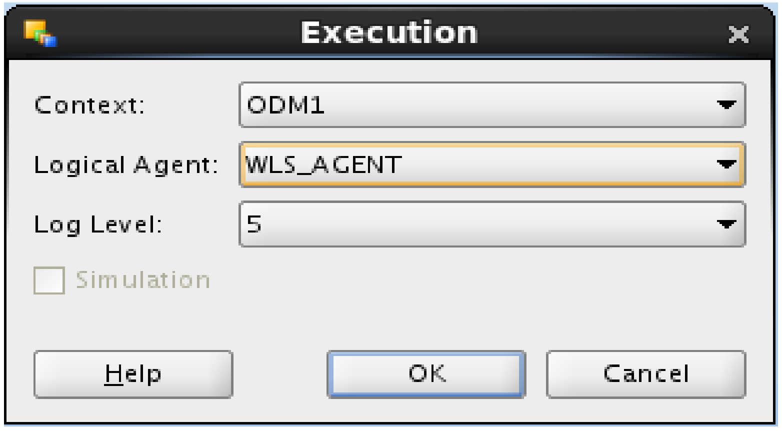
> **Note**: The steps shown below to reset an instance should be used with care. The replication layer data should not be purged already, or else, the target does not load with the expected amount of data.

1. Login to the **Oracle Data Integrator** client.

2. Navigate to **Load Plans & Scenarios** section. Select the scenario named **B1_RESET_INSTANCE**. Right-click and execute.


Designer
Operator
Projects
Models
Load Plans and Scenarios
Accelerators
Framework
Config Views
B1_CFG_INSTANCE_JOBS Version 001
B1_EXPORT_METADATA Version 001
B1_IMPORT_METADATA Version 001
B1_INITIAL_SETUP_PKG Version 001
B1_PURGE_SESSIONS Version 001
B1_RESET_ENTITY Version 001
B1_RESET_INSTANCE Version 001
B1_RUN_ALL Version 001
B1_SYNC_CONTEXT Version 001
B1_SYNC_MODEL Version 001
B1_SYNC_TABLE Version 001

3. In the popup, select the context.
For example, to reset all entities for Oracle Utilities Operational Device Management (ODM) product instance 1, select the context **ODM1**.


Execution

Context: ODM1

Logical Agent: WLS_AGENT

Log Level: 5

4. After verifying the configurations, enable all the entities to be load.

   Note: When a reset instance is done on the data warehouse, the materialized view refresh jobs that are triggered subsequently can result into an error. It would result in an materialized view refresh path error sometimes. This is an known behavior of the Oracle database. To workaround this issue, do a complete refresh of all the materialized views manually.

Resetting the Age Buckets

Several key performance indicators in Oracle Utilities Analytics look at measurement values and classify the value into an age range. The analysts can use these metrics to review the ages classified into different groups, such as 0-30 days, 30-90 days, or 90+ days. Such age ranges configuration is usually done before the implementation, but there may be scenarios where you would like to change these age ranges.

The ELT jobs are configured to be initial load only. Any incremental changes to these buckets after the initial data load are not reflected in the warehouse. However, if there is a need to reconfigure the buckets (If age buckets are associated with a dimension), perform the following steps:

1. Reset all the facts that are associated with the age bucket being re-configured using **B1_RESET_ENTITY** package.
2. Reset the dimension corresponding to the age bucket.
3. Reload the dimension.
4. Reload the facts.

Resetting Range Lookup for the Age Buckets

This section is applicable for the age buckets, which are referenced by the fact ELT jobs.

The age buckets configured in the source, are loaded in the **MDADM.B1_RANGE_LOOKUP** table in the data warehouse.

The ELT job for this is configured to be initial load only. Any incremental changes to these buckets after the initial data load are not reflected in the warehouse. However, if there is a need to reconfigure the buckets, perform the following steps.

1. Reset the facts that are associated with the age bucket to be re-configured. Refer to the section **Resetting an Entity** for more details. The fact names associated with various age buckets are mentioned in the table below.
2. Delete the entries from the **Range** lookup table for the specific age buckets using the business object name mentioned in the table below.
3. Re-configure the age buckets in the source.
4. To load the **Range** lookup table, run the view generator with the source context.

   Note: Refer to the section "**Generating the Views (For Source Instance)**" in *Oracle Utilities Analytics Installation Guide* for processing configurations and regenerating views.
5. Enable the fact ELT job using the **Administration Tool** to start the data load.

Age Bucket	Business Object Name	Fact Name
PA Future Payment Age Configuration	C1-PAFuturePaymentAge	CF_PA, CF_PA_SNAP
PP Future Payment Age Configuration	C1-PPFuturePaymentAge	CF_PAY_PLAN, CF_PAY_PLAN_SNAP
SA Arrears Configuration	C1-SAArrearsBuckets	CF_ARREARS

Resetting the Extract Parameters

If you need to modify the extract parameters defined on the source, perform the following steps:

1. Reset all the facts that are associated with the parameters being re-configured.
2. Modify the parameters on the source.

Note: Refer to the section "**Generating the Views (For Source Instance)**" in *Oracle Utilities Analytics Installation Guide* for processing configurations and regenerating views.

3. Reload the facts.

Configuring the Analytics

This section describes the various Administration Dashboards and the details about configuration of dashboards.

- **Maintaining the Administration Dashboards**
- **Configuring the Dashboards**

Maintaining the Administration Dashboards

The following dashboard pages are described in this section.

- **Maintaining the Base Fields**
- **Maintaining the Custom Fields**
- **Configuring the Source Drill Back**
- **Monitoring the ETL Jobs**

Maintaining the Base Fields

This dashboard allows you to provide an override description for the fields delivered along with the product. These fields contain descriptions, which appear on the report titles and the column titles for the dashboards delivered with the product.

Note: For the complete details on the field labels, refer to the **Configuring the Labels** in the section **Configuring the Dashboards**.

Maintaining the Custom Fields

This dashboard allows you to define additional fields and your description, which you may want to add as a part of customization.

This dashboard has the following pages to allow you to add, edit, and delete your own fields:

- Update
- Insert
- Delete

Note: For the complete details on the field labels, refer to the **Configuring the Labels** in the section **Configuring the Dashboards**.

Configuring the Source Drill Back

This dashboard page allows you to provide the source application details for Oracle Utilities Analytics integration with other application. The source application details, such as hostname, port number, and context are entered in this dashboard page. These details can be entered for Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management edge applications depending on which application is being used.

This information is used by reports in the detail pages for providing a link back to a specific source application page.

Note: For more detailed information on drill back configuration, refer to the section **Configuring the Drill Back**.

Monitoring the ETL Jobs

This dashboard page allows you to monitor the Oracle Warehouse Builder ETL jobs. You can track the status of the jobs along with its various attributes. During its lifecycle, when a job fails, you can resubmit the job for reprocessing using this page.

> **Note**: For more detailed information on **ETL Job Control** page, refer to the section **Monitoring the Jobs**.

Configuring the Dashboards

This section describes the configuration dashboards in Oracle Utilities Analytics, including:

- **Configuring the Drill Back**
- **Configuring the Oracle Utilities Meter Data Management Answers**
- **Configuring the Labels**
- **Configuring the Spatial Data**
- **Viewing the About Page**

Configuring the Drill Back

Oracle Utilities Analytics provides multiple **Drill Back** functionality from various reports in Oracle Utilities Analytics to the source applications, such as Oracle Utilities Meter Data Management (MDM), Oracle Utilities Mobile Workforce Management (MWM), and Oracle Utilities Customer Care and Billing (CC&B). You must configure **Drill Back** by providing the required information for this functionality to work.

The **Configuration Dashboard** under **Administration** group can be used for configuring various options in Oracle Utilities Analytics. The **Configuration** tab contains **Drill Back** settings for the source applications.

You can update the host name, port, and the context root folder for various edge applications with Oracle Utilities Analytics application through this page. After updating the values for the environment, the **Drill Back** links on various dashboard pages use these new values when an item is selected.

Currently, this drill back configuration supports the following edge applications:

- Oracle Utilities Meter Data Management
- Oracle Utilities Mobile Workforce Management

> **Note**: You are required to configure only those source applications for which you have implemented Oracle Utilities Analytics.
>
> **Note**: The **Drill Back Configuration** option for Oracle Utilities Network Management System, Oracle Utilities Customer Care and Billing is not available. Since the ELT methodology for these source products are based on Oracle Data Integrator, the source Drill Back URL is provided as a part of the **Product Instance** configuration in Oracle Utilities Analytics Administration Tool. For further information, refer to the **Configuring the Product Instance** section in the **Appendix B**.

Configuring the Oracle Utilities Meter Data Management Answers

The following Oracle Utilities Meter Data Management Answers should be configured before viewing the data:

- Tamper Events Answer (**Overview** dashboard)
- Usage Unreported for > 30 Days (**Overview** dashboard)
- Percent of Normal Intervals (**Overview** dashboard)

- Percent of On-Time Intervals (**Overview** dashboard)
- Degree Days (**Overview** dashboard)

After customizing the answers, save the reports in a separate CM catalog.

> **Note**: For details, refer to *Oracle Utilities Analytics Dashboards for Oracle Utilities Meter Data Analytics Metric Reference Guide.*

Configuring the Labels

This section describes how to create and customize the labels that appear in answers and dashboards.

> **Note**: You must have a full license of Oracle Utilities Analytics to use this feature.

This section includes the following topics:

- **Overview of Labels**
- **Overriding the Base Labels**
- **Supporting the Multiple Languages**

Overview of Labels

Oracle Utilities Analytics uses labels for the columns and the tables in the delivered Oracle Business Intelligence Enterprise Edition repository file when displaying the columns in the **Presentation** folders for you. These labels are displayed in the answers for the columns on the dashboards. In addition, the answers are also titled based on labels stored in the metadata displayed in report titles, sub-titles, and other dashboard strings.

The application uses labels instead of the hard coding the text values in the answers and the RPD file for supporting translation of the dashboards into the different languages and allowing easy overriding of the labels for users if you wish to customize the field label.

For example, within an answer, the labels can be referred to by Oracle Business Intelligence Enterprise Edition server variables. For example, the **Device Activity - Distributions** report uses this variable in the title section of the answer:

```
@{biServer.variables['NQ_SESSION.B1_RPT_DEV_ACTI_DISTRIBUTION']}
```

The **B1_RPT_DEV_ACTI_DISTRIBUTION** label is defined in the **B1_MD_FLD** table in **DWADM** schema.

For the columns in the fact and dimension tables, the labels exist for every field. For example, the **UDF1_DESCR** column in the **CD_ACCT** table has the description of the **Customer Class**, and the **Customer Class** label is displayed in the **Presentation** folder for this field.

Overriding the Base Labels

There are several reasons that an implementer may want to update an existing label:

- A field may contain data that does not match the default extracted data for that field.
  In the **CD_ACCT** example, described in the above section, you may elect to store information other than customer class in the **UDF1_DESCR** field. If an extract change is made to the default **CD_ACCT** extract, then an implementation change in the label for the **UDF1_DESCR** field of the **CD_ACCT** table at one place changes the label in all the dashboards and answers that display that field. This reason also applies if data is extracted to a User Defined Field (UDF) field that is not already having a default population.
- Even if you use the default extract code, you may choose to use some other name for the extracted data other than the default name.
  In the **CD_ACCT** example, if you execute the field extracted into the **UDF1_DESCR** field account class instead of customer class, you can make this change at one place and have it updated on all dashboards and answers.

- You may want to provide multilingual labels for your users. Oracle Utilities Analytics application provides the labels to a user based on the language selected when logging into Oracle Business Intelligence Enterprise Edition, assuming that the language is present in the **B1_MD_FLD** table. An implementation can add its own translated fields, or can download supported language packs from the Oracle Software Delivery Cloud.

  Note: The multilingual support is only provided for labels and not for the underlying data in the data warehouse. The data displayed in all the database tables is not translatable from the extract language.

Supporting the Multiple Languages

Oracle Utilities Analytics is released with default support for English labels on all the dashboards and answers. Both Oracle Business Intelligence Enterprise Edition and Oracle Utilities Analytics support the multiple languages.

The default language on the **Login** page is English. However, you can select any of the supported language on the **Login** page or can change the preferred language under the **Administration** menu to view dashboards in a different language. If you have not purchased and applied the specific language pack and if you select a language other than English, the default Oracle Business Intelligence Enterprise Edition labels is still be translated in the selected language, but Oracle Utilities Analytics product specific labels appear in English.

Oracle Utilities Analytics may release various language packs depending on user demands. Hence, for the language that is already released, installing the language pack is sufficient for creating the labels needed by the dashboards.

To view the list of Oracle Utilities Analytics language pack applied on an environment, you can navigate to the **About Oracle Utilities Analytics** dashboard under the **Viewing the About Page** in the Oracle Business Intelligence Enterprise Edition dashboards menu.

Contact your Oracle support representative to purchase a Oracle Utilities Analytics language pack for additional language support.

To update a label for a base field, use the **Base Field Maintenance** dashboard in the **Administration** portal.

- **Table Labels**: For records that have the **Table Name** field populated, but not the **Field Name,** this label is shown in the **Presentation** folder for the fields available in this table. For example, the **CD_ACCT** table has the label '**Account Dimension**' displayed in the **Presentation** folder wherever it is used.
- **Field Labels**: For records that have both the **Table Name** and **Field Name** fields populated, this label is shown in the **Presentation** folder and on answers whenever that field is used. For example, the **UDF1_DESCR** field in the **CD_ACCT** table has the label '**Customer Class**' displayed whenever it is used in an answer or when you select it from the **Presentation** folder when creating a new answer.
- **Report Labels**: Records that have a field name, such as 'B1_RPT%' and no table name value are used for the titles of answers in the dashboards. For example, the **B1_RPT_DEV_ACTI_DISTRIBUTION** label is defined to be '**Device Activity Distribution**', and this is displayed on the **Oracle Utilities Meter Data Management** dashboard when the answer is displayed.
- **Other Labels**: All other non-report labels that have a field name, but no table name is used for calculations that are computed in the RPD logical layer for display on answers. For example, the **B1_APPT_IND_SUM** label is defined to be '**Number of Appointments**', and is used in Oracle Utilities Mobile Workforce Management answers that compute the number of crew appointments based on the **Appointment Indicator** field in the **CF_CREW_TASK** fact table.

If a base field label should be changed, then the implementation team can query the required record on the **Base Field Maintenance** dashboard; populate a value in the **Override**

Description field, and click **Update**. Once populated, the Oracle Business Intelligence Enterprise Edition Server must be restarted for the changes to take effect.

Configuring the Spatial Data

This section describes how to configure mapping for Oracle Utilities Analytics. It includes the following topics:

- **Loading the Geographical Data**
- **Setting Up the Network Model Spatial Data in Outage Analytics**
- **Configuring the Google Map Tile Layer**
- **Implementing the Maps**

Loading the Geographical Data

In order to place information on a geographic map, data in the data warehouse must match geographic data (themes) that are configured in Oracle MapViewer.

The standard map answers delivered with Oracle Utilities Analytics include maps that query state, city, county, postal codes, and network model summary data. As Oracle Utilities Analytics does not have access to this spatial data (and as each user may require different spatial data), user must set up the geographic themes used in the maps.

> **Note**: For details regarding setting up these standard spatial themes, refer to *Oracle Utilities Analytics Installation Guide.*

The installation instructions refer to shape files downloaded from the US Census Bureau. However, the shape files can also be used for the state, city, county, and zip code boundaries. The only requirement is that the names of the geographic entities in the shape file should match the corresponding name in the **CD_ADDR** table. This is not usually a problem for postal code data, but can be an issue for city and county names, as different sources may use different names to refer to the geographic places. Make sure that after loading the **MapViewer shapefiles** that the names in the geographic tables match the names that are in the **CD_ADDR** table. If these do not match, then the maps may not display correctly.

Setting Up the Network Model Spatial Data in Outage Analytics

This section provides an overview Oracle Utilities Network Management System (NMS) network model representation in Outage Analytics.

Oracle Utilities Network Management System provides model build process to generate geometry data for various components of network model. The geographic data is stored in the DIAGRAM_OBJECTS table.

> **Note**: For details regarding the steps required to set up the Network Management System Model Build process, refer to *Oracle Utilities Network Management System Installation Guide.*

To plot these elements in Outage Analytics Dashboards, the DIAGRAM_OBJECTS table should be replicated as per the instructions available in the section **Installing US State Spatial Data** in *Oracle Utilities Analytics Install Guide.*

The proper mechanism should be in place to be up-to-date with the DIAGRAM_OBJECTS table in the Source database.

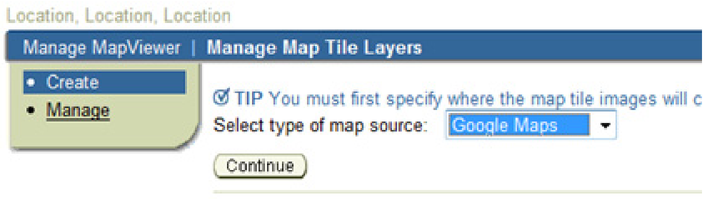
To ensure that the network model can be displayed without coordinate translations during runtime, one of the geometry columns should use the same projection coordinate system as the base map used by the Outage Analytics Outage Maps. Out of the box, Oracle eLocation is being used for the base map and the SRID (Spatial Reference System Identifier) is 54004. Note that the SRID is a unique value used to identify the coordinate system used in a Geographic Information System (GIS) application.

Configuring the Google Map Tile Layer

Out of the box Oracle Utilities Analytics is configured to fetch the map tiles from the eLocation. However, you can switch to Google as an alternative source for the map tiles.

Perform the following steps:

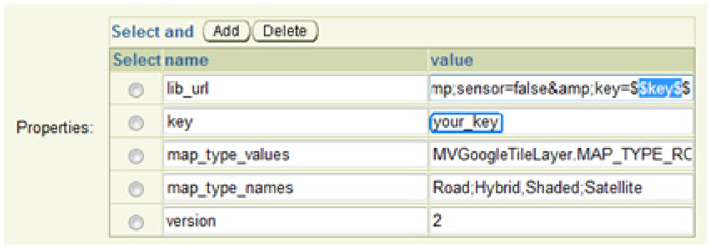
1. Log onto the **MapViewer** console.
   Sample URL:
   http://<host-name>:<port>/mapviewer
2. Click the **Admin** and enter the credentials.
3. On the **Management** tab, click the **Manage Map Tile Layers**.
4. Create a new tile layer by selecting **Google**.


Location, Location, Location
Manage MapViewer | Manage Map Tile Layers
Create
Manage
Continue

TIP You must first specify where the map tile images will c

Select type of map source: Google Maps

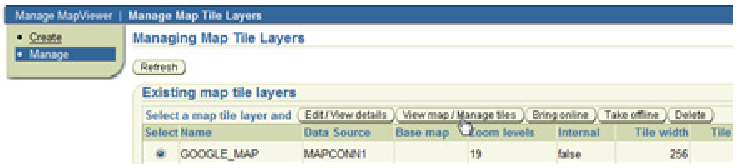
5. Enter the key fetched from Google in the **Key** field, edit the default **lib_url** to include the key value and choose the appropriate data source.



Properties:

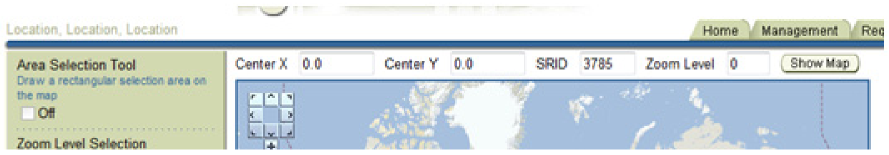
Select	name	value
○	lib_url	mp;sensor=false&amp;key=$$key$$
○	key	your_key
○	map_type_values	MVGoogleTileLayer.MAP_TYPE_RC
○	map_type_names	Road;Hybrid,Shaded;Satellite
○	version	2

6. Click **Submit** to save the information.



Select	Name	Data Source	Base map	Zoom levels	Internal	Tile width	Tile
●	GOOGLE_MAP	MAPCONN1		19	false	256	

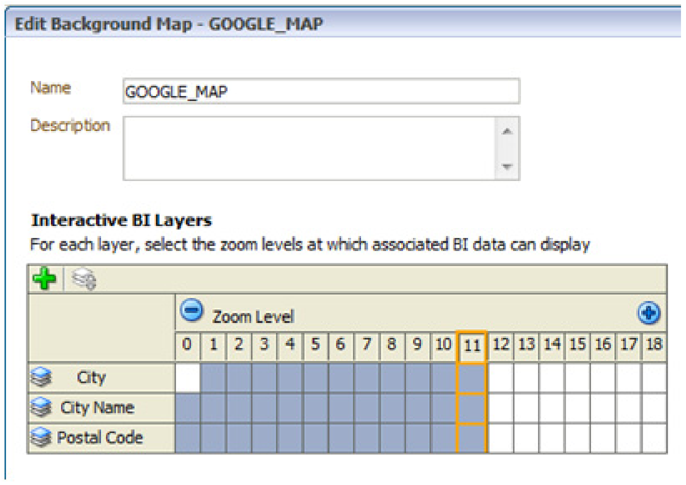
7. Click **View map/Manage tiles** to verify the new tile layer shows up properly.
8. Click **Show Map** to view the map.
9. Logon to Oracle Business Intelligence Enterprise Edition Analytics and navigate to **Administration > Manage Map Data** and click on the **Background Maps** tab.


Location, Location, Location
Home
Management
Req
Area Selection Tool
Draw a rectangular selection area on the map
Off
Zoom Level Selection
Show Map

Center X 0.0 Center Y 0.0 SRID 3785 Zoom Level 0

10. Import the tile layer '**GOOGLE_MAP**' created on the **MapViewer** console.

11. Add layers to the map and save it.


Edit Background Map - GOOGLE_MAP

Name	GOOGLE_MAP
Description	

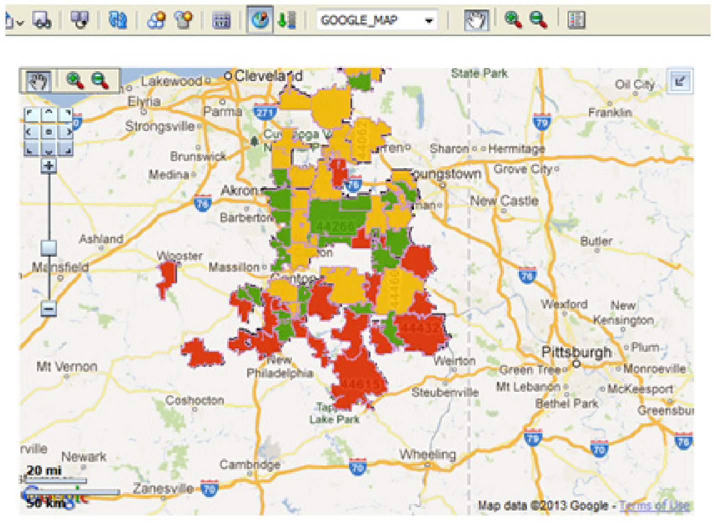
Interactive BI Layers

For each layer, select the zoom levels at which associated BI data can display

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
City		■	■	■	■	■	■	■	■	■	■	■							
City Name	■	■	■	■	■	■	■	■	■	■	■	■							
Postal Code	■	■	■	■	■	■	■	■	■	■	■	■							

12. Subject areas will now be associated to the map.

13. Create a new answer and add a new map-view to have the data displayed on the map.



Implementing the Maps

This section describes the method of implementing maps in Oracle Utilities Analytics (OUA): The implementation method is the default implementation method for Oracle Business Intelligence Enterprise Edition 11g. This form of map can be seen in various dashboards, such as Oracle Utilities Meter Data Analytics, Oracle Utilities Mobile Workforce Analytics, and Oracle Utilities Customer Care and Billing Analytics. Using this method, you can create new answers using the MapView. This view uses the configuration defined under the **Administration** menu in the **Manage Map Data**.

The layers, background maps, and the images being used in the map must be defined in this page. The key column and geographical columns are mapped for each subject area used in the analysis. This is a one-time setup unless new subject areas are added.

> **Note**: You should not customize the map metadata until you import the Spatial Catalog file.

For customizations that involve map analysis, all the modifications must be done in a separate folder in order for those modifications to be preserved when upgrading Oracle Utilities Analytics.

Viewing the About Page

The **About** page shows information about the product name along with the current release version and patch number. It also lists all of the languages that are currently installed in the product.

Configuring the Database

This section discusses how to configure the database for better system performance, including:

- **Partitioning**
  - **Partitioning of the Snapshot Facts**
  - **Partitioning of the Accumulation Facts**
- **Parallelism**
- **Optimizing the Top N Answers**
- **Archiving the Historical Data**

Partitioning

Partitioning helps to scale a data warehouse by dividing the database objects into smaller pieces, enabling access to smaller, more manageable objects. Having direct access to the smaller objects addresses the scalability requirements of the data warehouses.

It takes longer to scan a big table than it takes to scan a small table. Queries against partitioned tables may access one or more partitions that are small in contrast to the total size of the table. Similarly, queries may take advantage of partition elimination on indexes. It takes less time to read a smaller portion of an index from the disk than to read the entire index. The index structures that share the partitioning strategy with the table, such as local partitioned indexes, can be accessed and maintained on a partition-by-partition basis.

The partitioning of the facts is explained below in:

- **Partitioning of the Snapshot Facts**
- **Partitioning of the Accumulation Facts**

Partitioning of the Snapshot Facts

Most of the out of the box snapshot facts are delivered as Monthly snapshots. This means that the fact will have rows for each month from the beginning of the implementation.

All snapshot facts should be partitioned by the snapshot date. Listed below are the Snapshot facts and the column on which the partition should be created. When converting an existing fact into a partitioned fact ensure that the indexes, which do not have the partition key column are created as the local indexes. Ensure that the partitions are also created for the future periods so that the future snapshot loads do not fail while adding data.

For monthly snapshots primary partition based on the year and sub-partition based on the calendar month work well. For snapshots that are not monthly, different set can be used.

Fact	Column	Partition Type
CF_ARREARS	SNAPSHOT_DT	List
CF_PAY_PLAN_SNAP	SNAPSHOT_DT	List
CF_PA_SNAP	SNAPSHOT_DT	List
CF_CITY_OUTG	SNAPSHOT_DT	List

Fact	Column	Partition Type
CF_CTRL_ZONE_OUTG	SNAPSHOT_DT	List
CF_FEEDER_DLVRD_LOAD	SRC_DTTM	List
CF_OUTG	SNAPSHOT_DTTM	List
CF_OPR_DEVICE_SNAP	SNAPSHOT_DT	List

Partitioning of the Accumulation Facts

The accumulation facts can also be partitioned based on a date key. However, the data in the accumulation facts will be spread across multiple dates and a list based partition will not work for the accumulation facts. A range based partitioning scheme should be employed on the primary date_key column in the specific fact.

Given below is a query which gets the start and end range values from the CD_DATE dimension. Using this range values for individual partitions can be defined.

```
select abs_month_nbr  part_name_suffix
     , min(date_key) part_range_start
     , max(date_key) part_range_end
  from dwadm.cd_date
  group by abs_month_nbr
  order by 1;
```

Parallelism

The database can take advantage of the distinct data sets in separate partitions if you use parallel execution to speed up queries, Data Manipulation Language (DML), and Data Definition Language (DDL) statements. Individual parallel execution servers can work on their own data sets, identified by the partition boundaries.

The parallel execution enables the application of multiple CPU and I/O resources to the execution of a single database operation. It dramatically reduces response time for data-intensive operations on large databases typically associated with a Decision Support System (DSS) and data warehouses. You can also implement parallel execution on an Online Transaction Processing (OLTP) system for batch processing or schema maintenance operations, such as index creation. Parallel execution is also called ‘Parallelism’. The parallelism involves breaking down a task so that instead of one process doing all of the work in a query; many processes do part of the work at the same time. For example, when four processes combine to calculate the total sales for a year, each process handles one quarter of the year instead of a single process handling all four quarters by it. The improvement in performance can be quite significant. The parallel execution improves processing for:

- Queries requiring large table scans, joins, or partitioned index scans
- Creation of large indexes
- Creation of large tables (including materialized views)
- Bulk insertions, updates, merges, and deletions

Note: For details on parallelism, partitioning, and other performance enhancement options, refer to *Oracle Database VLDB and Partitioning Guide 11g Release 2*.

Note: Refer to the section “**Deploying Oracle Warehouse Builder Workflows”** in *Oracle Utilities Analytics Installation Guide* for setting the parallelism in materialized views and Oracle Warehouse Builder mapping.

Optimizing the Top N Answers

The Top N charts have to go through millions of records to find out the Top N objects that meet the criteria. Top N materialized views rearrange and partition the data so that the data reads are optimal. However, at times, depending on the amount of data, additional configuration may be required to reduce the size of the data set, which is being scanned in order to identify the Top N objects.

For example, a large sized utility with 6 million customers may have around 6 million service points in their Oracle Utilities Meter Data Management application. This means that the Consumption fact in the Oracle Utilities Meter Data Management star schema have several million records for every snapshot month. So, for each month level partition on the **Consumption Detail Level** materialized view, there will be several million records. The detail level pages (Top N Analysis and Unreported Usage Details) on the **Usage Summary** dashboard under Oracle Utilities Meter Data Analytics tries to access a huge volume of data, which can result in sluggish performance sometimes. The recommendation in such scenarios is to make a key prompt filter such as, **'City'** as mandatory. This ensures the report looks at a smaller data set; thereby, improving the report performance. You can choose to make any set of filters mandatory as per your data requirement.

Archiving the Historical Data

Assuming that the facts are partitioned as per the recommendations, it is now possible to archive historical data by selecting specific partitions and archiving them to a backup. Archival rules should be based on the implementer's data retention policies. This can be set up as a scheduled activity.

Chapter 5

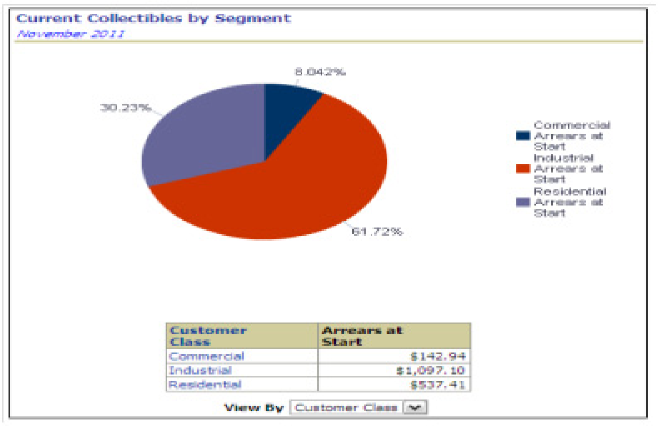
Extending Oracle Utilities Analytics

The data warehouse schema in Oracle Utilities Analytics (OUA) covers a wide range of reporting requirements. You might often require to add additional data elements to meet site specific requirements. Oracle Utilities Analytics allows such extensions to the schema through the use of user-defined constructs, such as User Defined Fields (UDFs), User Defined Measures (UDMs), User Defined Degenerate Dimensions (UDDGENs), User Defined Foreign Keys (UDDFKs) and User Defined Dimensions (UDDs). Using these constructs, you can extend the star schemas that are delivered along with the product. With the additional data now available in the warehouse, the custom reports can be created in Oracle Business Intelligence Enterprise Edition (OBIEE) to leverage this additional data. Using these features, you can easily customize the product to suite your extended requirements. The following sections are described in this chapter:

- **Sample Use Cases**
- **User Defined Columns**
- **Extending Star Schemas**
- **Customizing Analytics**

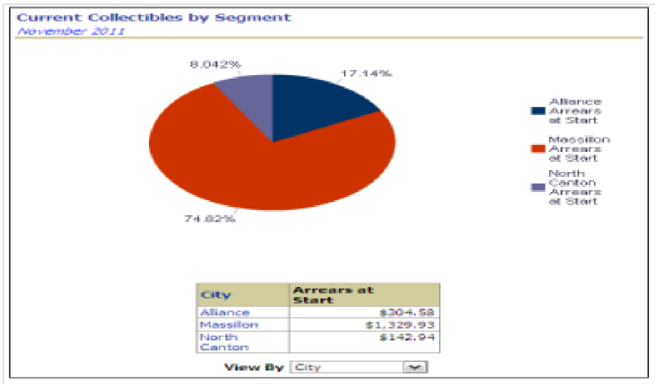
Sample Use Cases

You look at the facts in a data warehouse by slicing and filtering the analytic data by different dimensions. For example, the following graph shows Collectible sliced by the Customer Class field (the Collectibles fact is sliced by the Customer Class field on the Account dimension):


Current Collectibles by Segment
November 2011
8.042%
30.23%
61.72%
Commercial Arrears at Start
Industrial Arrears at Start
Residential Arrears at Start
View By
Customer Class

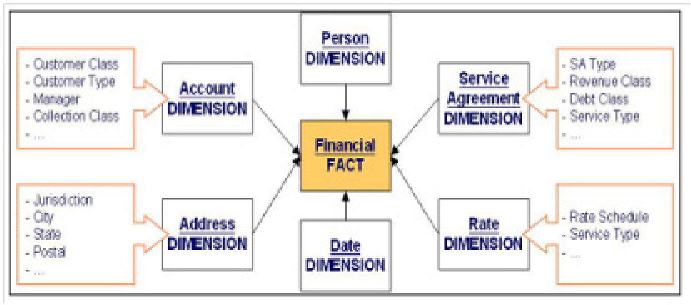
Customer Class	Arrears at Start
Commercial	$142.94
Industrial	$1,097.10
Residential	$537.41

Whereas the below report slices the same fact by a different field on a different dimension (the City on the Address dimension). In addition, it limits the analysis to a specific customer class, i.e., Commercial. The below diagram shows how a single report can be sliced and filtered by different dimensional attributes.


Current Collectibles by Segment
November 2011
8.042%
17.14%
74.82%
Alliance Arrears at Start
Massilon Arrears at Start
North Canton Arrears at Start
View By
City

City	Arrears at Start
Alliance	$304.58
Massilon	$1,329.93
North Canton	$142.94

You can slice and filter a fact using any field on any dimension linked to the analytic's fact. For example, you can slice reports related to the Financial fact by any field on its dimensions. The following simplified data model of the Financial fact's star schema helps clarify this concept.



This data model shows that the Financial fact has six dimensions. This means that graphs can be configured to allow you to slice the Financial fact by any of the attributes on the six dimensions. For example, you can set up a report to slice the Financial fact by any combination of:

- The Account Dimension's Customer Class and Manager
- The Address Dimension's Jurisdiction
- The Service Agreement Dimension's Revenue Class
- The Rate Schedule Dimension's Rate

You can set up another report to slice and filter this fact by a different combination of fields. It should be noted that the above example is a simplified version. In reality, the most facts have more than six dimensions and most dimensions have several fields.

While Oracle Utilities Analytics allows you to slice and filter a fact by any field on its dimension, it also enables you to limit the number of fields on your report to a discreet group. This helps the materialized views that are configured, may make the system slow if these views contain too many fields.

User Defined Columns

The predefined facts and dimensions are provided with a set of user extensible columns, which can be used for extending the existing entities. These columns include:

- **User Defined Field**
- **User Defined Measure**
- **User Defined Degenerate Dimension**
- **User Defined Foreign Key Dimension**
- **User Defined Dimension**

User Defined Field

The User Defined Fields (UDFs) reside on the dimension tables in the star schemas. In general, a dimension consists of minimum of ten UDF columns. You can make of use of these columns to extract additional information from the source system.

Once configured appropriately, the ETL provided for the star schemas automatically loads the data into the appropriate columns.

You can change the User Defined Field on your dimensions at a later date.

This UDF column feature allows all facts to be sliced and filtered by the newly added data in these user-defined fields. To slice and filter historical facts by the new fields, you must update the historical dimensions to contain the current value of the new user-defined fields.

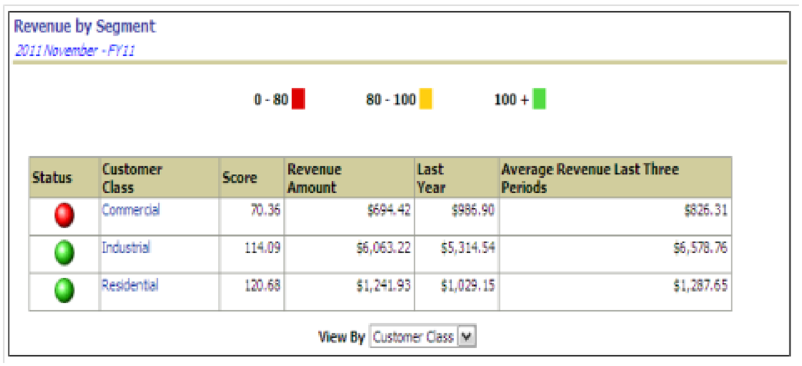
> **Note**: Refer the section **Customizing the Column Properties of Type 2 Slowly Changing Dimension** for adding the user-defined field in the report.

User Defined Measure

The User Defined Measure (UDM) column is used to reference the measures on the facts that you populate with implementation-specific measures.

A measure is a column on a fact that holds a measurable value. For example, the Financial fact has a measure that holds the amount of revenue produced by a bill. Most facts in the Oracle Utilities Analytics data warehouse have several measures. For example, in addition to the revenue measure, the Financial fact also has measures holding the total taxes and other charges on a bill.

For example, the following report shows several measures - score, revenue amount for the current, last, and the average revenue for the last three periods.


Revenue by Segment

2011 November - FY11

0 - 80 🟥   80 - 100 🟨   100 + 🟩

Status	Customer Class	Score	Revenue Amount	Last Year	Average Revenue Last Three Periods
🔴	Commercial	70.36	$694.42	$986.90	$826.31
🟢	Industrial	114.09	$6,063.22	$5,314.54	$6,578.76
🟢	Residential	120.68	$1,241.93	$1,029.15	$1,287.65

View By: Customer Class

The facts and their extract programs are delivered with all of the relevant measures populated. However, if your implementation requires additional measures, you can populate UDM column on the facts. To do this, you can introduce logic to the fact's extract program (in a user exit code) to populate one or more UDM column accordingly.

Note: No database, Oracle Warehouse Builder, or Oracle Data Integrator changes are necessary as both the data warehouse and Oracle Warehouse Builder / Oracle Data Integrator are delivered in a ready state to support the newly populated UDM columns.

User Defined Degenerate Dimension

The User Defined Degenerate Dimension (UDDGEN) columns reside directly on the fact table and can be used to filter the fact data in the same way as the User Defined Field (UDF). For example, the currency code columns are commonly used UDDGEN in Oracle Utilities Analytics application. These columns exist on most of the fact tables and can be used to limit the fact data shown in reports to a given currency code. Most fact tables in Oracle Utilities Analytics are delivered with multiple UDDGENs. These columns are populated by introducing user-exit code in the respective fact's extract programs. The main benefit of using UDDGENs in comparison to using UDDs is that UDDGENs can be populated in the fact's extract program and thereby, can reduce implementation time.

User Defined Foreign Key Dimension

At times, there are requirements that can be easily satisfied by adding an existing dimension to a fact. For example, the Case fact is not delivered referencing the service agreement dimension. If you require analytics that slice and filter cases by service agreement dimensional attributes, you can configure the system to reference the existing service agreement dimension on the Case fact. The facts that support this type of extension contain columns are called User Defined Foreign Keys (UDDFKs). If you do not see these columns on a fact, then it means that this functionality is not available in it.

User Defined Dimension

The User Defined Dimensions (UDDs) are empty dimensions that are delivered along with the star schemas in Oracle Utilities Analytics. You can make use of these dimensions to extend the existing star schemas.

As described earlier, you can set up the analytic reports to slice and filter a fact using any field on the dimensions linked to the fact. Oracle Utilities Analytics (OUA) delivers facts referencing the relevant dimensions. However, your implementation may need you to link additional dimensions to some facts. For example, the financial fact is delivered assuming that the revenue, tax, and expense amounts are aggregated regardless of the General Ledger (GL) accounts impacted by a financial transaction. If a given adjustment references six revenue GL accounts, all six revenue amounts are summarized onto a single Financial fact. This means that you cannot slice and filter revenue by specific General Ledger accounts.

If you have such a requirement to slice and filter an existing fact data using an additional group of related attributes, then you can opt to make use of these User Defined Dimension (UDD) tables. For each fact, two UDDs are provided. These tables can be utilized for extending the star schemas. Five unbound UDD foreign key columns are also provided.

You can use user-defined dimensions only if the dimension does not change the granularity of a fact.

Extending Star Schemas

You can extend star schema without actually updating the table structure. This can be achieved by using the user-defined columns. The specific details on how to extend the star schema for a particular source product is dependent on the ETL methodology. The below-mentioned sections cover both Oracle Warehouse Builder and Oracle Data Integrator based approaches:

- **Oracle Warehouse Builder Based ETL**
- **Oracle Data Integrator Based ELT**

Oracle Warehouse Builder Based ETL

For the source applications whose ETL is based on Oracle Warehouse Builder, the customization starts from the source application itself where the data extraction is done. The following source applications make use of Oracle Warehouse Builder based ETL.

- **Oracle Utilities Work and Asset Management**
- **Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management**

Since different source applications employ different approaches to extract their data, the sequence of steps done for schema extension varies on the extraction approach as well. Once the extractors have been configured to pull the additional data, the Oracle Warehouse Builder ETL delivered along with Oracle Utilities Analytics automatically loads these additional data into the warehouse.

The following sections cover the necessary steps to be done for each of the source applications.

Oracle Utilities Work and Asset Management

Oracle Utilities Work and Asset Management (WAM) uses the trigger-based approach to detect the changes on the base table that needs to be populated in the data warehouse.

Most of the extract programs support populating User Defined Field (UDF) and User Defined Measure (UDM) fields on their flat file records with specific fields from the source system. For example, you can set up the premise extract program to populate the first user-defined field on its flat file with the premise's city, county, or any address-oriented field.

You should specify to the extract program which fields require to be populated on the flat file by populating the batch process parameters. The number and type of parameters differ depending on the extract and type of information being extracted. However, in general, there are two types of fields that can be transferred to the flat file:

- **Columns**: Many dimensional extracts allow pre-defined columns to be populated on the flat file. For example, you can set up the premise extract program to populate the first User Defined Field on its flat file with the premise's city, county, or any address-oriented column. An analogous concept is used to populate the User Defined Measure on the fact extracts.
- **Characteristics**: Many dimensional extracts allow characteristics to be populated on the flat file. For example, you can set up the premise extract program to populate the first User Defined Field on its flat file with the premise's tax characteristic, or/and premise- oriented characteristic.

  Note: Most dimensional extracts support the population of their user-defined fields with characteristic values. A limited number of the fact extracts allow characteristics to be used to populate user-defined measures. This is because most of the transaction files that trigger the fact extracts do not contain characteristics.

  You can identify how an extract populates its user-defined fields and user-defined measures by populating parameters. Each user-defined field/ user-defined measure supported by an extract has two parameters that must be populated:

- **Type**: This parameter is defined or populated if the field is a true column or a characteristic. Enter PROG to populate a User Defined Field / User Defined Measure with a column. Enter CHAR to populate the User Defined Field / User Defined Measure with a characteristic.
- **Value**: This parameter defines the respective column or characteristic.
  To define a column, the value will be in the formation Table.Column (for example, CI_PREM.CITY_UPR - used on the address extract to populate a UDF with the upper-case format of an address's city).
  To define a characteristic, enter the characteristic's type code. Note, in the current release, only pre-defined value characteristics are supported.

Note: For details, refer to the relevant fact and dimension chapter in this guide for a description of each extract program and the various User Defined Field (UDF) and User Defined Measure parameters that are supported in each.

Extracting Additional Fields

While the extract programs are delivered to populate their flat files with commonly used information, to populate the User Defined Field and User Defined Measure with information not supported by the base-package.

Note: Refer to the **Appendix D** for complete details regarding batch control and trigger names respectively while trying to customize the extracts.

Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management

Oracle Utilities Meter Data Management (MDM) and Oracle Utilities Mobile Workforce Management (MWM) generally use the Sync Business Object (BO) based approach to detect the change in the source table that needs to be populated in the data warehouse. In some cases, where complex logics perform the extract, idiosyncratic batch processes perform the job.

Both of these approaches support the population of User Defined Field (UDF) and User Defined Measure on their flat file records with specific fields from the source system. For example, you can set up the metadata in the source system to populate the first user-defined field on a service point dimension's flat file with the time zone, or any service point-oriented field.

A particular fact or dimension may fall into one of these two extraction styles. Both of them have their own ways of being extended. In both the methods, once the metadata has been configured appropriately, the respective batch controls run again to generate the new flat files with the User Defined Field /User Defined Measure columns populated.

Each fact and dimension use two types of approaches for extract:

- **Synchronization BO Based Extract**
- **Idiosyncratic Batch Extract**

Synchronization BO Based Extract

Each fact and dimension that uses the synchronization Business Object (BO) approach for extract has its own Sync Request business object. Each of these business objects has the following options:

- **Snapshot DA**: This is the data area, which contains the elements sent to the data warehouse. The elements in this Data Area (DA) are listed in the order in which the ETL expects them.
- **BO to Read**: This is the primary Business Object (BO). The contents of this business object are sent to the data warehouse. Its elements should match those in the Snapshot DA as it is read, and then its contents “moved by name” to the Snapshot DA.

- **Element Population Rule**: These options are used to populate an element on the Snapshot DA that cannot be moved by name from the BO to read. The option value contains the following mnemonic:

Option Type	Sequence	Option Value
Element Population Rule	<A unique sequence number>	sourceElement=XXX populateTargetWithInfo String=true/false targetElement=XXX

The **sourceElement** attribute refers to an element in the Business Object (BO) specified in the option type 'BO to Read'. The **targetElement** attribute refers to an element in the Snapshot DA that needs to be populated. The **populateTargetWithInfoString** accepts a 'true' or 'false' value (a value of 'true' means that the sourceElement's foreign key reference in the BO to read schema is used to retrieve the element's information string).

- **Post Service Script**: This is the script, which is executed after the Snapshot DA is populated with the BO to read elements and the element population rules (if any). It is responsible for populating elements on the Snapshot DA that cannot be derived directly from the business object to read.
  You can extend the facts and dimensions by adding new options:
- **Element Population Rule**: To extend the element population rules, you should add new element population rule options following the mnemonic specified above. You can set up as many element population rules as you need, and the sequences do not matter.
- **Post Service Script**: To extend the post service script, you should create a new service script.

Option Type	Sequence	Option Value
Post Service Script for Extract	<A unique sequence number>	Custom Service Script Name

The option value to be supplied is the name of the customer service script, which has the logic to populate the User Defined Field/User Defined Measure columns with the desired values. There are few things to note here. First, the schema of the service script should match the schema of the Snapshot DA used in the Sync Request BO (this can be done by including the Snapshot DA specified on the Sync Request BO's option in the post service script's schema). Second, the post service script with the highest sequence number is executed. Hence, make sure to provide the appropriate sequence number on your custom service script. And lastly, because the highest-sequenced post service script overrides any existing scripts, if Oracle Utilities Meter Data Management, or Oracle Utilities Mobile Workforce Management has been delivered with a post service script, your new custom service script should either execute the existing post service script (and avoid duplicating the same steps in your custom script), or not, depending on what you want to populate.

Idiosyncratic Batch Extract

For snapshot facts and certain dimensions, there are idiosyncratic batch jobs that are created to handle complex extraction logic.

- **Batch Parameters**: Certain batch jobs delivered as a part of the source system allow additional batch parameters for the end users to supply custom service script. The service script must include the same Snapshot DA defined as batch parameter on the batch job.
- **Algorithm Soft Parameters**: Certain algorithms delivered as part of the source system allow additional soft parameters for the end users to supply custom element population rules and service script. The usage of these parameter values is similar as explained for the 'Element Population Rule' and 'Post Service Script' above in the Synchronization BO-based style. The

service script must include the same Snapshot DA defined as the soft parameter on the algorithm.

- **New Algorithm**: In case you need to drastically change the logic of populating the UDF/ UDM columns or even the remaining set of fields in the flat files, you may wish to create a new algorithm and plug it in on the originating Business Object (BO). This algorithm is plugged in with a higher sequence than the base product algorithm. Additional care needs to taken since this completely overrides the existing algorithm logic and this new algorithm has the logic of populating the entire flat file.

Extracting Additional Fields

While the extract programs are delivered to populate their flat files with commonly used information, you may want to populate the User Defined Field and User Defined Measure with information not supported by the base-package. To do this, you must use the “Sync BO Options” to the respective extract objects.

Note: Refer to the **Appendix E** and the **Appendix F** for the complete details regarding **Batch Control** and **Sync BO Names** respectively while trying to customize the extracts.

Note: For additional details, refer to the knowledge article “Extending the OUBI Data Warehouse (Using Sync BO in Oracle Utilities Meter Data Management)” (Doc ID 1516859.1) available at the My Oracle Support website.

Oracle Data Integrator Based ELT

Oracle Utilities Analytics (OUA) provides placeholder columns for the user extension to the out of the box star schemas. You can add logic to populate these user-defined columns. Refer to the section **Extending Star Schemas** in this chapter for details, such as which columns can be customized and what is the purpose of these columns.

The ELT for the following source applications are based on Oracle Data Integrator, and can be extended based on the steps described in this section.

- Oracle Utilities Customer Care and Billing (CC&B)
- Oracle Utilities Operational Device Management (ODM)
- Oracle Utilities Network Management System (NMS)

To start populating the additional user-defined columns, you should identify the source table, which is used to populate the target column. If the table is not in the list, follow the steps in the section below “**Extending the Replication**” to add this table for replication. The next step is to write a user extension procedure that can populate these columns. The interfaces have been created so that these perform additional actions, such as translating the source foreign keys to appropriate dimension keys. This section covers:

- **Extending the Replication**
- **Upgrading/Customizing the Edge Application**
- **Populating the User-Defined Columns**
- **Creating a Custom Dimension**
- **Creating a Custom Fact**
- **Utilizing a Predefined UDD dimension**
- **Utilizing an Unused User Dimension FK**
- **Configuring the Storage**
- **Customizing the Column Properties of Type 2 Slowly Changing Dimension**

Extending the Replication

Out of the box Oracle Utilities Analytics ships a required set of tables marked for the replication. For extensibility purposes, additional tables may be configured to be replicated.

Configuring the Additional Tables for Replication

To configure additional tables for replication, perform the following steps:

1. Check if the table to be replicated is listed in the **B1_SOURCE_TABLE.**
2. If the table is not listed in the table, add an entry in the **B1_OBJECT_MAP** setting the **SOURCE_OBJECT_NAME** as the table name and the **TARGET_OBJECT_NAME** as the target fact or dimension, which has some attributes loaded from this table.
3. The **SOURCE_OBJECT_NAME** can also be set to a Maintenance Object (MO) or Business Object (BO) in which case all the tables comprising the Maintenance Object (MO) is pulled in.
4. Execute the scenario **B1_OUAF_CFG_METADATA**. Now, an entry for this table will be present in the **B1_SOURCE_TABLE**.
5. Set the **CM_REPLICATE_FLAG** to 'Y'.
6. Execute the **initateSetup.cmd**, which creates a new model and the steps required for the replication process to be initiated. Do not deploy the generated Oracle GoldenGate scripts.

   For details, refer to *Oracle Utilities Analytics Installation Guide* and check the section **Initiate Setup** under the chapter **Installation**.
7. Execute the scenario B1_REPL_COLUMNS. This scenario is available in the Scenario Folder Accelerators. This builds some additional configuration for the replication.
8. Execute the **initiateSetup.cmd**, which creates a new model and the steps required for the replication process to be initiated.

   For details, refer to *Oracle Utilities Analytics Installation Guide* and check the section **Executing the Knowledge Modules (KMs)** under the chapter **Oracle Utilities Analytics Initial Installation**.
9. Follow the instructions to set up the Oracle GoldenGate scripts generated to start.

   The need for executing the **initateSetup.cmd** twice is a limitation that will be addressed in the next release.

Upgrading/Customizing the Edge Application

Any product can undergo data structure changes as a result of upgrades and/or customizations. There may be data patches that are applied on the source application. Upgrades to the source system should not impact the product negatively. Here are the some possible edge application upgrade/customization scenarios:

- Table structure changes
- Table/Column is renamed or dropped
- New column(s) added
- New table(s) added
- Column precision/size change
- Column data type changes

Category	Change	Expected Behavior
Structure change	Table/Column is renamed or dropped	The DDL replication propagates the DDL change on the source to the target. The table rename is not supported in the DDL replication. However, the column rename is supported in the DDL replication with some expected assumption described earlier. The table dropped is also not supported, but dropping of the column is handled by the DDL replication. In case, the table is renamed or a column is renamed, then the views referring to the original tables and the columns are invalidated and target fact dimension loads will fail. This requires manual steps to resolve table/column rename.
Structure change	Column(s) Added	The DDL replication adds a new column before the JRN columns in the replication table and no components are impacted. Oracle GoldenGate processes continue to run and the new column stores the updated or inserted values at the source. Below are some restrictions: • If a column is added to a HIST type table and if the update does not include a key column filter in the where clause, then any subsequent updates on the added column are not captured. • The XMLTYPE columns are not supported as the DBMS_REDEFINITION does not support online redefinition of the tables with the binary XML columns. • Every DDL that is applied on the replication layer via Oracle GoldenGate will cause the replicat process to stop. This is required to prevent any data loss during the time that the replicated table is redefined. After every DDL, the replicat needs to be started again.
Structure change	Table(s) Added	No impact on BI.
Structure change	Column data type changed	The corresponding change will be applied on the replication layer and the new data will be processed without interruptions.

Category	Change	Expected Behavior
Content change	CLOB xml structure change	Assuming that previously a node is used: <isResidentialCustomer>true</ isResidentialCustomer> and this is changed to <customerType>residential</ customerType> • None of the interfaces fails, because of this change. • Post this change, the logic to identify customer type as residential now returns null and all the customers are now tagged as non-residential. • The data prior to this change have the old xml structure and the new data have the new structure. This may require a change on the BI interfaces and / or data cleansing in the replication layer.
Content change	New/Modified characteristic types	No impact on BI. The new characteristic types may need to be configured in the job parameters.
Content change	Data correction patch	Post an upgrade where the table structures have been changed, it is possible that a data correction patch is provided to convert the older version data to the newer version. • For the tables where the history is maintained in the replication layer, this data correction may automatically propagate to the replication layer, depending on the type of structural change. • For the tables where the history is maintained, this causes a new row to be created on the target for each of the impacted tables. The older data will still have the old data and will require a cleanup on the replication layer to fix the data appropriately. • This potentially creates a spike in the data volume to be reprocessed for the dimensions and the facts. • Depending on the functionality impact of the data correction, facts/ dimensions may need to be reprocessed completely.
Version	Multiple instances on different versions	The multiple edge app instances on the different versions are not supported as the replication schema structures are shared and the two different versions can possibly have the different structures.

During setup and configuration, the Oracle Utilities Analytics application is aware of only two users (GoldenGate user and the schema owner). Any DDL performed by a different user will not be processed. This is a restriction imposed by Oracle GoldenGate where every user needs to be explicitly remapped to a user on the target.

Populating the User-Defined Columns

Extending the data load process for populating the user-defined columns (UDF, UDM, UDDGEN and UDDFK) consists of the following steps.

1. Create a user exit procedure for the target entity in replication schema. The recommended naming convention is **CM_UDX_<entity_name>**.
2. The **User Exit** procedure signature supports three parameters. This is to ensure that the **User Exit** procedure works on a specific dataset only. The **User Exit** procedure updates the user-defined columns in the UDX table.
    - **V_JOB_NBR**: This is the job number for processing. All the SQL statements processing on the UDX table have a filter on the job number.
    - **V_SLICE_BEG_TS**: This is the slice begin timestamp. It is used for filtering data for the particular slice or snapshot. Data processed is greater than, or equal to this value.
    - **V_SLICE_END_TS**: This is the slice end timestamp. It is used for filtering data for the particular slice or snapshot. The data processed is less than this value.

    Note: Refer to the section **Configuring the Jobs** in the **Appendix B** to set the user exit procedure name.

For Example:

Below is a template code for extending an entity.

```
create or replace
procedure <%=odiRef.getSchemaName()%>.cm_cd_acct_udx
          (v_job_nbr      in number
          ,v_slice_beg_ts in date
          ,v_slice_end_ts in date)
as
begin

  update udx_cd_acct  udx
     set (udf10_cd
         ,udf10_descr)
       = (select char_val
               , 'srch-val:'||srch_char_val
            from ci_acct_char
           where char_type_cd = 'CI_VATCA'
             and acct_id      = udx.src_acct_id)
   where job_nbr = v_job_nbr;
end;
```

When writing the code for extending any target entity:

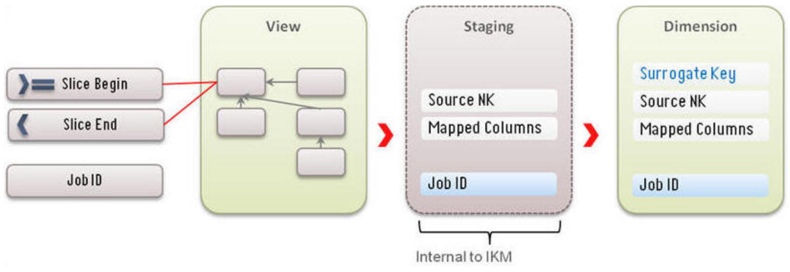
- **<entity_name>**: Replace with the appropriate target entity name. For example, CF_OPR_DEVICE.
- **<TABLE_A>**: Replace with the source table or tables that are required for fetching the source data.
- **<some_column>**: The source column to be used for populating the user-defined column.
- **<nk1>**: In the case of **CF_OPR_DEVICE**, the natural keys are **SRC_ASSET_ID** and **DATA_SOURCE_IND**.

- **<trx_date>**: Typically, this is the **update_dttm** column of the source table. However, depending on the functionality, this can be a transactional date column.

  Note: The above- mentioned example on how you can select a value from one of the replicated columns. You can replace this logic by substituting code of any complexity as long as it returns the parameters mentioned above.

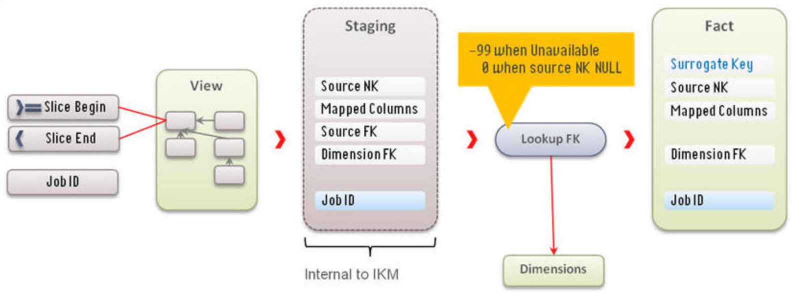
Creating a Custom Dimension

The following pattern is followed for creating a dimension:



Creating a Custom Fact

The following pattern is followed for creating a fact:



A view interface is created to encapsulate the replication schema from the interface that loads the data into the target. This interface uses tables from the replication schema and the target is a view that is generated. This uses the Integration Knowledge Module View Generator. After the interface is generated, execute in the appropriate contexts to create the view. This view should contain the unique key combination, the JRN_FLG columns from the primary driving table and the JRN_SLICING_TS column from the driving table.

- Reverse engineer the view into Oracle Data Integrator model and utilize the view as the source.
- Create a interface with the View defined in step above as source and include a filter on the JRN_SLICING_TS as shown in the diagram above.
- Create a package and include the variables GLOBAL.B1_SLICE_BEG_TS, GLOBAL.B1_SLICE_END_TS, GLOBAL.JOB_ID. These should be defined as set variables in the package. These should be before the interface in the package flow.

For facts, the FK joins to the dimensions should be outer joins and should set the dimension FK as 0 when the source natural key for lookup is null and should set the FK as -99 when the source natural key is not null, but a corresponding dimension key is not found.

Configure the entity in “Target Entities” page and create job configuration for the entity in “Job Configuration”.

You can start by looking at an out of the box dimension/fact package and interfaces.

Utilizing a Predefined UDD dimension

This is equivalent to creating a custom dimension. Follow the steps mentioned above to populate the predefined UDD dimension.

Utilizing an Unused User Dimension FK

Most facts have placeholders for 5 user-defined dimensions. These are the UDD1_FK to the UDD5_FK columns.

First, design the dimension and create the appropriate interface as explained above. For each of these unknown dimensions, a default view is provided. This view is structurally similar to an Type 2 Slowly Changing Dimension (SCD2). Override this view by replacing the selected part with columns from the custom dimension.

Configuring the Storage

In this release, an additional storage configuration is provided that enables implementers to manage the tablespaces utilized for the database objects that are created during the execution of the Oracle Data Integrator jobs.

The following attributes are available:

Attribute	Purpose
Context Code	Identifies the context for which this is applicable. A value of “GLOBAL” will be allocable across all the contexts unless overridden by an entry for the specific context.
Logical Schema	The logical Schema for which the configuration is applicable. A empty value indicates that this is applicable to all logical schemas.
Object Type	The type of object for which the configuration is applicable. This can be “Table, Index, Partition’.
Sequence	The default is 0. This is applicable only for partitions. This can be used to map multiple table spaces in a round-robin scheduling while creating the new partitions within the same table.
Table Space Name	The name of the table space to be associated with the object.

Customizing the Column Properties of Type 2 Slowly Changing Dimension

A type 2 slowly changing dimension has the following column types:

- Natural Key
- SCD Start
- SCD End
- SCD Flag

Any change in any of the remaining columns is treated as a new history item for the same natural key. In the next release of Oracle Utilities Analytics, you will be able to decide which of the columns should not be considered for creating a new history item.

Using Oracle Data Integrator, this is done by changing the corresponding property of the column in the model of the target dimension. However, this change requires regeneration of the dimension load scenario. The customizations done by you will also get overridden by any future upgrades provided by Oracle.

Implementers can control the columns where a change will cause a history to be created. You can change the properties by accessing Administration user interface. Once the configurations have been changed, the following script needs to be executed to ensure that the changes have been applied to the appropriate scenarios:

For UNIX:

Perform the following steps:

1. Navigate to <Install_Dir>/bin directory.
2. Initialize the environment with the ./splenviron.sh -e <envname> command.
3. cd $SPLEBASE/bin.
4. Run ksh ./updateODIMetadata.sh

For Windows:

Perform the following steps:

1. Navigate to <Install_Dir>/bin directory.
2. Initialize the environment with the ./splenviron.cmd -e <envname> command.
3. cd %SPLEBASE%\bin.
4. Run updateODIMetadata.cmd

Customizing Analytics

This section describes how to use Oracle Business Intelligence Enterprise Edition (OBIEE) to customize Oracle Utilities Analytics. It includes the following topics:

- **Modifying the RPD file**
- **Customizing the Answers**
- **Customizing the Report Labels**

Modifying the RPD file

All the customer modifications must be done in a separate repository file, which is separate from the product's out-of-the-box repository. Any customization done is merged into the upgraded repository file through the Merge utility of Oracle Business Intelligence Enterprise Edition (OBIEE) along with the product's out-of-the-box repository file.

Oracle recommends that you use a staging environment for the repository upgrade. However, as long as the customer modifications are done on the top of a copy of the base repository file, the Oracle Business Intelligence Enterprise Edition upgrade process is able to handle most customizations that may be made to the repository file. The simpler the changes, the less complex upgrade procedure; hence, you should try to limit the changes made to the repository file.

Note: For more information about managing, upgrading and merging repository (.rpd) files, refer to *Oracle Business Intelligence Server Administration Guide*.

Customizing the Answers

All user modifications are done in a separate folder in order to preserve these modifications for upgrading Oracle Utilities Analytics. If an existing answer needs to be changed to meet your requirements, a copy of the product report is created, and changes are made to the copy (not to the original report). The dashboard is changed to point or refer to the new copy instead.

Note: The dashboards are overwritten during the upgrade. Any mappings between dashboards and customized answers are lost and must be re-mapped manually. Therefore, you should use a staging environment for upgrade and manually remap dashboards before moving the upgraded customized content into the production environment.

Note: For more details about managing, upgrading, and merging presentation catalogs, refer to *Oracle Business Intelligence Presentation Services Administration Guide*.

Note: For more details on how to create or edit answers, refer to *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

Customizing the Report Labels

You can choose to opt for a different label or caption for an existing report or any of the report columns in it. You can provide an override description instead of the product provided description. These details can be provided in the **Base Field Maintenance Page** under the **Administration Dashboard** in the **Oracle Business Intelligence Enterprise Edition Dashboards** menu. Once the changes are saved and the cache is clear, upon the next login, the new descriptions can be seen on the report title or the column title.

Note: For more details, refer to the '**Maintaining the Administration Dashboards**' section in the **Chapter 4**: **Configuring Oracle Utilities Analytics**.

Chapter 6

Adding New Components

This chapter focuses on how you can create new components, such as analytics, star schemas, and ETL/ELT processes to load them as per their additional requirements. Oracle does not support changes to the base products (other than the ones explained below), additional star schema, or related functionality. The following topics are covered in this chapter:

- **Creating a New Star Schema for Oracle Warehouse Builder Based ETL**
- **Defining Oracle Warehouse Builder Based ETL for a New Star Schema**
- **Creating a New Oracle Warehouse Builder Workflow**
- **Creating a New Object for Oracle Data Integrator Based ELT**
- **Creating a New Analytics**

Note: This chapter serves as a developer's guide on how to create custom components on the top of the base product. Oracle does not support these components, or any issues that might arise. Thereof, you need to develop and maintain these components independently.

Creating a New Star Schema for Oracle Warehouse Builder Based ETL

The star schema is perhaps the simplest data warehouse schema. It is called a star schema as the entity-relationship diagram of this schema resembles a star with points radiating from a central table. The center of the star consists of a large fact table. The end points of the star are the dimension tables.

A star query is a join between a fact table and a number of dimension tables. Each dimension table is joined to the fact table using a primary key to foreign key join. However, the dimension tables are not joined to each other. The optimizer recognizes star queries and generates efficient execution plans. It is not mandatory to have any foreign keys on the fact table for star transformation to take effect.

A typical fact table contains keys and measures. A star join is a primary key to foreign key join of the dimension tables to a fact table. The main advantages of a star schema are as follows:

- Provides a direct and intuitive mapping between the business entities analyzed by the end users and the schema design.
- Provides highly-optimized performance for the typical star queries.
- It is widely supported by a large number of business intelligence tools, which may anticipate or even require that the data warehouse schema contain dimension tables.

The star schemas are used for both simple data marts as well as very large data warehouses. Once the model has been designed, the Oracle Warehouse Builder Code Generator can be utilized to generate the mappings and process flows.

Note: For the details regarding data modeling, refer to the **Chapter 19: Schema Modeling Techniques** of the guide *Oracle® Database Data Warehousing Guide 11g Release 2*.

Defining Oracle Warehouse Builder Based ETL for a New Star Schema

Since different source applications employ different approaches to extract their data, the sequences of steps are done for schema extension varies on the extraction approach as well. Once the extractors have been configured to pull the additional data, the Oracle Warehouse Builder ETL delivered along with Oracle Utilities Analytics automatically takes care of loading these additional data into the warehouse. The following sections cover the necessary steps to be done for each of the source applications.

- **Oracle Utilities Work and Asset Management**
- **Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management**

Oracle Utilities Work and Asset Management

Oracle Utilities Work and Asset Management (WAM) uses the trigger-based approach to detect the change in the source table that needs to be populated in the data warehouse.

To populate the new facts/dimensions, perform the following steps:

1. Create the new triggers that have the business logic to extract the required data from the new tables.
2. Create a new extract program/procedure to with the desired extraction logic from the change log tables.
3. Create the new batch controls for the extract programs.
4. Run the new batch controls to extract data into the flat files.

Note: Refer to the **Appendix D** for the existing Batch Control and Trigger Names while trying to create the new extracts.

Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management

The source applications of Oracle Utilities Meter Data Management (MDM) and Oracle Utilities Mobile Workforce Management (MWM) make use of the Sync Business Object (BO) based approach for data extraction. For all the facts and dimensions other than those of snapshot type and certain dimensions with complex extract logic, it is advisable to use the Synchronization BO mechanism provided by Oracle Utilities Application Framework (OUAF). There are two options to extract a new dimension or fact data.

This section covers:

- **Creating a New Synchronization Request Business Object**
- **Creating an Idiosyncratic Java Batch Program to Extract Data**

Creating a New Synchronization Request Business Object

Perform the following steps to create the new synchronization request Business Object (BO):

1. Copy an existing Sync Request BO.
2. Modify the following BO options in the newly created Sync Request BO:
    - **BO to Read**: This needs to be populated with the BO, which has elements that need to be extracted.
    - **Snapshot Data Area**: This defines the schema, which is exactly extracted in the flat file, including the order of elements and the element types.
    - **Post Service Script**: To extract data, which is not available in the schema BO to Read BO, write a processing script to extract such elements.
    - **Element Population Rule**: To move data from an element defined on BO to read BO to another element defined in the Snapshot Data Area (DA), add the element population rule.
    - **Batch for Extract**: This is the name of the batch that needs to be executed to extract the flat files. It is recommended that you create a new extract batch control for your new fact or dimension, so you can define and default the parameters specific to your fact or dimension, but you should be able to use the existing extract java batch program.
    - **Star Schema Type**: Mention whether this Business Object (BO) is for a fact or dimension.
3. To handle initial synchronization, it is recommended that you create a new initial load batch control for your new fact or dimension if there is no existing one for the Maintenance Object (MO) of your ETL source. If there is an existing initial load batch control for the MO, add the newly created Sync Request BO as an additional sync request BO batch parameter.
4. To handle incremental sync, create an audit algorithm to define the logic to control the creation of pending sync request record. This should be plugged in on the MO of your ETL source. Refer to an existing audit algorithm delivered in the edge application. Depending on the audit algorithm created, add the newly created sync request BO as an option on the MO of your ETL source.

For snapshot type facts and certain dimensions with complex extraction logic, you can create an idiosyncratic java batch program to extract the data without using the sync request BO.

Creating an Idiosyncratic Java Batch Program to Extract Data

Perform the following steps to create such program:

1. Define the Data Area (DA) that reflects the structure of the flat file, including the data type and the order of elements.
2. Write the batch extract program that retrieves each record that needs to be extracted and performs the following:
    - Populate the data area with the information to be extracted as appropriate.
    - Invoke the BusinessService “F1-ConvertXMLToFileFormat” to transform Data Area into the fixed length string. This string is written to the extract file when the program is executed.
    - Write the record to the extract the flat file.
3. Create a new batch control for the new extract program.

    Note: Refer to the **Appendix E** and the **Appendix F** for the details regarding the existing Batch Control and Sync Business Object (BO) Names respectively while trying to create the new extracts.

    Note: For additional details, refer to the knowledge article “Extending the OUBI Data Warehouse (Using Sync BO in Oracle Utilities Meter Data

Management) (Doc ID 1516859.1)" available at the My Oracle Support website (https://support.oracle.com/).

Creating a New Oracle Warehouse Builder Workflow

The ETL for the following source applications are based on Oracle Warehouse Builder and new load processes can be added based on the steps described in this section:

- Oracle Utilities Work and Asset Management
- Oracle Utilities Meter Data Management and Oracle Utilities Mobile Workforce Management

Note: Before creating Oracle Warehouse Builder workflows, it is recommended that you should have knowledge of Oracle Warehouse Builder, Oracle Business Intelligence Enterprise Edition and Data Warehouse concepts. It is also recommended that you should have experience in developing Oracle Warehouse Builder components.

This section covers the following:

- **Creating the Oracle Warehouse Builder (OWB) ETL Code**
- **Refreshing the Cache Data**

Creating the Oracle Warehouse Builder (OWB) ETL Code

The steps below outline the high level steps to be carried out for creating the Oracle Warehouse Builder (OWB) ETL code:

1. Create a fact or dimension.
2. Specify the external table name.
3. Map a column from external table to target.
4. Specify the join conditions.
5. Generate the Oracle Warehouse Builder code.

Note: For specific details, refer to the **Appendix J**: **Oracle Warehouse Builder Deployment**.

Refreshing the Cache Data

Oracle Business Intelligence Enterprise Edition (OBIEE) provides a mechanism called Event Polling, which allows Oracle Business Intelligence Enterprise Edition to query a database table to find out when data has been updated in fact or dimension tables. By modifying the Oracle Warehouse Builder load process to populate an Event Polling table, you can let Oracle Business Intelligence Enterprise Edition know when data has been updated in the data warehouse, and enable Oracle Business Intelligence Enterprise Edition to know when to refresh the cache data that has been queried before.

A new event polling table B1_OBIEE_EVENT_POLLING is available as a part of Oracle Utilities Analytics.

The use of an Oracle Business Intelligence Enterprise Edition Server Event Polling table (event table) is a way to notify Oracle Business Intelligence Enterprise Edition server that one or more physical tables have been updated, and then that the query cache entries are stale.

Each row that is added to an event table describes a single update event, such as an update occurring to a product table.

The Oracle Business Intelligence Enterprise Edition server cache system reads rows from, or polls, the event table, extracts the physical table information from the rows, and purges stale cache entries that reference those physical tables.

For new requirements, the new extractors are created along with new Oracle Warehouse Builder load processes to load data into the new fact or dimension tables. Here in order to ensure that the Oracle Business Intelligence Enterprise Edition cache data is automatically refreshed, an additional step has to be included in the Oracle Warehouse Builder process flow to ensure that an entry is made in the available event polling table B1_OBIEE_EVENT_POLLING.

This ensures that whenever the new data is loaded by the Oracle Warehouse Builder process flow, the Oracle Business Intelligence Enterprise Edition cache is automatically refreshed and made available for the analytics reports.

Refer to an existing base product supplied Oracle Warehouse Builder process flow for samples.

Creating a New Object for Oracle Data Integrator Based ELT

The ELT for the following source applications are based on Oracle Data Integrator and new requirements can be added based on the steps described in this section.

- Oracle Utilities Customer Care and Billing (CC&B)
- Oracle Utilities Operational Device Management (ODM)
- Oracle Utilities Network Management System (NMS)

> **Note**: Before creating new objects, it is recommended to be proficient with Oracle Data Integrator, Oracle Business Intelligence Enterprise Edition and Data Warehouse concepts. It is also recommended that you should have experience in developing Oracle Data Integrator components.

You can create the following:

1. All the custom interfaces, packages, or procedures in a separate folder under the main project folder.
2. All the custom models under a separate folder in the models section.

The following sections list out the activities that need to be performed to add new facts and dimensions using Oracle Data Integrator:

- **Configuring the Entities**
- **Setting Up the Replication**
- **Creating a New Model for the Facts**
- **Creating a New Model for the Dimensions**
- **Creating a New Model for the Replicated Objects**
- **Creating a New Interface**
- **Creating a New Package**
- **Creating a New Interface for the Materialized View**

Configuring the Entities

Once you have designed the new facts and dimensions, perform the following steps to start the development.

1. Create the table structures in the DWADM schema. The recommended practice is to prefix customer owned objects with CM_.

2. Use Oracle Utilities Analytics Administration to create entries in target entity and job configuration for the new objects that you have created.

   Note: Refer to the **Appendix B** for further details.

Setting Up the Replication

While developing the requirements for a new star schema objects, identify the source entities that is required to populate the target entities. Using the **Admin** user interface, check if these entities are available in the table **B1_SOURCE_TABLES.** If yes, follow the steps mentioned in the **Extending Replication** section to enable replication for additional tables.

Assuming that the source system maintenance objects or specific tables that are sources for your new interfaces are not available in the table **B1_SOURCE_TABLES**, follow these steps:

1. In the current release, the metadata table **B1_OBJECT_MAP** is not exposed via the Admin user interface. You should not modify any entries already present in this table. For a new MO or the table source, create a new entry mapping your source MO/table to the target entity.
2. Once configured, execute the package **B1_OUAF_CFG_METATADA**. The **B1_OUAF_CFG_METATADA** package is an implementation accelerator for Oracle Utilities Application Framework products, and it pulls additional information for these objects into the metadata tables. This package is automatically executed during the addition of a new instance as a source for Oracle Utilities Analytics.

   Note: Refer to the **Appendix B** for a list of the metadata tables and the purpose of each of these metadata.
3. Follow the steps for extending replication to enable these new entities to be replicated.

Creating a New Model for the Facts

To create a new model for new facts, perform the following steps:

1. Create a new model for the facts.
2. Set the logical connection as “Target”.
3. Reverse engineer your newly created facts into this model.

Creating a New Model for the Dimensions

To create a new model for new dimensions, perform the following steps:

1. Create a new model for the dimensions.
2. Set the logical connection as “Target”.
3. Select the custom Reverse-engineering Knowledge Modules (RKM) for reverse engineering the dimensions. You can set the prefix used for the effective date columns in the type 2 dimensions.

The custom RKM auto configures type 1 and 2 dimension configurations based on the unique and primary keys on the database tables. The dimension should have a primary key constraint on the surrogate key column and a unique key on the natural key.

Creating a New Model for the Replicated Objects

Once the required objects have been replicated, follow the instructions below to set up a source model for the replicated tables.

1. Create a new model, set the logical schema to “Replication”.
2. Reverse engineer all objects those you need as source for the target entities.

Creating a New Interface

To create a new interface for your target entities, follow the steps below:

1. Under your custom folder, right-click and select the new interface.
2. Drag and drop your target table to the target section of the interface.
3. Drag and drop your source tables (from the replication model) into the source section of the interface.
4. Set up the appropriate join conditions between the tables.
5. Map the columns from the source to the target as per your design.
6. Add new filter.
    - Add the condition on slicing_ts column using the global variables, as below:

```
          (src.slicing_ts >=
to_date('#B1_SLICE_BEG_TS','yyyymmddhh24miss')
        ( src.slicing_ts <
to_date('#B1_SLICE_END_TS','yyyymmddhh24miss'))
```

    - For multiple tables add an “OR” condition to ensure that all changed records are picked up even if the change occurred in only one table.
7. Mark the columns other than the Primary Key and Unique Key columns for update and all columns for insert.
8. In the **Flow** tab, select the appropriate Knowledge Module (KM) based on the target entity. The Knowledge Modules supplied with the product should be used and they have been named so that it is easy to identify the fact/dimension knowledge modules to be used.
9. Save the interface.

The following expression should be used for mapping the target dimension key columns:

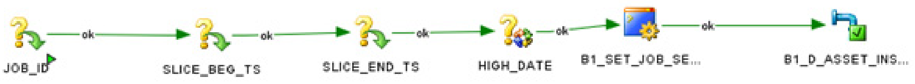
```
(CASE WHEN SRC.<src_nk> IS NULL
      THEN #B1_DEF_NULL_KEY
      WHEN <dim>.<dim_pk> IS NULL
      THEN #B1_DEF_MISSING_KEY
      ELSE <dim>.<dim_pk>
  END)
```

In the sample code mentioned above, SRC is the alias of the primary source table that contains the source foreign key "<src_nk>" to the dimension. <dim> is the alias for the dimension and <dim_pk> is the primary key of the dimension.

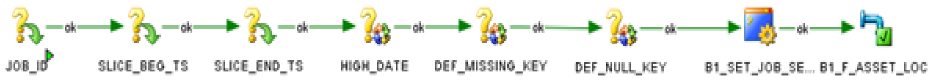
Creating a New Package

A new package has to be created for each target entity. The easiest way to create a package for the entity is to copy an existing package based on the target object type and replace the interface with the new interface. While copying the package, remember to move it to the custom folder.

Below is a sample package for a type 2 dimension.



Below is a screenshot for a fact package.



Once the package has been created, right-click and generate the scenario. Ensure that the new package name has been configured appropriately in the **Job Configuration** section using the Admin user interface.

Creating a New Interface for the Materialized View

To create a new interface for your materialized views, perform the steps below:

1. Under the custom folder, right-click and select the new interface.
2. Drag and drop the source tables (facts and dimensions from target model) into the source section of the interface.
3. Set up the appropriate join conditions between the tables.
4. Map the columns from the source to the target as per your design.
   A new package needs to be created for each materialized view interface. The package should contain the global variable **B1_JOB_ID**.

Creating a New Analytics

This section describes how to use Oracle Business Intelligence Enterprise Edition (OBIEE) to add new analytics. It includes the following topics:

- **Creating the New Answers**
- **Adding the New Labels**

Creating the New Answers

Note: Before creating Oracle Warehouse Builder workflows, it is recommended that you should have knowledge of Oracle Business Intelligence Enterprise Edition and Data Warehouse concepts. It is also recommended that you should have experience in developing reports using Oracle Business Intelligence Enterprise Edition.

Oracle Utilities Analytics (OUA) provides out-of-the-box dashboards with rich and varied set of analytics for Credit and Collection Analytics, Customer Analytics, Distribution Analytics, Meter Data Analytics, Mobile Workforce Analytics, Outage Analytics, Revenue Analytics, Exception Analytics, and Work and Assets Analytics. However, if required, you can create new answers, or dashboards.

As noted in the above-mentioned section regarding customization of the existing answers, the new answers should also be saved in a separate folder so that they are not overwritten while upgrading Oracle Utilities Analytics.

An implementation can create field labels for use in their answers, or the labels can just be created directly in the answer if there are no multilingual/localization requirements. If the product labels are used in an answer, they can be modified during an upgrade (unless you have entered an override label). At best, it is recommended to limit the changes to the existing labels; however, there can be certain situations, when they are updated.

Adding the New Labels

To use the label mechanism for new answers, the **Custom Field Maintenance** dashboard can be used to add, update, and delete custom labels. These custom labels can then be used in answers as well as in the logical and physical objects in the repository or RPD file.

> **Note**: Only custom field labels, identified by a C**ustomer Modification (CM)** owner flag, can be updated or deleted. The new labels are created with a **Customer Modification (CM)** owner flag. A label that already exists cannot be created, so if a base labels already exists, you can update the override label as described in the preceding section **Creating the New Answers**.

> **Note**: For more details, refer to the '**Maintaining the Administration Dashboards**' section in the **Chapter 4**: **Configuring Oracle Utilities Analytics**.

Chapter 7

Maintaining Environments

This section describes how to maintain your Oracle Utilities Analytics (OUA) environments, including moving code changes from one environment to another.

This section includes the following topics:

- **Overview of Environment Maintenance**
- **Moving Oracle Warehouse Builder Based Code**
- **Cloning a Database for Oracle Warehouse Builder**
- **Maintaining Products Integrated Using Oracle Data Integrator**

Overview of Environment Maintenance

You should implement processes to maintain code control over various environments. The following components of Oracle Utilities Analytics (OUA) should be handled separately.

- Oracle Business Intelligence Enterprise Edition web catalog
- Oracle Business Intelligence Enterprise Edition repository file
- Field Label metadata
- Oracle Warehouse Builder repository
- Oracle Utilities Application Framework (OUAF) metadata (only required if there are Oracle Warehouse Builder customizations)
- Mapviewer spatial data

Assuming that the custom changes are made to any of these objects, then a mechanism must be put in place to develop, test, and move these customizations to the production environment.

Moving Oracle Warehouse Builder Based Code

During the development phase of coding, you usually do not need to move code from a development environment to any other environments. However, once a project moves to the quality analyst or production phases, code changes may be needed to be migrated.

For example, in an internal development process, there may be two development environments: one for Oracle Business Intelligence Enterprise Edition (OBIEE) dashboard creation and one for Oracle Warehouse Builder (OWB) development. You can build a QA environment from the scratch: create an empty database, the Oracle Warehouse Builder objects get imported and deployed, and the Oracle Business Intelligence Enterprise Edition web catalog and repository file get created fresh.

To do this, use the installation process for creating an empty Oracle database, Oracle Warehouse Builder repository, and WebLogic environment. Then, use the Oracle Warehouse Builder Export process to create two MDL files from the development environment: one for the locations, and one for all of the other Oracle Warehouse Builder objects. You should then copy the Oracle Business Intelligence Enterprise Edition repository file, and use the Oracle Business Intelligence Enterprise Edition export process to create the Oracle Business Intelligence Enterprise Edition web catalog files.

Once you have these files, follow the install process to load the MDL files in the Oracle Warehouse Builder repository, copy the Oracle Business Intelligence Enterprise Edition repository file into the Quality Analysis (QA) WebLogic environment, update the user name, password, and database connections for the quality analysis databases, and import the web catalog export files into the same location as they were in the development environment.

This process works well for a development move to QA, but does not work once a system goes into production, because parts of this process require the creation of an empty database, which is not something that should be done in a production environment.

In situations, where bug fixes have been made in a development environment and they need to be moved to a production environment, you can export the entire Oracle Warehouse Builder repository and Oracle Business Intelligence Enterprise Edition web catalog and replace this in the production environment. Use this method to move the code if it is not known exactly which objects have changed.

For Oracle Warehouse Builder though, if the modified objects are known, then it is possible to export only the changed objects, import them in the QA environment, and then, once QA is successful, do the same import process into the production environment. Another option is to save the TCL files that were created by the Oracle Warehouse Builder Code Generator, and then load them into the QA and production environments.

Creating an MDL File

To create an MDL file for a known set of Oracle Warehouse Builder objects, follow these steps:

1. Log onto the Workflow Development Database using the Design Center as the Repository Owner (BIREPOWN).
2. Review the modified the Oracle Warehouse Builder (OWB) objects, and then select the Oracle Warehouse Builder objects.
3. Export them by navigating to **Design > WareHouse Builder Metadata**.
4. Review the path for MDL and log file, and select the Export All Dependencies.
5. Click **Export.**
6. Use the created MDL file to move objects from a development environment to a QA, or production environment.

This process assumes that database changes are handled outside of Oracle Warehouse Builder. Hence, new tables, modifications to existing tables, or materialized view log changes should be handled via the SQL scripts created by the development team.

For Oracle Business Intelligence Enterprise Edition (OBIEE), a full move is not recommended; there are ways of merging code in the web catalog and also in the repository file. These methods are documented in *Oracle Business Intelligence Enterprise Edition Administration Guide.*

> **Note:** Refer to the chapter 24, "**Moving Between Environments**," in *System Administrator's Guide for Oracle Business Intelligence Enterprise Edition 11g Release 1.* This chapter shows how you can move Oracle Business Intelligence Enterprise Edition to a new environment, or from a test to a production environment.

For the MapViewer data, modified override labels are added to the QA, or the production environments the same way that it is added to the development. If a shapelier is downloaded and added to the development environment, the same shapelier should be added to the QA and the production environments.

Finally, it is very important that the process of moving code from the development to the QA is exactly the same as the process that moves the code to the production. It is only by following the same sequence of steps in both cases that the movement process is also tested. If one process is followed to move code to the QA and another process is followed to move code to the production, then problems can arise in the move to the production that were not seen in the move to the QA environment.

Cloning a Database for Oracle Warehouse Builder

You can use database cloning to move the entire development database to a QA environment, or a new production environment. This cannot be used to move a development database to an existing production environment, as the existing production database tables are overwritten. But for QA purposes, this can move the development database quicker than a fresh install.

> **Note**: This does not move the Oracle Business Intelligence Enterprise Edition objects, but does handle all of the Oracle Warehouse Builder code, field label changes, marvin meditated, and any new database objects.

Cloning a Development Database

To clone a development database, follow these steps:

1. Clone the existing database.
2. Go to the **$ORACLE_HOME/owe/Underpays** directory.
3. Execute the **reset_owbcc_home.sql** with the OWBSYS user and **remote_owb_install.sql** scripts with the sys user.
4. Create a password file for the database if it does not exist.
5. Go to the **$ORACLE_HOME/owe/repossess/up** directory and execute the **upg112to11203.sql** as **scalpels /no-load @upg112to11203.sql** if database is cloned from the lower Oracle Home version to Oracle 11.2.0.3 version home and this script is already not executed.
6. Wait till execution of this script completes. It takes time.
7. Connect as the OWBSYS user and execute:
   - @$ORACLE_HOME/owb/reposasst/secHelper.pks
   - @$ORACLE_HOME/owb/reposasst/secHelper.plb
   - ORACLE_HOME/owb/reposasst/upg/load_java.sql OWBSYS <OWBSYS_PASSWORD>
8. Navigate to **$ORACLE_HOME/owb/bin/admin** and rename the **rtrepos.properties**.
9. Connect as the **OWBSYS user**, and execute the **@$ORACLE_HOME/owb/rtp/sql/ reset_repository.sql**.
10. Submit the **SELECT * FROM OWBRTPS** query.
    It shows updated Oracle Home in the **Value** column.
11. Submit select **SERVER_SIDE_HOME** from the **WB_RT_SERVICE_NODES** query.
    It shows the updated Oracle Home.
12. Log into the design repository and get names of all the Control Centers with which locations are registered.
13. Connect with the **OWBSYS** and execute the **UpdateControlCenter.sql** for all the Control Centers with which locations are registered. Provide the below inputs:

Enter Workspace Name: SPLBIREP

Enter Workspace User Name: BIREPOWN

Enter Control Center Name: <Control Center Name>

Host: <Hostname>

Port: 1521

Service Name: <DB name

New Net Service Name: <DB name>

14. Run **@UpdateLocation.sql** for all the locations. Make sure to provide the correct version.

    For example:

    Enter Workspace Name: SPLBIREP

    Enter Workspace User Name: BIREPOWN

    Enter Location Name: SPL_BI_TGT_LOC

    New Host: <Hostname>

    New Port: 1521

    New Service Name: <DB name>

    New Net Service Name: <DB name>

    New Version: 11.2

    Provide version 0 for SPL_BI_FF_LOC and SPL_BI_LOG_LOC locations and 2.6.4 for SPL_BI_WF_LOC location.

    You can also use the below block to update the workflow location:

```
    Declare
    v_result boolean := FALSE;
    Begin
    v_result := wb_rt_reset_location.fixDTLocation
    ('SPL_BI_WF_LOC','<Hostname>','1521', '<DB name>','2.6.4', '<WF
user>','<WF user's password>');
    END;
/
```

15. EXECUTE UPDATE WB_RT_SERVICE_NODES SET CONNECT_SPEC='<hostname>:1521:<DB name>'; commit;
16. Connect to the Control Center Manager in the Design repository.
17. Register all the locations.
18. Unregister all the locations.
19. Save all objects, and exit from the Control Center Manager.
20. Double-click the Control Center and move all the selected locations to the available locations. and click **OK**.
21. Rename the Control Center as required.
22. Set the other Control Center with which the locations are registered as default Control Center for default_configuration and connect to the Control Center.
23. Register all the locations.
24. Unregister all the locations. Save all the objects, and exit from the Control Center
25. Double-click the Control Center, move all the selected locations to the available locations and click **OK**.

26. Rename the Control Center as required.

27. Repeat steps from 22 to 26 for all the registered Control Centers.

    Doing so, you can remove all the Control Center info from the **Registration** tab of all the locations.

    If you are not removing the Control Center dependency, you can register location with any Control Center, but will not be able to update location info after you unregister the location.

28. Move the locations from available section to selected location in the required Control Center, set the required Control Center as the default Control Center for default_configuration and log into the Control Center.

29. Now all the locations are available for the updates and can be registered followed by objects deployment.

Maintaining Products Integrated Using Oracle Data Integrator

This section includes topics:

- **Moving the Oracle Data Integrator Repository**
- **Moving the Metadata**
- **Purging**

Moving the Oracle Data Integrator Repository

To move an Oracle Data Integrator repository, perform the following steps.

1. Log into the Oracle Data Integrator client using a supervisor user.
2. Export the master and work repository using the Oracle Data Integrator client.
3. Log out of the Oracle Data Integrator client.
4. Use the master repository creation wizard and create a new repository if it does not exist.
5. Log into the new master repository and import the XMLs exported earlier.
6. Modify the connections in the Topology section as required.
7. Delete/drop the older Oracle Data Integrator master and work repository schemas, or alternatively delete all connections, users from the **Topology and Security** sections respectively. This is a precautionary action to prevent any users from accidentally logging into the older repository and executing jobs.

Moving the Metadata

The metadata objects reside in the DWADM schema by default. In case you have made some configuration changes and want to change to a different database, export the metadata objects and import them into the new environment.

Purging

A data warehouse is designed to accumulate data across many years. Over a period of time, the data is accumulated in the data warehouse. The data can consist of the following

- Star Schemas
- Logs (Audit/Process/Error)
- Replicated Data
- Staging Data

The target data warehouse data is usually expected to be retained for periods longer than seven years and even when the threshold is reached, the older data is expected to be archived. Since the archival routines and rules may differ for different implementers, archival of the target data warehouse is not covered here.

Depending on the policies in place, it is always advisable to take a backup before purging data. A routine backup should be set up for the data warehouse.

This section provides details on what can be purged and how to purge. Not all topics are applicable in all the situations. The following topics are covered here:

- **Tracking the Outage Data in Oracle Utilities Network Management System**
- **Purging the Oracle Warehouse Builder Audit Logs**
- **Purging the Oracle Data Integrator Session Logs**
- **Purging the Staging Data**
- **Purging the Replication Data**

Tracking the Outage Data in Oracle Utilities Network Management System

The Oracle Utilities Network Management System RDBMS is generally not intended to be in the long-term (longer than a year or two) repository for the historical customer outage data. Rather Oracle Utilities Network Management System (NMS) is intended to track the current and relatively recent status of the utility load area infrastructure. The Oracle Utilities Network Management System RDBMS generally includes a record of relatively recent customer outages for the operational reference/review purposes. Beyond this project specific “reasonably recent” window, it is expected that some type of Oracle Utilities Network Management System outage data archive/purge process is executed on a regular (daily or weekly) basis to keep the Oracle Utilities Network Management System RDBMS instance relatively lean and performing optimally.

The Oracle Utilities Analytics data warehouse is intended to be the long-term home for the historical customer outage data. However, the Oracle Utilities Analytics data warehouse tracks two different types of outages in two sets of Oracle Utilities Analytics data warehouse tables and only one of them is considered a long-term repository. The Oracle Utilities Analytics data warehouse tracks both recent and restored outages:

- **Recent outages**: It is also called as Near Real Time (NRT) outages. The primary purpose of the NRT data store is to support tracking current (active) and relatively recent completed outages. The NRT data store can also be used to help gauge the ability of the existing resources to deal with a current storm to help determine if external/foreign (crew) resources are or are not be required. Just like the Oracle Utilities Network Management System RDBMS data store, the Oracle Utilities Analytics data warehouse “recent outage” data store is not intended to be a long-term repository.
- **Restored outages:** It is also called as historical outage data. The primary purpose of the restored outage tables is for the long-term reporting purposes. The restored outage data store should be the most accurate data store and is intended to support the required regulatory customer impact reports. This data store has no periodic purge requirements. The historical outage data is intended to be held in this data store indefinitely.

Purging the Oracle Warehouse Builder Audit Logs

Note: This is applicable to the Oracle Warehouse Builder based ETL only.

The following topics are discussed in this section:

- **Configuring Oracle Warehouse Builder**
- **Purging ETL Job Control Tables**

Configuring Oracle Warehouse Builder

Oracle Utilities Analytics (OUA) utilizes Oracle Workflow when running Oracle Warehouse Builder (OWB) process flows to load extract files into the data warehouse. Even if the extract files are not present, the records are created in the audit tables each time a process flow is run. Depending on the frequency with which process flows are scheduled, these audit tables can grow to become unmanageable and can cause upgrades or process flow changes to fail when deployed.

A few of these audit tables include the run-time Oracle workflow audit tables and can grow very large:

- **WF_ITEM_ATTRIBUTE_VALUES**: This table stores the run-time values of the item attributes for a particular process flow.
- **WF_ITEM_ACTIVITY_STATUSES**: This table, along with the **WF_ITEM_ACTIVITY_STATUSES_H** contains all of the activities executed by a specific occurrence of a process flow.
- **WF_NOTIFICATION_ATTRIBUTES**: This table contains the run-time values of all the message attributes for a specific notification.
- In addition, Oracle Warehouse Builder (OWB) also contains audit tables that can also grow very large if not purged periodically.

Oracle Utilities Analytics includes a purge process flow that calls Oracle Warehouse Builder and Oracle Workflow Application Programming Interfaces (APIs) to purge these audit tables as well. The **OUBIWF_PURGE_RT_AUDIT** process flow in the **INIT_PKG** package is set up to purge audit data that is older than one month.

The **OUBIWF_PURGE_RT_AUDIT** process flow is not run from the File Processor daemon, so you must schedule it using a scheduler tool that can run Oracle Warehouse Builder process flows. You can also schedule the procedure that this process flow calls, **OUBI_PURGE_RT_AUDIT_DATA_PRC** using a tool that can call a PL/SQL command. This procedure requires no parameters, and can be called directly from a PL/SQL block, like this:

```
BEGIN
OUBI_PURGE_RT_AUDIT_DATA_PRC;
END;
```

You should run either this purge routine, or the Oracle Warehouse Builder and Oracle Workflow purge routines at least monthly, so that the audit tables remain small. It is recommended to purge these tables at least once a month.

Purging ETL Job Control Tables

Note: This is applicable to the Oracle Warehouse Builder based ETL only.

It is recommended that the Extract, Transform, and Load Job Control tables be purged on a regular basis. For analysis purposes, it is suggested to retain 30-90 days of data, but depending on your need this value should be appropriately adjusted.

The ETL Job Control table resides in the DWADM schema on the Oracle Utilities Analytics database instance. A sample script to purge the ETL Job Control table is shown below.

Provide an appropriate value for the number of days for which the data needs to be retained in the ETL Job Control table.

```
   delete from b1_etl_job_ctrl
   where start_dttm < sysdate -&days_to_retain and end_dttm is not
null
   and job_status_flag = 'JC';
   commit;
```

Purging the Oracle Data Integrator Session Logs

Note: This is applicable to Oracle Data Integrator ELT only.

Oracle Data Integrator session logs that are older than a calculated date are purged periodically. The date is calculated as below:

purge date = Least of (current date - b1_retain_days, the oldest session start where job is in error and has not been reprocessed)

The Oracle Data Integrator global variable b1_retain_days is set to 1 by default. It can be configured using Oracle Data Integrator client to a different value.

Purging the Staging Data

Note: This is applicable to Oracle Data Integrator ELT only.

There is one staging table for each target entity. The staging data is useful for tracing the data propagation and debugging in case of issues. The retention period is configured in the metadata table **B1_TARGET_ENTITY** using the column **STG_RETAIN_DAYS**. The default value is 7.

The staging data older than configured retention period is automatically purged as a part of the execution of the interface.

Purging the Replication Data

Note: This is applicable to Oracle Data Integrator ELT only.

Each source table that has been marked for replication has a replica table created in the replication schema. The tables that are used as source for type 2 dimensions retain history for each change. Over a period of time, the volume in the replication area can grow huge. It is recommended to retain the data in the replication layer for as long as it is feasible to do so considering the data storage requirements as this enables you to reload the warehouse at any given point of time. This comes in handy when additional user-defined columns are being added and to reload all entities where the new user-defined column has been configured for load.

The table **B1_SOURCE_TABLE** has a column **REPL_RETAIN_DAYS**, which can be modified to control the purge in the replication area.

Note:

A minimum retention period of 30 days is enforced.

If there are multiple target entities dependent on the source table, then the least of the last sync timestamp of the entities is considered.

If the least of last_sync_dttm is older than the retention period, then the data older than this data is purged.

Chapter 8

Licensing and Optional Features

This chapter describes about licensing and optional features, including:

- **Oracle Database Licensing and Optional Features**
- **Oracle Warehouse Builder Licensing and Optional Features**
- **Disabling the Optional Features in Oracle Warehouse Builder**
- **Oracle GoldenGate Licensing**

Oracle Database Licensing and Optional Features

With Oracle Utilities Analytics v2.5.1, the **Standard Edition** of Oracle database is now supported. Currently, this support is only for the customers using those Extractors and Schema products, which are based on Oracle Data Integrator (ODI) based extraction, and includes:

- Oracle Utilities Operational Device Management Extractors and Schema
- Oracle Utilities Customer Care and Billing Extractors and Schema
- Oracle Utilities Network Management System Extractors and Schema

However, Oracle recommends using the **Enterprise Edition** of Oracle database for performance and scalability reasons, as the Oracle Utilities Analytics data warehouse is expected to handle large volumes of data. Oracle Utilities Analytics also supports the *Oracle Partitioning* feature, which is an extra cost option on the top of the Oracle Database Enterprise Edition. Using this feature, it is also recommended for the efficient data storage and retrieval by Oracle Utilities Analytics.

By default, the *Enterprise Edition* and the *Partitioning* features are turned off in Oracle Utilities Analytics. Once the appropriate licenses have been purchased, these features can be turned on in the **Global Configuration** settings using the Oracle Utilities Analytics Administration Tool described in **Appendix B**.

Oracle Warehouse Builder Licensing and Optional Features

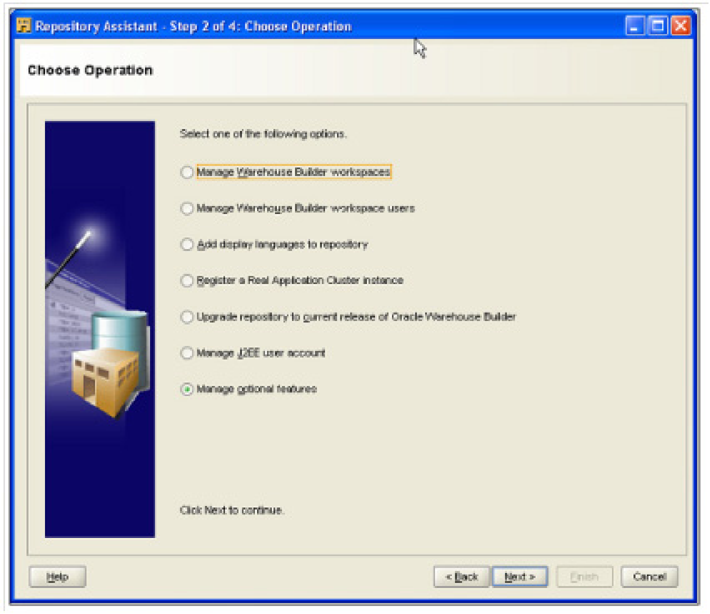
Oracle Warehouse Builder (OWB) provides various optional features, which are not included in the basic Extraction, Transformation, and Loading (ETL) feature group. The basic ETL feature group is included in the Oracle Database Enterprise Edition license. Hence, there is no additional license cost required to use, or install the basic features. The standard ETL processes included in Oracle Utilities Analytics (OUA) uses only the features that are included in the basic ETL feature group.

In addition, the Oracle Warehouse Builder Code Generator does not create any code that requires the use of optional Oracle Warehouse Builder features. Hence, any additional Extraction, Transformation, and Loading (ETL) code created by an implementation using the Oracle

Warehouse Builder Code Generator does not require any additional Oracle Warehouse Builder license costs. However, if Oracle Warehouse Builder is used to create other ETL code outside of the Oracle Warehouse Builder Code Generator, then using some of these optional features may require additional Oracle Warehouse Builder licenses.

Disabling the Optional Features in Oracle Warehouse Builder

In order to ensure that optional features are not used, Oracle Warehouse Builder (OWB) provides a means to disable the use of optional features. After starting the Warehouse Builder Repository Assistant, choose the "**Manage optional features**" operation, as shown in the following screenshot.



Repository Assistant - Step 2 of 4: Choose Operation

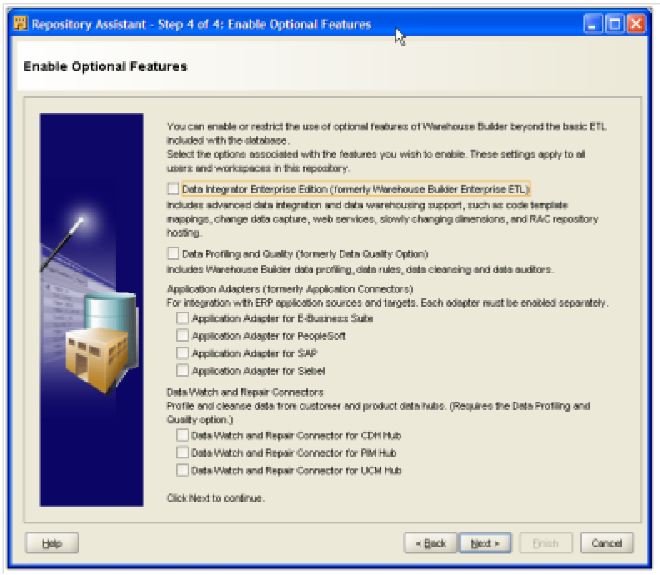
Choose Operation

Select one of the following options.

- ( ) Manage Warehouse Builder workspaces
- ( ) Manage Warehouse Builder workspace users
- ( ) Add display languages to repository
- ( ) Register a Real Application Cluster instance
- ( ) Upgrade repository to current release of Oracle Warehouse Builder
- ( ) Manage J2EE user account
- (✓) Manage optional features

Click Next to continue.

After entering the password for the OWBSYS user, deselect all of the licensed option names on the **Enable Optional Features** page.



Repository Assistant - Step 4 of 4: Enable Optional Features

Enable Optional Features

You can enable or restrict the use of optional features of Warehouse Builder beyond the basic ETL included with the database.
Select the options associated with the features you wish to enable. These settings apply to all users and workspaces in this repository.

- [ ] Data Integrator Enterprise Edition (formerly Warehouse Builder Enterprise ETL)

Includes advanced data integration and data warehousing support, such as code template mappings, change data capture, web services, slowly changing dimensions, and RAC repository hosting.

- [ ] Data Profiling and Quality (formerly Data Quality Option)

Includes Warehouse Builder data profiling, data rules, data cleansing and data auditors.

Application Adapters (formerly Application Connectors)
For integration with ERP application sources and targets. Each adapter must be enabled separately.

- [ ] Application Adapter for E-Business Suite
- [ ] Application Adapter for PeopleSoft
- [ ] Application Adapter for SAP
- [ ] Application Adapter for Siebel

Data Watch and Repair Connectors
Profile and cleanse data from customer and product data hubs. (Requires the Data Profiling and Quality option.)

- [ ] Data Watch and Repair Connector for CDH Hub
- [ ] Data Watch and Repair Connector for PIM Hub
- [ ] Data Watch and Repair Connector for UCM Hub

Click Next to continue.

Once the options are deselected, the new selections take effect for any new connections to Oracle Warehouse Builder, and if options are used that are not available, an error dialog is displayed.

Note: For further details regarding the feature groups and licensing of Oracle Warehouse Builder, visit the Oracle Warehouse Builder page on the Oracle Technology Network (OTN) at this location: http://www.oracle.com/technetwork/developer-tools/warehouse.

Oracle GoldenGate Licensing

The Oracle GoldenGate license purchased along with Oracle Utilities Analytics v2.5.1, can be used to replicate additional tables from the any of the source systems products whose ELT processes are based on Oracle Data Integrator (ODI), which includes:

- Oracle Utilities Operational Device Management
- Oracle Utilities Customer Care and Billing
- Oracle Utilities Network Management System

For any other usage of the Oracle GoldenGate product, you need to purchase a full license of the Oracle GoldenGate product. You are requested to contact Oracle Support for the details.

Appendix A

Oracle Utilities Application Framework Extractors

This section contains the following topics and describes the general structure and process followed for the extractors for Oracle Utilities Application Framework based applications, such as Oracle Utilities Work and Asset Management:

- **Change Detect Mechanism**
- **Fields in the Change Log**
- **Structure of the Triggers**
- **Rows in the Change Log**
- **Extracting and Transforming the Data**
- **Extracting the Data**
- **Parameters Supplied To the Extract Processes**

Note: This section applies to Oracle Utilities Work and Asset Management.

Change Detect Mechanism

Every production database table used to populate the data warehouse must be monitored for changes so that these changes can be reflected in the data warehouse. The triggers insert a row into the change log when the source tables change.

Fields in the Change Log

The sole job of the triggers is to populate the change log. Therefore, you must understand the fields of the change log table in order to understand the triggers. The change log contains the following primary fields

Field	Purpose
Change Log ID	This is a random prime key of the change log, and is generated by the trigger.
Batch Code	This is the code for the extract process that processes this change.
Batch Number	This is the current run number for the extract process.
Change Date and Time	The date and time of the change.

Field	Purpose
Change Type	This indicates if a row in the table was inserted, updated, or deleted.
Table Name	The name of the table that is changed.
Prime Key 1 - 5	The prime key of the object that is affected. The change log accommodates prime keys with up to five parts. The prime key stored on the change log is not the prime key of the record that is changed, but the prime key of the object. For example, if the phone number of a person is changed, these prime key fields contain the prime key of the person object, not the prime key of the phone number record. When any field on an object is changed, the entire object is re-extracted.

Structure of the Triggers

All the triggers populate the change log, and they are similar in the following ways:

- Determine if a row needs to be inserted into the change log. All table changes do not need to be reflected in the data warehouse, so not all changes need to be noted in the change log. For example, if an unfrozen financial transaction is created, a change log record does not need to be inserted if the data warehouse only tracks frozen financial transactions.
- Generate a prime key for the change log.
- Know the codes for the appropriate extract processes that handle the table change.
- Retrieve the current run numbers for the extract processes.
- Determine the prime key of the main object.

Rows in the Change Log

A record in the change log is processed by only one extract process. If multiple extract processes are needed to handle a single change in a source table (for example, if a new object requires the addition of multiple facts or dimensions), then multiple rows are inserted into the change log. This can be accomplished with one trigger inserting multiple rows into the change log, or with multiple triggers on the same table, each trigger inserting one row.

Extracting and Transforming the Data

Oracle Utilities Work and Asset Management use batch controls with an underlying extract program to generate the flat files based on the change log tables populated by the triggers.

Extracting the Data

Most extract programs support two modes of execution (you can control the mode by a parameter supplied to the extract process):

- **Extract Everything Mode, or Initial Extract**: This mode extracts every row on the operational table. You can use this mode to instantiate the data warehouse. For example, if you run the extract accounts program in "extract everything mode", every account is extracted.
- **Extract Recent Changes Mode, or Incremental Extract**: This mode only extracts data that is added or changed since the last time the extract was executed. For example, if you run

the extract accounts program in "extract recent changes mode", every account that is added or changed since the last execution is extracted.

Parameters Supplied To the Extract Processes

All the extract processes are submitted in their source system (For example, programs that extract data from Oracle Utilities Work and Asset Management are submitted in Oracle Utilities Work and Asset Management). The following points describe the parameters that are supplied to these processes for Oracle Utilities Work and Asset Management:

- **Batch Code**: The batch code is an unique identifier of the extract process. The batch code for each extract process is identified in the description of the various facts and dimensions.

  Note: Refer to the appropriate fact and dimension chapter for the details in the Data Mapping Guides.

- **Batch Thread Number**: The thread number is only used for extract processes that can be run in multiple parallel threads. It contains the relative thread number of the process. For example, if the arrears process has been set up to run in 20 parallel threads, each of the 20 instances receives its relative thread number (1 through 20).

  Note: Refer to the section **Optimal Thread Count for Parallel Background Processes** in the **Background Process** chapter of the source system's data mapping guide for more information.

- **Batch Thread Count**: The thread count is only used for the extract processes that can be run in multiple parallel threads. It contains the total number of parallel threads that have been scheduled. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives a thread count of 20.

  Note: Refer to the section **Optimal Thread Count for Parallel Background Processes** in the **Background Process** chapter of the source system's data mapping guide for more information.

- **Batch Rerun Number**: Rerun number should only be supplied if you need to download an historical run (rather than the latest run).

- **Batch Business Date**: The business date is only used for the extract processes that use the current date in their processing. For example, the Oracle Utilities Customer Care and Billing arrears extracts use the business date to extract arrears as of a given date. If this parameter is left blank, the system date is used. If supplied, this date must be in the format YYYY-MM-DD. This parameter is only used to test how processes behave over time.

- **Override Maximum Minutes Between Cursor Re-initiation**: This parameter is optional and overrides each extract process's standard cursor re-initiation minutes. Each extract process re-initiates cursors every 15 minutes. You can reduce these values. For example, if you are submitting a job during the day and you want more frequent commits to release held resources (or more frequent cursor initiations). You can increase these values when an extract process is executed at night or on weekends, and you have sufficient bandwidth and memory available on the servers. The maximum minute between cursor re-initiation parameter is relevant for Oracle implementations only. Most of the system extract processes contain an outermost loop/cursor. The cursor is opened at the beginning of the process and closed at the end. If Oracle detects that the cursor is open for too long, it may incorrectly interpret this as a problem and displays an error that the snapshot is too old. The processing for the extract processes is designed to refresh the cursor based on the minutes between cursor re-initiation in order to prevent this error.

- **User ID**: The following must be ensured with respect to a user ID:

  - **User ID:** The **User ID** is a user who has access to all application services in the system as some batch processes call application services to perform maintenance functions (for example, when an account is updated, the batch process may call the account maintenance application service). The display profile of the user ID controls how dates and currency values are formatted in messages.

- **Password**: Currently, the password is not used.
- **Language Code**: All language-sensitive data is extracted in this language. In addition, all error messages are presented in this language.
- **Trace program at Start (Y/N), Trace Program Exit (Y/N), Trace SQL (Y/N) and Output Trace (Y/N)**: These switches are only used during Quality Analysis (QA) testing and benchmarking. If the trace program start is set to **Y**, a message is displayed whenever a program is started. If the trace program at exist is set to **Y**, a message is displayed whenever a program is exited. If the trace SQL is set to **Y**, a message is displayed whenever an SQL statement is executed. If the output trace is set to **Y**, special messages formatted by the extract process are written.

The information displayed when the output trace switch is turned on depends on each extract process. It is possible that an extract process displays no special information for this switch.

- **Initial Load Switch**: This switch controls whether the extract program is run in extract everything mode or extract recent changes mode.
- **File Path and File Name**: These parameters define the file path and/or file name for the output file. When supplying a file-path variable, the directory specified in the file-path must already exist and must grant write access to the Oracle Utilities Analytics administrator account. You may need to verify a proper location with your system administrator. The syntax of the file-path depends on the platform used for Oracle Utilities Analytics application server. Contact your system administrator for verification. For example, if the platform is UNIX, use forward slashes and be sure to put a trailing slash, such as/spltemp/filepath/.

  Note: The control file is created with the same name as the data file, but with a fixed extension of CTL. For this reason, do not use CTL as the extension when defining value for the file-path parameter.

In order to avoid overwriting the flat files generated during the previous execution, the extractor programs insert a string containing the concatenated values of data source indicator, batch number, and the batch thread number in the name of the generated data and the control file. The value is inserted just before the extension of the file name specified.

- **Maximum Errors**: This parameter is not currently used.
- **User-Defined File UDF and UDMs**: Refer to extending extractors for the details on how to extend the various UDF and UDM fields.

Appendix B

Oracle Utilities Analytics Administration

The Oracle Data Integrator based Extract, Load and Transform (ELT) architecture is a metadata driven framework that allows you to configure the out of the box ELT jobs, as well as making it easy to integrate custom ELT jobs to Oracle Utilities Analytics. The out-of-the-box ELT jobs are set up during the installation of the product. However, you can alter or maintain some of the parameters using Oracle Utilities Analytics Administration Tool.

Note: Refer to the **Chapter 9**: **Installing the Oracle Utilities Analytics Admin Tool** in *Oracle Utilities Analytics Installation Guide* for details on how to install and access the Administration Tool.

This section contains the following topics:

- **Metadata Tables**
- **Configuring the Product Instance**
- **Configuring the Target Entities**
- **Configuring the Jobs**
- **Configuring the Source Table**
- **Executing a Job**
- **Configuring Oracle GoldenGate**
- **Configuring the Global Settings**

Metadata Tables

The following table lists the metadata tables used to store this metadata, and the purpose of each.

Entity	Purpose
Product	This table is used to identify, which products are supported, and to identify the Oracle Data Integrator scenario (accelerator) to be used to import source metadata and configurations. This table is for the internal use.

Entity	Purpose
Product Instance	You may have multiple instances of the same product that you want to integrate with Oracle Utilities Analytics solution. The instance object represents each instance of the same product that can be utilized as a source. The objective is to enable development of a single interface that can be utilized across multiple instances. Instances differ by the source database connections and possibly by the differing configurations. The data source indicator is unique across instances and products as this allows traceability to identify the source of the data in the data warehouse.
Target Entity	This table holds the configurations for target entities in the data warehouse. The target entities can be dimensions, facts, or materialized views. The same target entity may be loaded from multiple source instances. The configuration that is common across multiple instances is stored in this table.
Job Configuration	This table is used to provide configuration for the package (the executable logic) for populating the target entities for each instance. This table stores the current progress and an override package for use when the logic for one instance differs from the logic used for the other instances.
Object Map	The table contains mapping between the various source MOs, or the BOs to the target entity that they are used to populate.
Source Tables	This table contains configurations controlling, such as tables to be replicated and the mode of replication.
Job Executions	This table is used for tracking the execution of the ELT processes. An entry is created for each execution. Some attributes are populated from the SNP_SESSION table, which are used by Oracle Data Integrator to track sessions.

Entity	Purpose
Dependencies	This table is used to map a job to the dependent jobs. The dependent objects are only to be executed up to the minimum sync timestamp of all the dependencies. For example: • The Operational Device fact is associated with three type 2 dimensions (Operational Device, Utilities Asset, and Address). • Assuming that the Operational Device fact is scheduled to run at 11 AM, and the Address dimension is scheduled to run at 7 AM. • The Address dimension would have processed all changes to address till 7AM. Assume an address location exists for a warehouse location in NY with ID101, the effective start date of 01/10/2010 and the effective end date of 31/12/4000. • A device was added to the system at 9 AM and a change is made to the NY location 101. • At 11 AM when the fact is processed, since the Address dimension is loaded only till 7 AM, the new history starting 7/30/2013 09:00 AM is not yet in the dimension and this ends up referencing the older record. • To avoid this, the fact is loaded only up to the last load timestamp of the Address dimension, which in this example is 7 AM.
DSI Mapping	A source system may be integrated with other source systems. When pulling information from multiple source instances it is possible that the data in one instance refers to master data from another instance. This table allows such cross references to be configurable. By default, the configuration is at an instance level that is a product and instance number level. However, more fine grained control is possible by specifying the entity for which the configuration should be applied. **Note**: You need to configure data for this entity on the source application. For details, refer to the '**Configuration**' chapter in the Data Mapping Guides of the respective source application.
Oracle GoldenGate Configuration	This table is used to provide configuration details to be utilized for the Oracle GoldenGate script deployment on the source and target environments. An entry needs to be created for each source instance providing details, such as host, port, SID, Oracle home path etc.
Oracle GoldenGate Checkpoint	This table is used by the Oracle GoldenGate replication process to track its processing activities. This table is used by the scheduler process to ensure that the warehouse loading tasks only process data that has been synced. The structure is controlled by Oracle GoldenGate.

Entity	Purpose
Extract/Job Parameters	This table stores configurations that control the extraction of data. Each source system may have different codes used for the different purposes. This table allows you to configure the codes so that the extraction routine can utilize it appropriately. **Note**: You need to configure data for this entity on the source application. For details, refer to the '**Configuration'** chapter in the Data Mapping Guides of the respective source application.
Global Configuration	This table stores the global settings that are required for Oracle Utilities Analytics. The examples of these configuration settings are the database edition type owner by you, cut-off time to be used by ETL jobs, global extract date for the initial load, etc.

Many of the metadata tables mentioned above can be configured / maintained with the help of the Administration Tool. The Administration Tool is delivered along with Oracle Utilities Analytics product to allow you to maintain these tables. This Administration Tool has been built on the top of the **Oracle Application Express** feature, which is available with the Oracle database. The following sections explain the details of the metadata tables that you can maintain using the Administration Tool.

Configuring the Product Instance

The Product Instance represents a specific instance of a source application that can be configured as a source for Oracle Utilities Analytics solution. A record is created for every instance of the product for which ELT is setup to extract data. Oracle Utilities Analytics delivers a record for the product which uses Oracle Data Integrator for the ELT. You may have multiple instances of the same source application to integrate with the Oracle Utilities Analytics data warehouse.

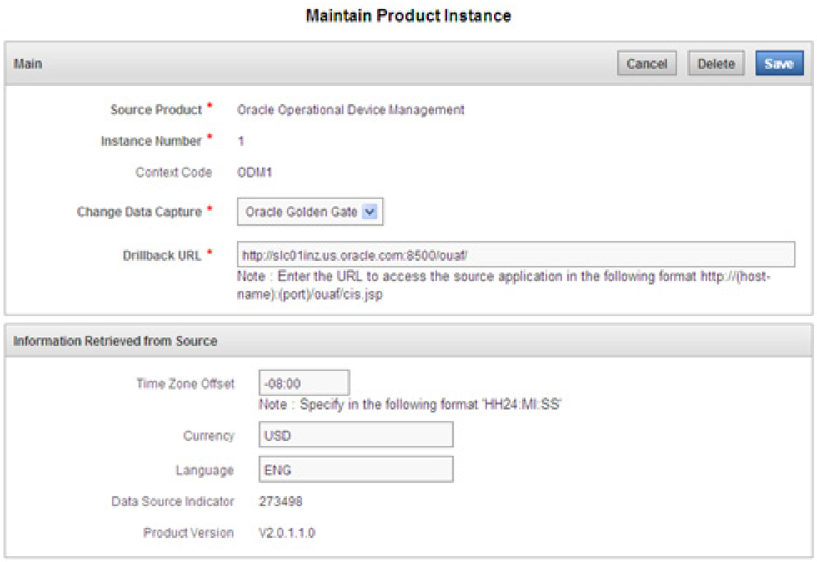
The following attributes are available on the Product Instance metadata table.

Attribute	Purpose
Product Code	A reference to the product code, identifying the product for this instance.
Instance Number	A unique number starting with 1 to uniquely identify the instance.
Context Code	A unique code comprised of the product code and instance number which is used to identify the connections in Oracle Data Integrator. Due to limitations imposed by Oracle GoldenGate (used for replication), the context code cannot be more than 5 characters in length.
Data Source Indicator	A unique value representing the instance. For Oracle Utilities Application Framework products, this is the environment ID of the source instance.
Journal Indicator	A flag identifying the methodology to be used for replicating source tables. The default is Oracle GoldenGate.
Drill back URL	A URL to be used to allow you to drill back to the source system from the analytics.

Attribute	Purpose
Currency code	The currency used in the source product instance. Multiple currencies are not supported.
Time Zone Code	The time zone of the source product instance. This allows the interfaces to be built so that dependencies on job execution for multiple time zones are handled correctly.
Language Code	The primary language supported by the data warehouse. This is used to filter language specific data in the data warehouse.
Source Instance Version	The current version number of the source product instance. This can be used to validate whether the version is supported by the data warehouse, or provide suggestions in case upgrades, or patches need to be applied to enable support for the source product.

The **Product Instance** page in the Administration Tool allows you to maintain the existing records. To add a new product instance, use the script delivered as part of the installation package.

For more details, refer to *Oracle Utilities Analytics Installation Guide* and check the **Configure Source** sub-section under **Post Installation Tasks of Oracle Data Integrator ELT Installation** section in the chapter 5.



Source Product *	Oracle Operational Device Management
Instance Number *	1
Context Code	ODM1
Change Data Capture *	Oracle Golden Gate
Drillback URL *	http://slc01inz.us.oracle.com:8500/ouaf/

Note : Enter the URL to access the source application in the following format http://(host-name):(port)/ouaf/cis.jsp

Time Zone Offset	-08:00
Currency	USD
Language	ENG
Data Source Indicator	273498
Product Version	V2.0.1.1.0

Note : Specify in the following format 'HH24:MI:SS'

Configuring the Target Entities

This table holds the configurations for target entities and their attributes in the data warehouse. The target entities can be dimensions, or facts. The same target entity may be loaded from multiple source instances. Configuration that is common across multiple instances is stored in this table.

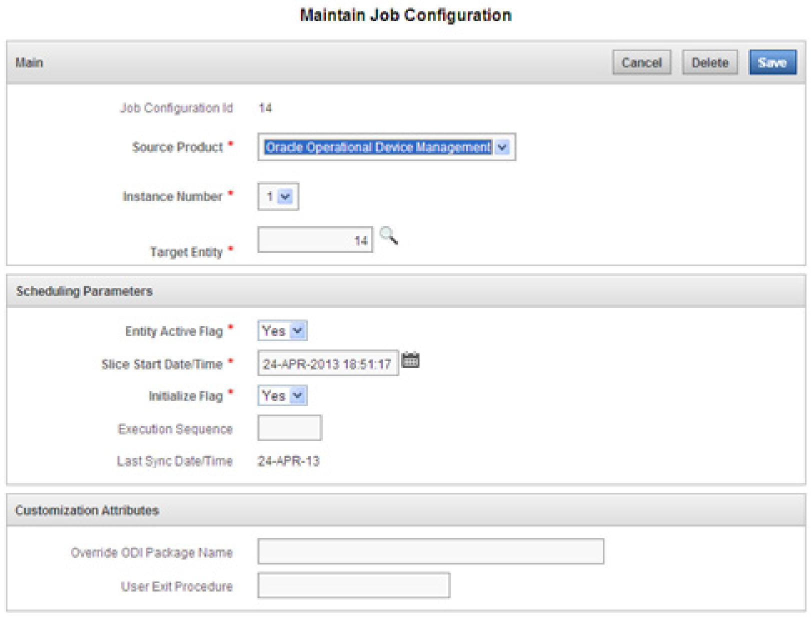
Attribute	Purpose
Name	The name of the entity, which needs to be scheduled for loading into the target.
Type	The type of the entity supported by Oracle Utilities Analytics: • Slowly changing dimension type 1 • Slowly changing dimension type 2 • Accumulation facts • Snapshot facts • Materialized views
Maximum Parallel Executions	To efficiently load data, it may be necessary to execute multiple instances, each instance working on a different data set. This attribute controls how many parallel executions can be spawned for a single entity load.
Retry Interval	The base architecture has been designed for automatic retries. In case of failures, jobs are retried and this attribute controls the interval between successive retries in case of a failure. Default is 30 minutes, but can be configured as per the requirement.
Maximum Retries per day	This attribute controls the maximum number of retry attempts in a day. Once this limit is reached and the load is still failing, it is retried next day.
Schedule Type	Three different modes of schedules are supported: • **Daily Incremental Load**: Loads are executed as soon as data is available for load. • **Near Real time Load**: Loads are executed within a configured interval. This is for future use. • **One time Load**: A load is executed only once.
Schedule Interval	This is applicable for NRT loads only and the value controls the frequency of execution of the load process.
Schedule Time	A job is executed only after the scheduled time each day. The default is 00:00. This can be changed as per the requirement.

Attribute	Purpose
Slice Duration Type	In any data warehouse, the basic challenge is to get the data loaded quickly and efficiently whether it is the initial load or the incremental load. Data volumes are high during initial load and during incremental loads, the volumes are considerably smaller. A slice is a volume of data bound within duration of time. Different objects have differing data distribution and load processing requirements. This attribute controls the duration between two slices. The following slicing intervals are supported: • Day(s) • Week(s) • Month(s) • Quarter(s) • Year(s) • Hour(s) • Minute(s)
Slice Duration	The number identifying the slice duration based on the slice duration type.
Package	The name of the Oracle Data Integrator scenario that should be executed. Refer to the **Appendix K** for a list of packages pre-configured for Oracle Data Integrator.
Staging Retention Period	The number of days to retain data in the staging tables.
Owner Flag	Indicates whether the record is owned by the base product (B1) or the customer (CM).

The **Target Entity** page in the Administration Tool allows you to add, edit, and delete records. Records are delivered out of the box for the star schemas tables delivered with Oracle Utilities Analytics. For these records, you can edit only select fields. You cannot delete base product owned records. Create new records for any entity that you want to add to the ELT. For owned records, you have full access to edit and delete them.

Configuring the Jobs

The **Job Configuration** page in the Administration Tool allows you to add, edit and delete records. As a part of the Oracle Utilities Analytics installation process, job configuration records are generated for the target entities available out of the box. You can then use this page to change the configurations.

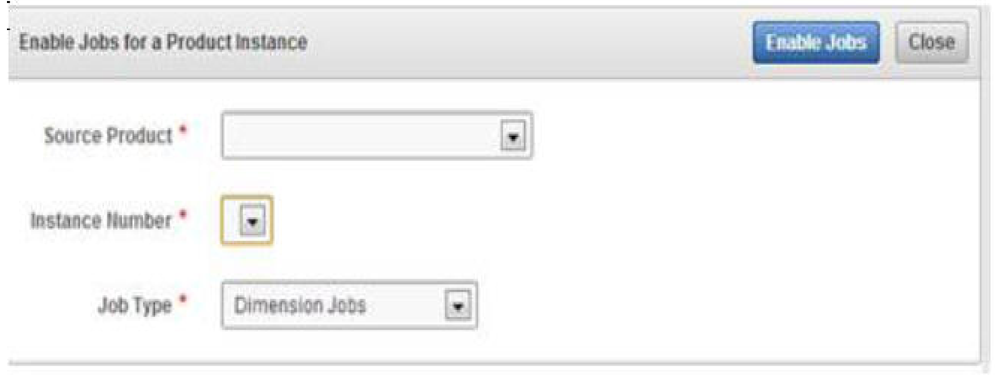


Job Configuration Id	14
Source Product *	Oracle Operational Device Management
Instance Number *	1
Target Entity *	14
Entity Active Flag *	Yes
Slice Start Date/Time *	24-APR-2013 18:51:17
Initialize Flag *	Yes
Execution Sequence	
Last Sync Date/Time	24-APR-13
Override ODI Package Name	
User Exit Procedure	

Enabling the Jobs

The **Enable Jobs** page can be accessed by clicking the button present on the **Job Configuration Report** page.

This page allows the end user to mass enable jobs by turning on the 'Entity Active Flag' for all those jobs that match the specified input criteria.



Source Product *	
Instance Number *	
Job Type *	Dimension Jobs

Configuring the Source Table

This table contains configurations controlling, such as the source tables to be replicated and the mode of replication.

Attribute	Purpose
Product Code	A reference to the product code, identifying the product for this instance.
Name	The name of the source table. There is a corresponding entry in the objects table also.
Mode	The following scenarios exist: • Source system tracks history using an effective date column • No history in source but needs history for the type 2 dimensions • No history required
Effective Date Column	The column name used for storing the effective dates in the source.
Base Replication	Controls whether the table is required to be replicated for target entity load.
Customize Replication	Extension for customizations. The additional tables to be marked for replication.
Purge Enabled	Controls whether the replicated table should be purged or not.
Retention Period	The number of days the data should be retained in the replication layer.
Owner Flag	Indicates whether the record is owned by the base product (B1) or the customer (CM).

The **Source Table** page in the Administration Tool allows you to add, edit, and delete records. For this table, records are delivered out of the box for the source tables that are configured to be part of the out-of- the- box ELT. For these records, you can edit only the selected fields. You cannot delete base product owned records. However, for owned records, you have full access to edit and delete them.

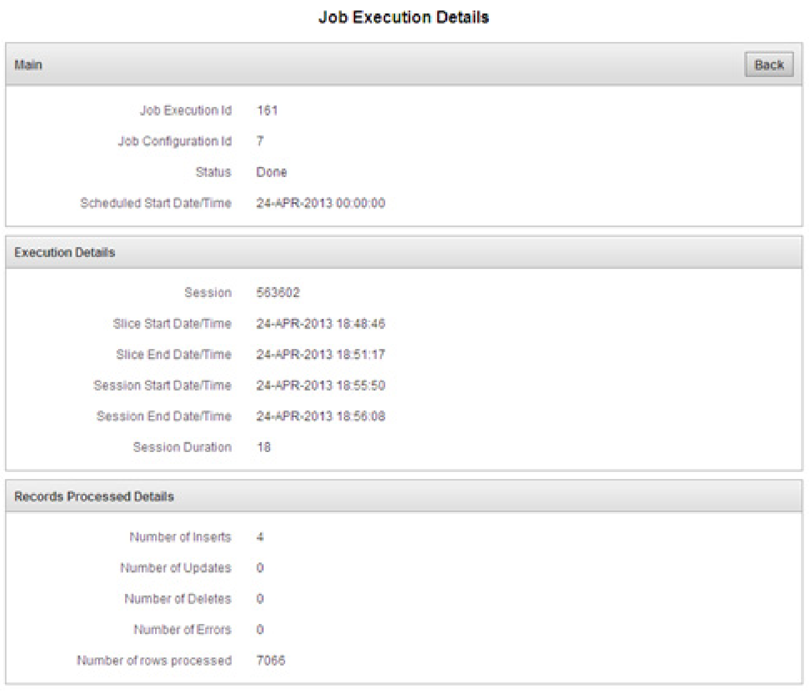
Executing a Job

This table is used for tracking the execution of the ELT processes. An entry is created for each job execution.

Attribute	Purpose
Job Configuration	A reference to the job configuration.
Session Number	Reference to the Oracle Data Integrator session number.
Scheduled Start Time	The start time as set in the schedule.
Slice Start	The starting timestamp of the slice.
Slice End	The ending timestamp of the slice.

Attribute	Purpose
Status	A composite status for the job. Primarily, the status is derived from SNP_SESSION. However, additional statuses are tracked in job executions: • Pending • Submitted • Running • Reprocessed • Error • Done
Actual Start time	The timestamp when the job actually started.
Actual End time	The timestamp when the job actually ended.
Duration	The total execution time in seconds.
Insert Count	The number of records inserted.
Update Count	The number of records updated.
Delete Count	The number of deletions.
Error Count	The number of rows identified as error.
Total Count	The number of rows processed. sum of the above.

Use the **Job Execution** page on the **Administration** Tool to monitor execution of the jobs. You cannot edit or create records in this table.



Job Execution Id 161
Job Configuration Id 7
Status Done
Scheduled Start Date/Time 24-APR-2013 00:00:00

Execution Details

Session 563602
Slice Start Date/Time 24-APR-2013 18:48:46
Slice End Date/Time 24-APR-2013 18:51:17
Session Start Date/Time 24-APR-2013 18:55:50
Session End Date/Time 24-APR-2013 18:56:08
Session Duration 18

Records Processed Details

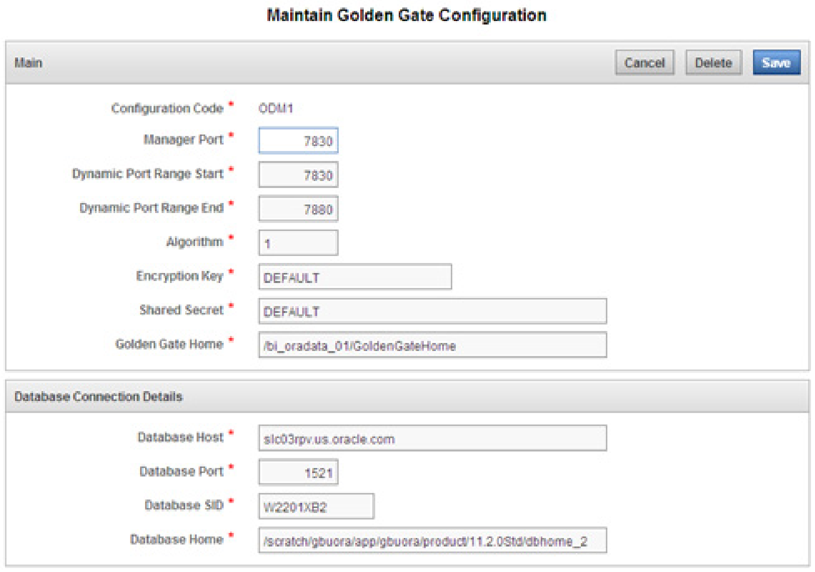
Number of Inserts 4
Number of Updates 0
Number of Deletes 0
Number of Errors 0
Number of rows processed 7066

Configuring Oracle GoldenGate

Oracle Utilities Analytics uses Oracle GoldenGate to capture the changed data. This table is used by the Oracle GoldenGate replication process to track its processing activities. This information is used by the scheduler process to ensure that the warehouse loading tasks only the processed data that has been synced.

Attribute	Purpose
Configuration code	The code is either be the context code identifying the instance uniquely or B1 for the target data warehouse.
Manager Port	The port that the Oracle GoldenGate Manager Port utilizes.
Dynamic Port Range	A port range that the Oracle GoldenGate process utilizes.
Algorithm	The encryption algorithm to be used. For example, BLOWFISH.
Encryption Key	The name of the key used for encryption/decryption.
Database Host	The host name of the database.
Database Port	The port name for the database.
Database ID	The system identifier for the database.
Database Home	The database home folder where the database software is installed.
GoldenGate Home	The home folder for Oracle GoldenGate.
Shared Secret	The Shared Secret is the shared public key. The Encryption mechanism utilizes a public key private key pair.

Oracle GoldenGate Configuration page in the Administration Tool allows you to add, edit, and delete records. As part of the Oracle Utilities Analytics installation process, Oracle GoldenGate configuration records are generated for the configured source application and the target Oracle Utilities Analytics application. You can then use this page to change the configurations.


Maintain Golden Gate Configuration
Main
Cancel
Delete
Save
Database Connection Details

Configuration Code *	ODM1
Manager Port *	7830
Dynamic Port Range Start *	7830
Dynamic Port Range End *	7880
Algorithm *	1
Encryption Key *	DEFAULT
Shared Secret *	DEFAULT
Golden Gate Home *	/bi_oradata_01/GoldenGateHome
Database Host *	slc03rpv.us.oracle.com
Database Port *	1521
Database SID *	W2201XB2
Database Home *	/scratch/gbuora/app/gbuora/product/11.2.0Std/dbhome_2

Configuring the Global Settings

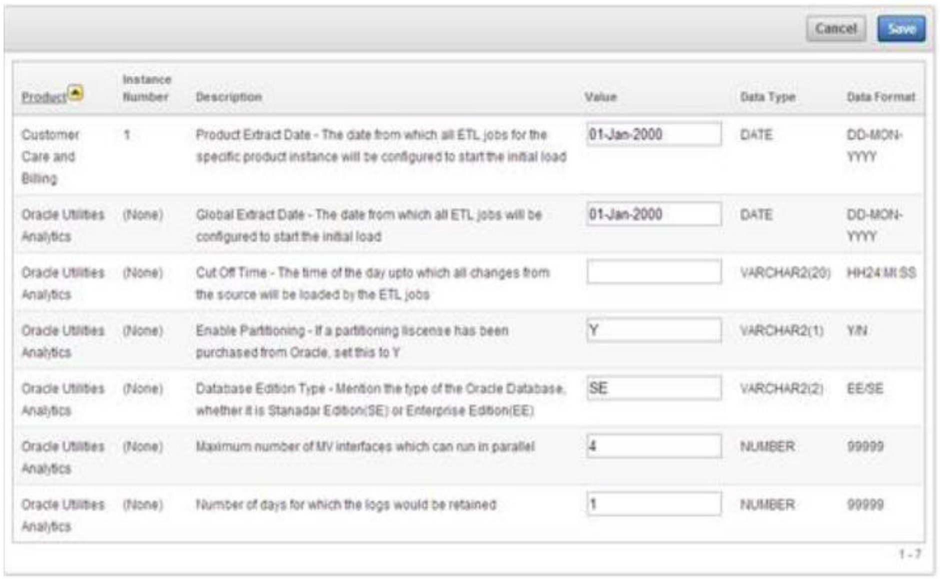
This table is used to capture the global settings needed for Oracle Utilities Analytics:

Attribute	**Purpose**
Product	A reference to the product code, identifying the product for this instance. This could be one of the configured source products of the base Oracle Utilities Analytics product itself.
Instance Number	The unique number of the configured source product instance. For the configuration settings of the base Oracle Utilities Analytics product, this value is set to null.
Description	This column describes the purpose of the specific global configuration setting.
Value	The configuration value that users can update based on their setup.
Data Type	The data type of the configuration value that a user is expected to enter.
Data Format	The specific format of the data that a user is expected to enter the configuration value in.

The **Global Configuration** page in the Administration Tool allows you to enter a configuration value for each of the configuration setting. As a part of the Oracle Utilities Analytics installation process, the global configuration records are generated for the configured source application and the target Oracle Utilities Analytics application. You can then use this page to change the configurations.

Users are expected to enter the configuration value strictly in the specified data format only. Entering value in other formats results in errors during the ETL processing.

Global Configuration



Product	Instance Number	Description	Value	Data Type	Data Format
Customer Care and Billing	1	Product Extract Date - The date from which all ETL jobs for the specific product instance will be configured to start the initial load	01-Jan-2000	DATE	DD-MON-YYYY
Oracle Utilities Analytics	(None)	Global Extract Date - The date from which all ETL jobs will be configured to start the initial load	01-Jan-2000	DATE	DD-MON-YYYY
Oracle Utilities Analytics	(None)	Cut Off Time - The time of the day upto which all changes from the source will be loaded by the ETL jobs		VARCHAR2(20)	HH24:MI:SS
Oracle Utilities Analytics	(None)	Enable Partitioning - If a partitioning liscense has been purchased from Oracle, set this to Y	Y	VARCHAR2(1)	Y/N
Oracle Utilities Analytics	(None)	Database Edition Type - Mention the type of the Oracle Database, whether it is Stanadar Edition(SE) or Enterprise Edition(EE)	SE	VARCHAR2(2)	EE/SE
Oracle Utilities Analytics	(None)	Maximum number of MV interfaces which can run in parallel	4	NUMBER	99999
Oracle Utilities Analytics	(None)	Number of days for which the logs would be retained	1	NUMBER	99999

1 - 7

Note: For application specific configuration details, see the respective *Oracle Utilities Schema and Extractors Data Mapping Guide*.

Appendix C

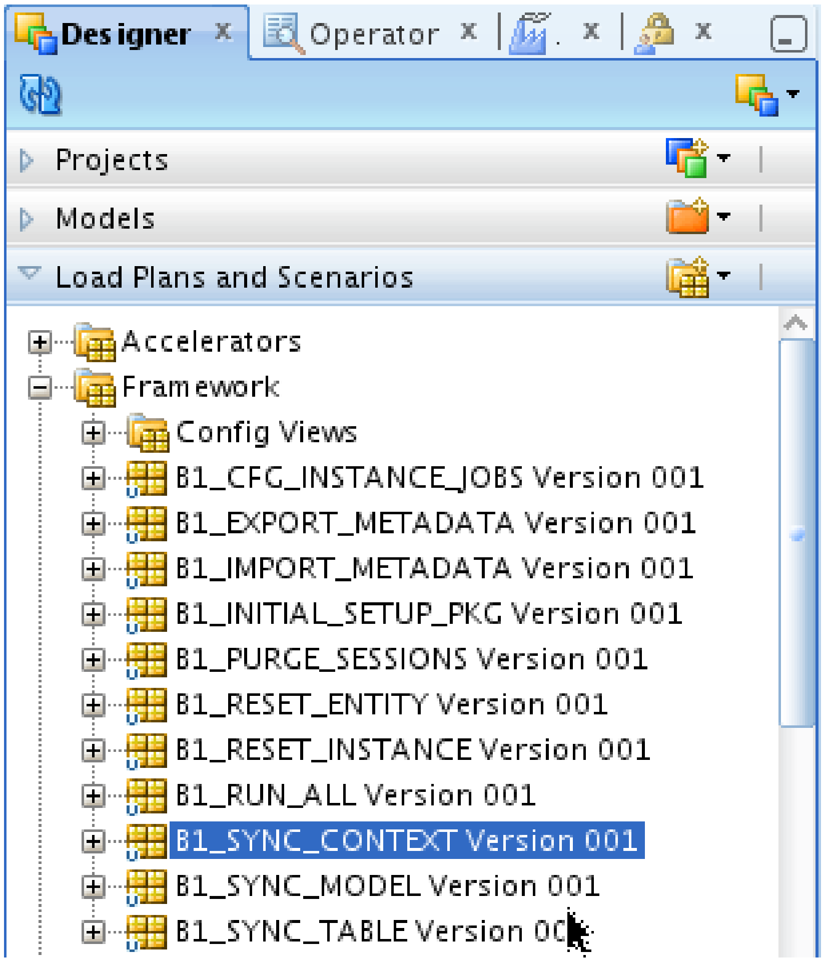
Oracle Utilities Analytics Initial Load

Setting up a data warehouse for the first time involves a large volume of data to be transferred from the source application to the target environment. In Oracle Data Integrator based architecture, Oracle GoldenGate is utilized to sync up a selective set of tables from the source system in a schema on the target. This schema is referred as the “Replication” schema.

The initial synchronization requires that the current snapshot of the tables at the source has to obtained and then applied to the replication schema. To perform this and utilize the database processing power efficiently, an Oracle Data Integrator scenario is provided. This process utilizes Oracle Database's Import/Export over the Database Link feature to perform the initial sync.

To reduce the time taken for this purpose, it is recommended to use a shared mount for export and import; otherwise use a file transfer mechanism. The database export data pump and import data pump require the directory to be created. The section below assumes the default directory **B1_DATA_PUMP_DIR** is being used for this process.

Create a database directory **B1_DATA_PUMP_DIR** on the target database and the location should be accessible to the database import/export process. This directory is used to store the data pump process log file.


Designer
Operator
Projects
Models
Load Plans and Scenarios
Accelerators
Framework
Config Views
B1_CFG_INSTANCE_JOBS Version 001
B1_EXPORT_METADATA Version 001
B1_IMPORT_METADATA Version 001
B1_INITIAL_SETUP_PKG Version 001
B1_PURGE_SESSIONS Version 001
B1_RESET_ENTITY Version 001
B1_RESET_INSTANCE Version 001
B1_RUN_ALL Version 001
B1_SYNC_CONTEXT Version 001
B1_SYNC_MODEL Version 001
B1_SYNC_TABLE Version 00

Execute the scenario **B1_SYNC_CONTEXT** while setting up the Oracle GoldenGate replication.

> **Note:** The incremental replication scripts for the source capture and the target apply process should be configured prior to executing this scenario. The sequence of execution is available in the **readme.txt** file generated along with the Oracle GoldenGate scripts.

A global configuration **B1_HIGH_VOL_THRESHOLD** is provided with a default value of 1000,000. This value controls the identification of tables as high volume or low volume. A high volume table processes in its own thread and this allows the high volume tables to be processed in parallel. This value is configurable and can be changed prior to execution of the scenario.

> **Note**: Executing this scenario in GLOBAL context performs the initial sync of all contexts assuming that the Oracle GoldenGate scripts have been deployed and replicate process is stopped for all the contexts.
>
> Executing this scenario for a specific context (ex CCB1) initiates the initial sync only for the models in the specified context
>
> If Oracle GoldenGate scripts have not been set up, the job gets complete, but does not sync any data. Ensure that there are records in the **B1_CHECKPOINT** table in the metadata schema with group name corresponding to each model created for the specific context.
>
> The status of extract and load for each table of the model can be checked by querying the table **B1_TABLE_SYNC** in the metadata schema post execution.

The status of the sync process for each context can be checked by querying the table **CDC_SYNC_LOG** in the replication schema. The table provides log of process including the start and end timestamps of the process.

Appendix D

Oracle Utilities Work and Asset Management Extractor Details

This section includes the details for the Oracle Utilities Work and Asset Management extractors used in Oracle Utilities Work and Asset Management (WAM) edge application:

Fact/Dimension Table Name	Batch Control Name	Source Table Name	Trigger Name	UDFs/UDMs being used
CD_ASSET	EXTDASSET	SA_ASSET	SDBT_BI_ADIU_ASSET	UDF1: Asset Class UDF2: Criticality UDF3: Building UDF4: Location UDF5: Process UDF6: Asset Record Type UDF7: Facility UDF8: Organization UDF9: Company
CD_CREW	EXTDWRKC	SA_CREW	SDBT_BI_ADIU_CREW	
CD_FAILURE	EXTDFAIL	SA_AUTHORITY	SDBT_BI_ADIU_FAILURE	
CD_OP_ACCT	EXTDOPAC	SA_ACCOUNT_DATA	SDBT_BI_ADIU_ACCT_DATA	UDF1: Area UDF2: Facility UDF3: Organization UDF4: Company UDF5: Level-1 Department UDF6: Level-2 Department UDF7: Level-3 Department
CD_OP_ACTG_TY	EXTDOATT	SA_AUTHORITY	SDBT_BI_ADIU_OP_ACTG_TR_TY	
CD_OP_EXP	EXTDOPEX			UDF1: Expense Category UDF2: Facility
CD_OP_UOM	EXTDOUOM	SA_AUTHORITY	SDBT_BI_ADIU_OUOM	

Fact/Dimension Table Name	Batch Control Name	Source Table Name	Trigger Name	UDFs/UDMs being used
CD_PLANNER	EXTDWRKP	SA_RULE_KEY	SDBT_BI_ADIU_RULE_KEY_PLAN	
CD_REPAIR	EXTDREPR	SA_AUTHORITY	SDBT_BI_ADIU_REPAIR	UDF1: Facility
CD_ROOT_CAUSE	EXTDROOT	SA_AUTHORITY	SDBT_BI_ADIU_ROOTCAUSE	
CD_STOCK_ITMTY	EXTDSITE	SA_STOREROOM_LOG	SDBT_BI_AI_STRM_LOG	UDF1: Stock Type UDF2: Stock Class UDF3: Commodity Category UDF4: Commodity Name UDF5 Commodity UDF6: Facility
CD_STRM	EXTDSTRM	SA_STOREROOM_SETUP	SDBT_BI_ADIU_STRM_SETUP	UDF1:Storeroom Type UDF2: Facility UDF3: Organization UDF4: Company
CD_STRM_TR_TY	EXTDSTTT	SA_STOREROOM_LOG	SDBT_BI_AI_STRM_LOG	
CD_WRKORD_TY	EXTDWOTY	SA_AUTHORITY	SDBT_BI_ADIU_WRK_ORD_TYPE	
CF_OP_ACTG	EXTFOPAT	SA_AUTHORITY	SDBT_BI_ADIU_OP_ACTG_TR_TY	
CF_STRM_INV	EXTFSTOR	SA_STOREROOM_SETUP	SDBT_BI_ADIU_STRM_SETUP	
CF_STRM_TR	EXTFSTTR	SA_STOREROOM_LOG	SDBT_BI_AI_STRM_LOG	
CF_WRKORD_TK	EXTFWRKT	SA_WORK_ORDER_TASK	SDBT_BI_WORK_ORDER_TASK	UDM1: Days Late UDM2: Days to close UDM3: Scheduled Downtime Indicator

Appendix E

Oracle Utilities Meter Data Management Extractor Details

This section contains details regarding each fact and dimension from Oracle Utilities Meter Data Management (MDM) edge application for synchronization BO-based extract and idiosyncratic batch extract, as shown below:

- **Synchronization BO-Based Extract Details**
- **Idiosyncratic Batch Extract Details**

Synchronization BO-Based Extract Details

Fact / Dimension Table Name	Initial Load Batch Control	Extract Batch Control	Sync BO	Extension
CF_DEVICE_EVT	D1-DEVIL	D1-DEVFX	D1- DeviceEventFact	Via Element Population Rules and Post Service Script on the Sync BO
CF_INSTALL_EVT	D1-INEIL	D1-INEFX	D1- InstallEventFact	Via Element Population Rules and Post Service Script on the Sync BO
CF_SP	D1-SPIL	D1-SPAFX	D1-SPAccumulationFact	Via Element Population Rules and Post Service Script on the Sync BO
CF_DEVICE_ACTIVITY	D1-ACTIL	D1-ACTFX	D1-ActivityAccumulationFact	Via Element Population Rules and Post Service Script on the Sync BO
CD_CONS_TYPE	D2-CSTIL	D2-CSTDX	D2-ConsumSnapshotTypeDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_MTR_DEVICE	D1-DVCIL	D1-DVCDX	D1-DeviceDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_MC	D1-MCIL	D1-MCDX	D1-MeasuringComponentDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_SPR	D1-SPRIL	D1-SPRDX	D1-ServiceProviderDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_SP	D1-SPIL	D1-SPDX	D1-SPDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_ADDR	D1-SPIL	D1-ADRDX	D1-AddressDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_US	D2-USIL	D2-USDX	D2-USDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_USAGE_GROUP	D2-UGIL	D2-UGDX	D2-UsageGroupDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_CONTACT	D2-CONIL	D2-CONDX	D2-ContactDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_MSRMT_COND	D2-MRCIL	D2-MRCDX	D2-MsrmtConditionDimension	Via Element Population Rules and Post Service Script on the Sync BO

Fact / Dimension Table Name	Initial Load Batch Control	Extract Batch Control	Sync BO	Extension
CD_IE_STATUS	D1-IESIL	D1-IESDX	D1-IEBOStatusAndReasnDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_SP_STATUS	D1-SPSIL	D1-SPSDX	D1-SPBOStatusAndReasnDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_DAYS_LAST_MSRMT	D1-LNMIL	D1-LNMDX	D1-DaysSinceLastNormalMsrmtDim	Via Element Population Rules and Post Service Script on the Sync BO
CD_EXCP_TYPE	D2-EXTIL	D2-EXTDX	D2-ExceptionTypeDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_VEE_RULE	D2-VERIL	D2-VERDX	D2-VEERuleDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_DEVICE_ACTIVITY_STATUS	D1-ACSIL	D1-ACSDX	D1-ActivityBOStatusAndReasnDim	Via Element Population Rules and Post Service Script on the Sync BO
CD_DEVICE_ACTIVITY_TYPE	D1-ATYIL	D1-ATYDX	D1-ActivityTypeDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_DEVICE_EVT_STATUS	D1-DESIL	D1-DESDX	D1-DEBOStatusAndReasnDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_DEVICE_EVT_TYPE	D1-DETIL	D1-DETDX	D1-DeviceEventTypeDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_SP_UT_AGE_TYPE	D2-UTAIL	D2-UTADX	D2-SPUTAgingTypeDimension	Via Element Population Rules and Post Service Script on the Sync BO
CD_DAYS_LASTUT_TYPE	D2-LUTIL	D2-LUTDX	D2-DaysSinceLastUTDimension	Via Element Population Rules and Post Service Script on the Sync BO

Idiosyncratic Batch Extract Details

Fact / Dimension Table Name	Initial Load Batch Control	Extract Batch Control	Sync BO	Extension
CF_CONSUMPTION	D2-SPCFX	Usage Snapshot	D2-SP-CA	Via new snapshot algorithm
CF_SP_SNAP	D1-SPSFX	Service Point Snapshot	D1-SPSNAP-SE	Via Post Service Script on the snapshot algorithm's parameter
CF_SP_UT_AGE	D2-SUAFX	Unreported Usage Analysis Snapshot	D2-SP-UT-AGE	Via new snapshot algorithm
CF_VEE_EXCP	D2-SVEFX	SP VEE Exception Snapshot	D2-SPVEEEXC	Via new snapshot algorithm
CD_UOM_TOU	D2-UTIL	n/a	n/a	Via Post Service Script on the batch control parameter
CD_UOM_TOU_SQI	D2-UTSIL	n/a	n/a	Via Post Service Script on the batch control parameter
CD_IMD_TYPE	D2-ITLIL	n/a	n/a	No extension
CD_EXCP_SEV	D2-EXLIL	n/a	n/a	No extension

Appendix F

Oracle Utilities Mobile Workforce Management Extractor Details

The chapter contains summary of the batch programs and sync Business Objects (BO) for each fact/dimension used in Oracle Utilities Mobile Workforce Management (MWM) edge application:

Fact /Dimension Table Name	Initial Load Batch Control	Extract Batch Control	Sync BO
CD_CREW_SHIFT	M1-SFTIL	M1-CRSDX	M1-CrewShiftDimension
CD_CREW_TM_USG	M1-CTUIL	M1-CTUDX	M1-CrewTimeUsageDimension
CD_APPT_TM	M1-APTIL	M1-APTDX	M1-AppointmentTimeDimension
CD_APPT_TM_OF_DAY	M1-ATDIL	M1-ATDDX	M1-AppointmentTmOfDayDimension
CD_TRAVEL_DIST_DEV	M1-TDDIL	M1-TADDX	M1-TravelDurDeviationDimension
CD_SERVICE_AREA	M1-SERIL	M1-SERDX	M1-ServiceAreaDimension
CD_CREW	M1-CREIL	M1-CREDX	M1-CrewDimension
CD_TASK_TYPE	M1-TKTIL	M1-TKTDX	M1-TaskTypeDimension
CD_ADDR	M1-LOCIL,M1-CSAIL,M1-TKAIL	M1-ADRDX	M1-AddressDimension
CD_SHIFT_BO_STATUS	M1-SBSIL	M1-SBSDX	M1-ShiftBoStatusResDimension
CD_TASK_BO_STATUS	M1-TBSIL	M1-TBSDX	M1-TaskBoStatusReasonDimension
CD_LATE_LOGON_TM	M1-LLTIL	M1-LLTDX	M1-LateLogonTimeDimension
CD_EARLY_LOGOFF	M1-ELTIL	M1-ELTDX	M1-EarlyLogoffTimeDimension
CD_TRAVEL_DUR_DEV	M1-TADIL	M1-TDDDX	M1-TravelDistDevDimension
CD_WORK_DUR_DEV	M1-WDDIL	M1-WDDDX	M1-WorkDurationDevtnDimension
CD_RESP_TM_DEV	M1-RTDIL	M1-RTDDX	M1-RespTimeDevDimension
CF_FLD_ACTIVITY	M1-ACTIL	M1-ACTFX	M1-ActivityFact
CF_CMP_SHIFT	M1-SFTIL	M1-CCSFX	M1-CompletedShiftFact
CF_CREW_TASK	M1-SFTIL	M1-CRTFX	M1-CrewTaskFact

Appendix G

NMS Management Reporting Using Oracle Business Intelligence Publisher

This section includes the following topics:

- **Defining a Report**
- **Running a Report**

Defining a Report

To generate a standard report in Oracle Business Intelligence Publisher, follow these steps:

1. Make sure the Oracle Business Intelligence Publisher Server has been started. If not yet started, go to the Oracle BI Publisher site and login.
2. Click the **Reports** tab.

   Note: Before you create a new report, you may want to create a new folder to contain the reports to be created. Click the **Create a new folder link** to create a new folder.
3. Click the **Create a new report** link.
4. Specify the report name. A new report icon with the specified name appears on the webpage.
5. Click the **Edit** link of the newly created report.
6. Add a **Description** of the report and define a **Default Data Source**.

   Note: Users with Administrator role access can add Data Source values. Make sure that the Oracle Business Intelligence Publisher administrator has already set up the correct Data Sources.

Setting up the Data Model

7. Click the **Data Model** link on the tree structure. A drop-down opens, displaying all the data sets that have been created.
8. Click the **New** icon to create a new Data Set to be used for the report.
9. Provide the following information:
   - **Name** – Select a value from the drop-down.
   - **Type** – Select the mode type, such as SQL Query, Data Template, Web Service, or XML Feed. Oracle Utilities Network Management reports are defined as Data Templates.

- Depending on the Type selected, more attributes need to be set up to define where the data will come from. For example, a Data Template data set includes the SQL query used to retrieve the columns and also the output XML structure.

Defining the List of Values

10. Click the **List of Values** link to define a List of Values. This is only necessary if you have a predefined set of values that need to be supplied to a report.

Defining Report Parameters

11. Click the **Parameters** link to define report parameters.
12. Click the **New** icon to create a new report parameter.
13. Provide the following information:
    - **Identifier** – This is the name of the parameter. Parameters are referenced in the SQL Query statements, Data Templates etc. through this Identifier.
    - **Data Type** – A parameter can only be a String, Integer, Boolean, Date or Float data type.
    - **Default Value** - The default value for this parameter.
    - **Parameter Type** – A parameter can be any of the following:
        - **Text** – Use this if you need a text box to input the parameter values.
        - **Menu** – Use this if there is pre-defined list of values that can be supplied to the report. A drop-down menu will be available to input values.
        - **Date** – Use this if the parameter is a date. A calendar will be available to input values. This ensures that only valid dates can be entered.
        - **Hidden** – Use this if the user does not need to see the actual value of the parameter.
    - **Display Label** - The label to display for this parameter.
    - Other attributes may need to be setup depending on the Parameter Type selected.

Defining Layouts

14. Click the **Layouts** link to define the templates that can be used on the report.
15. Click the **New** icon to create a new layout.
16. Provide the following information:
    - **Name** - Enter a name for the layout.
    - **Template** - Select a template from the drop-down list of available template files for this report.

      Note: To add a new template, click the Layouts link and upload templates from the directory folder using the Browse and Upload buttons.
    - **Template Type** - The example reports provided have corresponding RTF templates. However, there are other template types that available (*e.g.*, PDF, Excel, XML, etc.).
    - **Output Format** - The output format to use.

Saving the Changes

17. Click **Save** to commit your changes.

Once you have defined a new report, you can run the report to see the output.

Running a Report

To run a previously defined report in Oracle Business Intelligence Publisher, complete these steps:

1. Click the Reports tab. Go to the reports folder of the report you want to run.
2. Click the **View** link to run the report. The output is always based on the following:
   - **Parameters** – There is a section on the page where the user can input parameter values. It is always populated with the default values (if defined).
   - **Template** – There is a drop-down list of the available templates for the report. The report is always presented in the template currently selected by the user.

   Note: You can click the **View** button again at any time to rerun the report again.

When the page has completely loaded in the web browser, click the **Export** button to save the output. Depending on the selected template, a related application will be launched (For example, MS Word will be launched for an RTF template).

Note: For details about Oracle Utilities Network Management System specific reports, see *Oracle Utilities Extractors and Schema for Oracle Utilities Network Management System Data Mapping Guide.*

Appendix H

Oracle Utilities Validation Functions and Error Identification Procedure Names

This section lists the names for the validation functions and error identification procedure names for all the facts, along with the CM error procedure name and validation function name for those facts to add more validations:

Fact Name	Error Procedure Name	Validation Function Name	Custom Error Procedure Name	UDFs/UDMs being used
CF_CMP_SHIFT	B1_ERR_F_CMP_SHIFT	B1_VAL_F_CMP_SHIFT	CM_ERR_F_CMP_SHIFT	CM_VAL_F_CMP_SHIFT
CF_FLD_ACTIVITY	B1_ERR_F_FLD_ACTIVITY	B1_VAL_F_FLD_ACTIVITY	CM_ERR_F_FLD_ACTIVITY	CM_VAL_F_ARREARS
CF_INSTALL_EVT	B1_ERR_F_INSTALL_EVT	B1_VAL_F_INSTALL_EVT	CM_ERR_F_INSTALL_EVT	CM_VAL_F_INSTALL_EVT
CF_OP_ACTG	B1_ERR_F_OP_ACTG	B1_VAL_F_OP_ACTG	CM_ERR_F_OP_ACTG	CM_VAL_F_OP_ACTG
CF_ORDER	B1_ERR_F_ORDER	B1_VAL_F_ORDER	CM_ERR_F_ORDER	CM_VAL_F_ORDER
CF_PAY_TNDR	B1_ERR_F_PAY_TNDR	B1_VAL_F_PAY_TNDR	CM_ERR_F_PAY_TNDR	CM_VAL_F_PAY_TNDR
CF_SP	B1_ERR_F_SP	B1_VAL_F_SP	CM_ERR_F_SP	CM_VAL_F_SP
CF_SP_SNAP	B1_ERR_F_SP_SNAP	B1_VAL_F_SP_SNAP	CM_ERR_F_SP_SNAP	CM_VAL_F_SP_SNAP
CF_SP_UT_AGE	B1_ERR_F_SP_UT_AGE	B1_VAL_F_SP_UT_AGE	CM_ERR_F_SP_UT_AGE	CM_VAL_F_SP_UT_AGE
CF_VEE_EXCP	B1_ERR_F_VEE_EXCP	B1_VAL_F_VEE_EXCP	CM_ERR_F_VEE_EXCP	CM_VAL_F_VEE_EXCP
CF_WRKORD_TK	B1_ERR_F_WRKORD_TK	B1_VAL_F_WRKORD_TK	CM_ERR_F_WRKORD_TK	CM_VAL_F_WRKORD_TK

Appendix I

Oracle Warehouse Builder ETL Components

This section covers Oracle Warehouse Builder ETL components, including:

- **External Table**
- **Data Mapping**
- **Workflow**
- **Packages**
- **File Manager**

External Table

The extract programs execute in the source application. They produce flat files that contain the data extracted from the source system. Each process creates:

- A single-record control file that contains information about the entire batch job.
- The data files that contain the information to be loaded into the warehouse. These files are also referred to as the staging files.

Oracle external tables are defined in the warehouse for each type of control and data file. Specifically, two external tables are defined for each fact and dimension that is loaded from flat files. These external tables provide a SQL based interface to the data in the flat files by the data mappings. A data mapping exists for each fact and dimension.

Within Oracle database, the external tables have the following naming formats:

- STG_<table_name>_EXT for the data files.
- STG_<table_name>_CTL_EXT for the control files that are used to load a specific table.

For example, the external tables used to load the CD_ACCT table are named STG_ACCT_EXT and STG_ACCT_CTL_EXT.

The flat file names are different from the name of the external tables. The standard format for the file names are table_name_EXT.DAT and table_name_EXT.CTL. So for the CD_ACCT table, the files are named as D_ACCT_EXT.DAT and D_ACCT_EXT.CTL.

Note: Refer to the Data Mapping Guides for the respective source application for the list of file names for each fact and dimension.

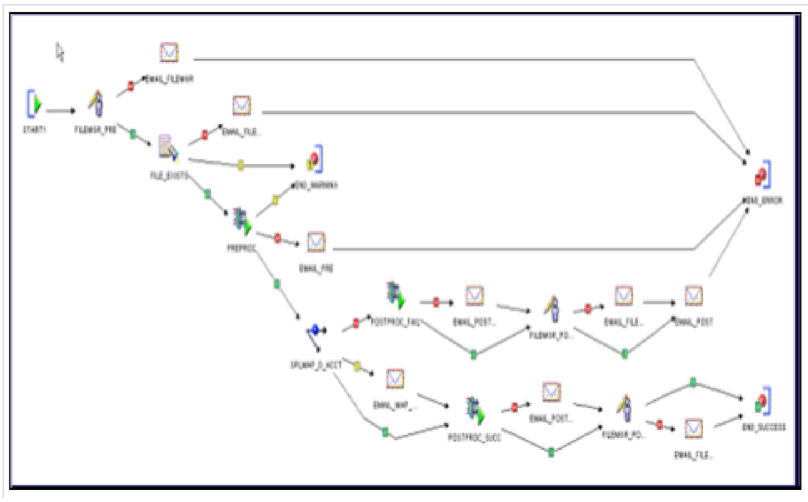
Data Mapping

The data mappings load data from the external tables (produced by the extracts) into the facts and dimensions in the warehouse.

Note: For a list of the facts and dimensions, their external tables, and the related data mappings, refer to the data mapping guide for your source application. This document describes the source application's facts and dimensions and how they are populated.

Workflow

A separate process flow exists to execute each mapping along with the pre-mapping and post-mapping processes. The following diagram shows a typical process flow:



Each data load process flow is designed to:

- Execute the file manager to perform housekeeping on the data and control files in pre-mapping and post-mapping modes.
- Execute the pre-mapping and post-mapping functions to validate, load, and maintain batch information in the ETL job control transaction.
- Execute the data mappings once the file is available and validated.
- Send an email if an error occurs. Also, if an error occurs before the mapping executes, the process flow aborts the complete process. Otherwise, it sends an email and continues. The process flow modules allow you to group process flow packages.

Packages

The process flow packages allow you to group process flows. Together, the process flow modules and packages provide two levels to manage and deploy process flows. You can validate, generate, and deploy process flows at either the module or the package level. All process flows are presently grouped under the following packages for easier administration:

- **INIT_PKG**: This package contains the process flows to load the default records into the dimensions. It also contains the process flows to load the date and time dimensions, and includes the purge workflow.
- **DIM**: This package contains the process flows for dimensions delivered in Oracle Utilities Analytics.
- **DIM_MDM**: This package contains the process flows for dimensions delivered in Oracle Utilities Analytics for all Oracle Utilities Meter Data Management dimension tables.
- **DIM_MWM**: This package contains process flows for dimensions delivered in Oracle Utilities Analytics for all Oracle Utilities Mobile Workforce Management dimension tables.

- **DIM_UDD**: This contains the process flows for all user-defined dimensions delivered in Oracle Utilities Analytics.
- **FACT**: This package contains the process flows to load all of the fact tables in Oracle Utilities Analytics.
- **FACT_MDM:** This package contains the process flows for facts delivered in Oracle Utilities Analytics for all the Oracle Utilities Meter Data Management fact tables.
- **FACT_MWM**: This package contains the process flows for facts delivered in Oracle Utilities Analytics for all the Oracle Utilities Mobile Workforce Management fact tables.
- **MV_RFSH**: This package contains the process flows to refresh the default materialized views created for each fact table. If custom materialized views are created, then a copy of the fact table process flow should be created and the new materialized view refresh added to the copied process flow. Note that the refresh of the materialized views is done in parallel.
- **LOADRFSH**: This package contains the process flows to load a fact table, and then refresh the materialized views for that fact table. A load refresh the process flow initiates the load for facts and subsequently executes the related materialized view refresh using the process flows under the package 'MV_RFSH'.

Note the following about the various the process flows:

- The process flows can be scheduled for execution using the File Processor daemon.
- The process flows for dimensions must be executed before the fact process flows.
- Each process flow executes its data mapping using parallel set-based processing with a commit frequency set to 0.

File Manager

The file manager is a PERL program that resides on the database server. The program is responsible for performing housekeeping activities against the files holding the extracted information. It also ensures that the files are supplied in the correct order.

The program accepts the following parameters:

- The name of the file that the external table reads the data from. This name should match the value of the flat file name without the file extension. So for the load of the CD_ACCT table, this is D_ACCT_EXT.
- File-Name parameter on the extract batch program.
- The location of the files.
- The program can be executed in pre-mapping and post-mapping modes.
- Processing condition (success or failure).
- In the pre-mapping mode, the file manager performs the following actions:
    - Creates “error” and “processed” files inside the folder where the files are located.
    - Sorts to get the name of the earliest control and data files that match the file name specified by the parameter passed.
    - Copies the data file and the control file to the files that the external table reads. This is required because the external tables are defined to read data from one particular file and the extractor programs insert the data source indicator, batch number, and batch thread number in the data and control file names to avoid overwriting the generated files.
    - Saves the name of the file being processed in a temporary file. This file is used later in the post-mapping stage to identify the name of the file that was processed. It is also used by the subsequent executions to know if a file is being processed.

In post-mapping mode, depending on the processing condition specified, the file manager moves the processed control and data file to either the error or the processed folder. It also removes the temporary file created in the pre-mapping mode.

Appendix J

Oracle Warehouse Builder Deployment

This section covers the details on how to generate and deploy the Oracle Warehouse Builder code for the ETL of new star schemas. It includes:

- **Oracle Warehouse Builder Code Generator**
- **Deploying the TCL Files**
- **Scheduling the New Workflows**

Oracle Warehouse Builder Code Generator

The Oracle Warehouse Builder (OWB) Code Generator is used to create **OMBPlus TCL** scripts. The **OMBPlus** is an Oracle Warehouse Builder scripting language that can create, alter, delete, and deploy Oracle Warehouse Builder objects. The **GenOWB.exe** program is located in the database package in the scripts folder.

The **GenOWB.exe** program must be run on a Windows machine that can connect to the data warehouse database. Use the following syntax to run the Oracle Warehouse Builder Code Generator:

The parameters and related values in the **GenOWB.exe** are described below:

```
GenOWB.exe -d <DBInfo> -t <TableName> -m <Mapping/Workflow
StagingTables/Facts/Dimension/All> -a <Dimensions/Facts/All> -h -g
```

The parameters and related values in the **GenOWB.exe** are described below:

Parameter	Value
-d	**Database Information**: Database User ID, password, database TNS name (for example, dwadm, dwadm, or bivmdv)
-t	Name of dimension or fact tables

Parameter	Value
-m	Generate the following: • Mapping (M) • WorkFlow (W) • Staging Tables (S) • Sequences (Q) • Facts (F) • Dimensions (D) • All (A)
-a	Generate mapping/Workflow/Staging Tables/Sequences/Cubes/Dimensions for all (D)imension Tables, all (F)act tables or (A)ll Dimension and Fact tables (D/F/A)
-x	**DROP statement**: (Y)es or (N)o. The default is No.
-h	Help
-g	generate debug info

When the Oracle Warehouse Builder (OWB) Code Generator is run for a table, the following files are generated:

- **seq_nam.TCL**: A file that creates the sequence in Oracle Warehouse Builder (OWB) used for the primary key of the table to be loaded. For example, the SPL_ACCT_SEQ.TCL file creates the sequence used to populate the primary key of the CD_ACCT table. Note that the sequence is also created manually in the database, as it is not recommended to deploy sequences from Oracle Warehouse Builder (OWB) to the database.
- **tbl_name.TCL**: A file that creates the table definition in Oracle Warehouse Builder. For example, the CD_ACCT.TCL script creates the CD_ACCT table in Oracle Warehouse Builder. Note that tables are also created manually in the database, as it is not recommended to deploy tables from Oracle Warehouse Builder to the database.
- **stg_file_name.TCL**: A file that creates the data file definition in Oracle Warehouse Builder. For example, the STG_ACCT_FF.TCL creates the definition of the CD_ACCT extract file.
- **ctl_file_name.TCL**: A file that creates the control file definition in Oracle Warehouse Builder. For example, the STG_ACCT_CTL_FF.TCL creates the definition of the CD_ACCT control file.
- **stg_tbl_name.TCL**: A file that creates the data file external table definition in Oracle Warehouse Builder. For example, the STG_ACCT_EXT.TCL creates the definition of the CD_ACCT external table STG_ACCT_EXT.
- **ctl_tbl_name.TCL**: A file that creates the control file external table definition in Oracle Warehouse Builder. For example, the STG_ACCT_CTL_EXT.TCL creates the definition of the CD_ACCT control table STG_ACCT_CTL_EXT.
- **owb_map_name.TCL**: A file that creates Oracle Warehouse Builder mapping, which loads the data from the external table into the data warehouse table. For example, the SPLMAP_D_ACCT.TCL script creates the SPLMAP_D_ACCT mapping, which reads records from the STG_ACCT_EXT and STG_ACCT_CTL_EXT files and loads the extracted records into the CD_ACCT table.

- **owb_wf_name.TCL**: A file that creates the process flow, which takes an extract file and loads it into the fact or dimension table. For example, the SPLWF_D_ACCT.TCL creates the SPLWF_D_ACCT process flow, which checks to see if an account extract file exists in the data load directory, and if it exists, then loads it into the CD_ACCT table.
- **OUBI_LDRF_wf_name.TCL**: A file that is created only for fact loads. It creates a process flow, which calls the data file loading process flow and the materialized view refresh process flow sequentially. For example, the OUBI_LDRF_FT.TCL file creates the OUBI_LDRF_FT process flow that calls the SPLWF_F_FT and OUBI_RFSH_FT process flows.

  Note: To create this process flow, the materialized view refresh process flow must exist.

Depending on which Oracle Warehouse Builder objects are changed, the parameters to the **GenOWB.exe** program is modified.

In the previous example, for the CF_CASE table change, the following two commands are run:

The first command shown above creates the SPLMAP_F_CASE.TCL file and the second command creates the SPLWF_F_CASE.TCL and OUBI_LDRF_CASE.TCL files, with drop commands in each file since the objects already exist in the Oracle Warehouse Builder repository.

```
GenOWB.exe -d spluser,spluser_pw,BICONN -t CF_CASE -m M -x Y
GenOWB.exe -d spluser,spluser_pw,BICONN -t CF_CASE -m W -x Y
```

The first command shown above creates the SPLMAP_F_CASE.TCL file and the second command creates the SPLWF_F_CASE.TCL and OUBI_LDRF_CASE.TCL files, with drop commands in each file since the objects already exist in the Oracle Warehouse Builder repository.

Deploying the TCL Files

Once the TCL scripts are created, they need to be loaded into the Oracle Warehouse Builder repository using OMBPlus. The OMBPlus is a flexible, high-level command line metadata access tool for Oracle Warehouse Builder. Use OMBPlus to create, modify, delete, and retrieve object metadata in Warehouse Builder design and runtime repositories.

Note: For more information about **OMBPlus**, refer to the Oracle Warehouse Builder API and Scripting Reference document.

To open an **OMBPlus** window, select **View->OMB*Plus** in the **OWB Design Center**.

From within the OMBPlus window, there are many OMBPlus commands available, but the following commands are usually used:

- cd SOURCE_DIRECTORY
- source TCL_FILE

  Note: The OMBPlus is case sensitive, so that the commands must be specified in lowercase. Also, the '\' is an escape character in OMBPlus, so within a directory name, two '\'s must be used if it is needed.

To load the files that were created by using the commands in the previous section, assume that the TCL files are in the c:\bi\tcl directory. The following **OMBPlus** commands can be used to load the files:

```
cd c:\\bi\\tcl
source SPLMAP_F_CASE.TCL
source SPLWF_F_CASE.TCL
source OUBI_LDRF_CASE.TCL
```

The order that the TCL files are loaded is important, as the objects are dependent on other objects. If an object is deleted from Oracle Warehouse Builder by running a TCL file, then references to that object are dropped from already existing Oracle Warehouse Builder objects. So in all the cases, the order that the TCL files are listed in the preceding section is always used. Also, if an earlier object is recreated, all of the other objects that are listed afterwards also need to be created.

For example, if a change needs to be made to an Oracle Warehouse Builder mapping, the owb_map_name.TCL, owb_wf_name.TCL and OUBI_LDRF_wf_name.TCL (for facts) scripts are regenerated and reloaded into OMBPlus.

After loading some of these TCL scripts, the customizations that are done prior to earlier deployments are lost, so the preconditions to deployment must be redone. The following is a list of the scripts that you must rerun after changes are made:

- **EditFFCS.tcl**: This is needed to be run if the flat file TCL scripts (stg_file_name.TCL and ctl_file_name.TCL) are loaded.
- **EditFFLL.tcl**: This is needed to be run if the external table TCL scripts (stg_tbl_name.TCL and ctl_tbl_name.TCL) are loaded.
- **EDITFP.tcl and editmail.tcl**: These scripts is needed to be run if the process flow TCL scripts (owb_wf_name.TCL) are loaded.

Scheduling the New Workflows

When new extracts are set up on the source application side and new Oracle Warehouse Builder process flows have been setup to load the new flat files, the File Processor daemon needs to be extended to allow the processing of the new flat files.

More specifically the parameters file needs to be extended to include the new mappings. In the new CM parameters file, these following two types of mapping need to be present:

- extract.file.mapping.override.count
- extract.file.mapping

> **Note**: For more details on the File Processor daemon and the above mentioned parameters, refer to **Chapter 4** under the section "**Scheduling Jobs Using the File Processor Daemon**".

Oracle Warehouse Builder object **deploymentOwbdepolyment.sh** (Owbdepolyment.cmd on Windows platform) performs the following tasks on the Unix platform:

- Imports the MDL files
- Configures Control Center
- Registers locations
- Runs TCL scripts to configure
- Deploys Oracle Warehouse Builder objects in following order:
  - Connectors
  - External Tables
  - Sequences
  - Dimensions
  - Transformations
  - Mappings
  - Workflows

Appendix K

Oracle Data Integrator ELT Components

Oracle Date Integrator ELT components include packages. The following table lists out the entities populated using Oracle Date Integrator, and the Oracle Date Integrator package that is executed to perform the data load from replication to the target entity:

Product	Entity	Package name
CCB	B1_ARREARS_MON_MV1	B1_PKG_B1_ARREARS_MON_MV1
CCB	B1_ARREARS_MON_TOPX_MV1	B1_PKG_B1_ARREARS_MON_TOPX_MV1
CCB	B1_BILLEDUSAGE_MON_MV1	B1_PKG_B1_BILLEDUSAGE_MON_MV1
CCB	B1_BILLEDUSAGE_MON_TOPX_MV1	B1_PKG_B1_BILLEDUSAGE_MON_TOPX_MV1
CCB	B1_CASELOG_MON_MV1	B1_PKG_B1_CASELOG_MON_MV1
CCB	B1_CASE_MON_MV1	B1_PKG_B1_CASE_MON_MV1
CCB	B1_CASE_MON_MV2	B1_PKG_B1_CASE_MON_MV2
CCB	B1_CASE_TOPX_MON_MV1	B1_PKG_B1_CASE_TOPX_MON_MV1
CCB	B1_CASE_TOPX_MON_MV2	B1_PKG_B1_CASE_TOPX_MON_MV2
CCB	B1_CC_HOU_MV1	B1_PKG_B1_CC_HOU_MV1
CCB	B1_CC_MON_MV1	B1_PKG_B1_CC_MON_MV1
CCB	B1_CC_TOPX_MON_MV1	B1_PKG_B1_CC_TOPX_MON_MV1
CCB	B1_COLLPROC_MON_MV1	B1_PKG_B1_COLLPROC_MON_MV1
CCB	B1_COLLPROC_MON_MV2	B1_PKG_B1_COLLPROC_MON_MV2
CCB	B1_COLLPROC_MON_TOPX_MV1	B1_PKG_B1_COLLPROC_MON_TOPX_MV1
CCB	B1_COLLPROC_MON_TOPX_MV2	B1_PKG_B1_COLLPROC_MON_TOPX_MV2
CCB	B1_FT_MON_MV1	B1_PKG_B1_FT_MON_MV1
CCB	B1_FT_MON_TOPX_MV1	B1_PKG_B1_FT_MON_TOPX_MV1
CCB	B1_PAY_PLAN_MON_MV1	B1_PKG_B1_PAY_PLAN_MON_MV1
CCB	B1_PAY_PLAN_SNAP_MON_MV1	B1_PKG_B1_PAY_PLAN_SNAP_MON_MV1

Product	Entity	Package name
CCB	B1_PAY_PLAN_SNP_MON_TOPX_MV1	B1_PKG_B1_PAY_PLAN_SNP_MON_TOPX_MV1
CCB	B1_PA_MV1	B1_PKG_B1_PA_MV1
CCB	B1_PA_SNAP_MON_MV1	B1_PKG_B1_PA_SNAP_MON_MV1
CCB	B1_PA_SNAP_MON_TOPX_MV1	B1_PKG_B1_PA_SNAP_MON_TOPX_MV1
CCB	B1_SA_BILLING_MON_MV1	B1_PKG_B1_SA_BILLING_MON_MV1
CCB	B1_SA_BILLING_MON_TOPX_MV1	B1_PKG_B1_SA_BILLING_MON_TOPX_MV1
CCB	B1_SA_MON_MV1	B1_PKG_B1_SA_MON_MV1
CCB	B1_SA_MON_MV2	B1_PKG_B1_SA_MON_MV2
CCB	B1_SA_TOPX_MON_MV1	B1_PKG_B1_SA_TOPX_MON_MV1
CCB	B1_SA_TOPX_MON_MV2	B1_PKG_B1_SA_TOPX_MON_MV2
CCB	B1_TD_ENTRY_DOW_MV1	B1_PKG_B1_TD_ENTRY_DOW_MV1
CCB	B1_TD_ENTRY_DOW_MV2	B1_PKG_B1_TD_ENTRY_DOW_MV2
CCB	B1_TD_ENTRY_MON_TOPX_MV1	B1_PKG_B1_TD_ENTRY_MON_TOPX_MV1
CCB	B1_UCOLLPROC_MON_MV1	B1_PKG_B1_UCOLLPROC_MON_MV1
CCB	B1_UCOLLPROC_MON_MV2	B1_PKG_B1_UCOLLPROC_MON_MV2
CCB	B1_UCOLLPROC_MON_TOPX_MV1	B1_PKG_B1_UCOLLPROC_MON_TOPX_MV1
CCB	B1_UCOLLPROC_MON_TOPX_MV2	B1_PKG_B1_UCOLLPROC_MON_TOPX_MV2
CCB	CD_ACCT	B1_PKG_CD_ACCT
CCB	CD_ADDR	B1_PKG_CD_ADDR
CCB	CD_ADJ_TYPE	B1_PKG_CD_ADJ_TYPE
CCB	CD_BILL_CAN_RSN	B1_PKG_CD_BILL_CAN_RSN
CCB	CD_BILL_CYC_SCH	B1_PKG_CD_BILL_CYC_SCH
CCB	CD_BILL_DAY_IN_WIN	B1_PKG_CD_BILL_DAY_IN_WIN
CCB	CD_BSEG_STATUS	B1_PKG_CD_BSEG_STATUS
CCB	CD_CAMPAIGN	B1_PKG_CD_CAMPAIGN
CCB	CD_CASETYPE_STATUS	B1_PKG_CD_CASETYPE_STATUS
CCB	CD_CASE_COND	B1_PKG_CD_CASE_COND
CCB	CD_CC_TYPE	B1_PKG_CD_CC_TYPE
CCB	CD_COLLEVT_TYPE	B1_PKG_CD_COLLEVT_TYPE
CCB	CD_COLLPROC_STATUS	B1_PKG_CD_COLLPROC_STATUS
CCB	CD_COLLPROC_TMPL	B1_PKG_CD_COLLPROC_TMPL
CCB	CD_DAYS_LAST_FRZ_BS	B1_PKG_CD_DAYS_LAST_FRZ_BS

Product	Entity	Package name
CCB	CD_DAYS_TO_WIN_CLS	B1_PKG_CD_DAYS_TO_WIN_CLS
CCB	CD_DAYS_UNBILLED_USG	B1_PKG_CD_DAYS_UNBILLED_USG
CCB	CD_FISCAL_CAL	B1_PKG_CD_FISCAL_CAL
CCB	CD_FT_TYPE	B1_PKG_CD_FT_TYPE
CCB	CD_GL_ACCT	B1_PKG_CD_GL_ACCT
CCB	CD_INSTALLMENT_CNT	B1_PKG_CD_INSTALLMENT_CNT
CCB	CD_MSG	B1_PKG_CD_MSG
CCB	CD_MSRMT_TYPE	B1_PKG_CD_MSRMT_TYPE
CCB	CD_ORDER_CAN_RSN	B1_PKG_CD_ORDER_CAN_RSN
CCB	CD_ORDER_STATUS	B1_PKG_CD_ORDER_STATUS
CCB	CD_PAY_CAN_RSN	B1_PKG_CD_PAY_CAN_RSN
CCB	CD_PAY_METHOD	B1_PKG_CD_PAY_METHOD
CCB	CD_PAY_PLAN_STATUS	B1_PKG_CD_PAY_PLAN_STATUS
CCB	CD_PAY_PLAN_TYPE	B1_PKG_CD_PAY_PLAN_TYPE
CCB	CD_PA_STATUS	B1_PKG_CD_PA_STATUS
CCB	CD_PER	B1_PKG_CD_PER
CCB	CD_PKG	B1_PKG_CD_PKG
CCB	CD_PREM	B1_PKG_CD_PREM
CCB	CD_RATE	B1_PKG_CD_RATE
CCB	CD_REC_CHARGE_AMOUNT	B1_PKG_CD_REC_CHARGE_AMOUNT
CCB	CD_SA	B1_PKG_CD_SA
CCB	CD_SA_STATUS	B1_PKG_CD_SA_STATUS
CCB	CD_SQI	B1_PKG_CD_SQI
CCB	CD_TD	B1_PKG_CD_TD
CCB	CD_TD_PRIORITY	B1_PKG_CD_TD_PRIORITY
CCB	CD_TD_ROLE	B1_PKG_CD_TD_ROLE
CCB	CD_TD_SKILL	B1_PKG_CD_TD_SKILL
CCB	CD_TD_STATUS	B1_PKG_CD_TD_STATUS
CCB	CD_TD_TYPE	B1_PKG_CD_TD_TYPE
CCB	CD_TNDR_SRCE	B1_PKG_CD_TNDR_SRCE
CCB	CD_TNDR_STATUS	B1_PKG_CD_TNDR_STATUS
CCB	CD_TNDR_TYPE	B1_PKG_CD_TNDR_TYPE
CCB	CD_TOU	B1_PKG_CD_TOU

Product	Entity	Package name
CCB	CD_UCOLEVT_TYPE	B1_PKG_CD_UCOLEVT_TYPE
CCB	CD_UCOLPROC_STATUS	B1_PKG_CD_UCOLPROC_STATUS
CCB	CD_UCOLPROC_TMPL	B1_PKG_CD_UCOLPROC_TMPL
CCB	CD_UOM	B1_PKG_CD_UOM
CCB	CD_USER	B1_PKG_CD_USER
CCB	CF_ARREARS	B1_PKG_CF_ARREARS
CCB	CF_BILLED_USAGE	B1_PKG_CF_BILLED_USAGE
CCB	CF_CASE	B1_PKG_CF_CASE
CCB	CF_CASE_LOG	B1_PKG_CF_CASE_LOG
CCB	CF_CC	B1_PKG_CF_CC
CCB	CF_COLL_EVT	B1_PKG_CF_COLL_EVT
CCB	CF_COLL_PROC	B1_PKG_CF_COLL_PROC
CCB	CF_FT	B1_PKG_CF_FT
CCB	CF_FT_GL	B1_PKG_CF_FT_GL
CCB	CF_ORDER	B1_PKG_CF_ORDER
CCB	CF_PA	B1_PKG_CF_PA
CCB	CF_PAY_PLAN	B1_PKG_CF_PAY_PLAN
CCB	CF_PAY_PLAN_SNAP	B1_PKG_CF_PAY_PLAN_SNAP
CCB	CF_PAY_TNDR	B1_PKG_CF_PAY_TNDR
CCB	CF_PA_SNAP	B1_PKG_CF_PA_SNAP
CCB	CF_RECENT_TD_ENTRY	B1_PKG_CF_RECENT_TD_ENTRY
CCB	CF_SA	B1_PKG_CF_SA
CCB	CF_SA_BILLING	B1_PKG_CF_SA_BILLING
CCB	CF_TD_ENTRY	B1_PKG_CF_TD_ENTRY
CCB	CF_UCOL_EVT	B1_PKG_CF_UCOL_EVT
CCB	CF_UCOL_PROC	B1_PKG_CF_UCOL_PROC
ODM	B1_ASSET_LOC_MON_MV1	B1_PKG_B1_ASSET_LOC_MON_MV1
ODM	B1_OPR_DEVICE_MON_MV1	B1_PKG_B1_OPR_DEVICE_MON_MV1
ODM	B1_OPR_DEVICE_MON_TOPX_MV1	B1_PKG_B1_OPR_DEVICE_MON_TOPX_MV1
ODM	B1_OPR_DEVICE_SNAP_MON_MV1	B1_PKG_B1_OPR_DEVICE_SNAP_MON_MV1
ODM	B1_OPR_DEVICE_SNP_MON_TOPX_MV1	B1_PKG_B1_OPR_DEVICE_SNP_MON_TOPX_MV1

Product	Entity	Package name
ODM	B1_SERVICE_HIST_MON_MV1	B1_PKG_B1_SERVICE_HIST_MON_MV1
ODM	CD_ADDR	B1_PKG_CD_ADDR
ODM	CD_ASSET_AGE	B1_PKG_CD_ASSET_AGE
ODM	CD_ASSET_DISP	B1_PKG_CD_ASSET_DISP
ODM	CD_ASSET_INSP_STATUS	B1_PKG_CD_ASSET_INSP_STATUS
ODM	CD_ASSET_INSTALL_AGE	B1_PKG_CD_ASSET_INSTALL_AGE
ODM	CD_ASSET_INSTORE_AGE	B1_PKG_CD_ASSET_INSTORE_AGE
ODM	CD_LOCATION	B1_PKG_CD_LOCATION
ODM	CD_OPR_DEVICE	B1_PKG_CD_OPR_DEVICE
ODM	CD_SERVICE_HIST_TYPE	B1_PKG_CD_SERVICE_HIST_TYPE
ODM	CD_UTIL_ASSET	B1_PKG_CD_UTIL_ASSET
ODM	CF_ASSET_LOC	B1_PKG_CF_ASSET_LOC
ODM	CF_OPR_DEVICE	B1_PKG_CF_OPR_DEVICE
ODM	CF_OPR_DEVICE_SNAP	B1_PKG_CF_OPR_DEVICE_SNAP
ODM	CF_SERVICE_HIST	B1_PKG_CF_SERVICE_HIST
NMS	CD_CALL_INFO	B1_PKG_CD_CALL_INFO
NMS	CD_CITY	B1_PKG_CD_CITY
NMS	CD_DURATION_DEVIATION	B1_PKG_CD_DURATION_DEVIATION
NMS	CD_EST_RST_DUR	B1_PKG_CD_EST_RST_DUR
NMS	CD_EVENT	B1_PKG_CD_EVENT
NMS	CD_EVENT_STATUS	B1_PKG_CD_EVENT_STATUS
NMS	CD_OUTG_DUR	B1_PKG_CD_OUTG_DUR
NMS	CD_STORM	B1_PKG_CD_STORM
NMS	CD_STORM_OUTAGE_TYPE	B1_PKG_CD_STORM_OUTAGE_TYPE
NMS	CD_SW_PLAN	B1_PKG_CD_SW_PLAN
NMS	CD_SW_PLAN_STATE	B1_PKG_CD_SW_PLAN_STATE
NMS	CD_ACCT	B1_PKG_CD_ACCT
NMS	CD_ADDR	B1_PKG_CD_ADDR
NMS	CD_CREW	B1_PKG_CD_CREW
NMS	CD_CTRL_ZONE	B1_PKG_CD_CTRL_ZONE
NMS	CD_DEVICE	B1_PKG_CD_DEVICE
NMS	CD_FEEDER	B1_PKG_CD_FEEDER
NMS	CD_METER	B1_PKG_CD_METER

Product	Entity	Package name
NMS	CD_PER	B1_PKG_CD_PER
NMS	CD_PHASE	B1_PKG_CD_PHASE
NMS	CD_PREM	B1_PKG_CD_PREM
NMS	CD_SNL	B1_PKG_CD_SNL
NMS	CD_USER	B1_PKG_CD_USER
NMS	CF_CUST_RECENT_OUTG	B1_PKG_CF_CUST_RECENT_OUTG
NMS	CF_CUST_RST_OUTG	B1_PKG_CF_CUST_RST_OUTG
NMS	CF_RECENT_CALL	B1_PKG_CF_RECENT_CALL
NMS	CF_RECENT_CREW	B1_PKG_CF_RECENT_CREW
NMS	CF_RECENT_JOB	B1_PKG_CF_RECENT_JOB
NMS	CF_RST_CALL	B1_PKG_CF_RST_CALL
NMS	CF_RST_CREW	B1_PKG_CF_RST_CREW
NMS	CF_RST_JOB	B1_PKG_CF_RST_JOB
NMS	CF_SW_PLAN	B1_PKG_CF_SW_PLAN
NMS	CF_SW_PLAN_STATE	B1_PKG_CF_SW_PLAN_STATE
NMS	CF_CITY_OUTG	B1_PKG_CF_CITY_OUTG
NMS	CF_CTRL_ZONE_OUTG	B1_PKG_CF_CTRL_ZONE_OUTG
NMS	CF_FEEDER_DLVRD_LOAD	B1_PKG_CF_FEEDER_DLVRD_LOAD
NMS	CF_OUTG	B1_PKG_CF_OUTG
NMS	B1_CITY_OUTG_MON_MV1	B1_PKG_B1_CITY_OUTG_MON_MV1
NMS	B1_CTRL_ZONE_OUTG_MON_MV1	B1_PKG_B1_CTRL_ZONE_OUTG_MON_MV1
NMS	B1_CTRL_ZONE_OUTG_MON_MV2	B1_PKG_B1_CTRL_ZONE_OUTG_MON_MV2
NMS	B1_CUST_RST_OUTG_MON_MV1	B1_PKG_B1_CUST_RST_OUTG_MON_MV1
NMS	B1_CUST_RST_OUTG_MON_MV2	B1_PKG_B1_CUST_RST_OUTG_MON_MV2
NMS	B1_FEEDER_DLVRD_MON_MV1	B1_PKG_B1_FEEDER_DLVRD_MON_MV1
NMS	B1_OUTG_MON_MV1	B1_PKG_B1_OUTG_MON_MV1
NMS	B1_RST_CALL_MON_MV1	B1_PKG_B1_RST_CALL_MON_MV1
NMS	B1_RST_CALL_MON_MV2	B1_PKG_B1_RST_CALL_MON_MV2
NMS	B1_RST_CALL_MON_MV3	B1_PKG_B1_RST_CALL_MON_MV3
NMS	B1_RST_CREW_MON_MV1	B1_PKG_B1_RST_CREW_MON_MV1
NMS	B1_RST_JOB_MON_MV1	B1_PKG_B1_RST_JOB_MON_MV1
NMS	B1_RST_JOB_MON_MV2	B1_PKG_B1_RST_JOB_MON_MV2
NMS	B1_RST_JOB_MON_MV3	B1_PKG_B1_RST_JOB_MON_MV3

Product	Entity	Package name
NMS	B1_RST_JOB_TOPX_MON_MV1	B1_PKG_B1_RST_JOB_TOPX_MON_MV1

Appendix L

Terminology

The following Oracle Utilities Analytics terminologies are explained in detail in this chapter:

- **Data Source Indicator**
- **Late Arriving Dimension**
- **Data Source Indicator**
- **Physical Schema**
- **Model**
- **Interface**
- **Package**
- **Agent**
- **Reverse Engineering**
- **Journalization**
- **Change Data Capture**
- **Scenario**
- **Knowledge Module**

Data Source Indicator

The data warehouse is an integrated database receiving data from the multiple sources applications. The Data Source Indicator (DSI) in Oracle Utilities Analytics is used as an identifier for the originating source application or system-of-record for the data. Both fact and dimension tables have a DSI value for each record.

In the edge applications integrated with Oracle Utilities Analytics, namely Oracle Utilities Customer Care and Billing, Oracle Utilities Meter Date Management, Oracle Utilities Mobile Workforce Management, Oracle Utilities Network Management System, and Oracle Utilities Work and Asset Management, the DSI value is defaulted from environment ID. In case of multiple instances of the same application loading the data in the data warehouse, such as two separate instances of Customer Care and Billing (CC&B) application, both the instances have different DSI value.

The DSI value is also used for data lookup when loading the fact data. The confirmed dimension like address may have data populated from multiple source applications, such as Oracle Utilities Customer Care and Billing and Oracle Utilities Meter Date Management, or may have data populated from multiple instances of same source application, such as two instances of Oracle Utilities Meter Date Management. It is also possible that two records, one from Oracle Utilities

Meter Date Management, and another from Oracle Utilities Customer Care and Billing (or second instance of Oracle Utilities Meter Date Management), may have the same ID in source application. Therefore, while loading the fact record, the DSI value is also included in dimension record lookup columns along with the source ID column(s) so that the fact record from a given source application is correctly linked to the dimension record from the same instance of the source application.

Late Arriving Dimension

Late arriving dimensions are the dimensions where the fact (measurable quantities) table records come early when compared to the dimension table records. An auto correction mechanism is implemented into the fact load process to handle such scenarios.

To support auto correction of late arriving dimensions, there are two default rows in each dimension.

Primary Key	Default Values	Data Source Indicator	Remark
0	***	NULL	This is used to refer to the Null value references in the source. Possibly valid reasons for the references to be Null.
-99	-N/A	NULL	This is be used to tag any unidentifiable references. The source data has a value for a reference, but corresponding record does not exist in the target. Possibly caused by late arriving dimensions.

Referencing Foreign Keys

The foreign key value is set to 0 for any rows with Null values for "Ref Natural Key". The foreign key value is set to -99 for any rows with some value for "Ref Natural Key", but no corresponding record in the dimension. Error reprocessing automatically binds these rows with the correct rows in the dimension, assuming that the dimension value has arrived.

Oracle Data Integrator

Context

A context is a collection of logical schemas mapped to physical schemas. All logical schemas may not be mapped to physical schemas in a context. The source, journal and replication schemas are not associated for the default Global context as this context is not associated with any source instance.

A context identifies the physical schemas and connections that are utilized for data access. Interfaces can be executed in any context as this allows reusing the same code for multiple instances.

The contexts are created while a source instance is configured for integration with Oracle Utilities Analytics. The context code is a composite string comprising of the product code and the instance number. This combination allows us to create uniquely identifiable context codes associated with independent source systems.

Logical Schema

A logical schema is a logical representation of a storage location. The same physical schemas can be associated with different logical schemas. Here are some examples of schemas:

- **Source**: This is associated with the source instance.
- **Journal**: The journal schema is utilized for non-GoldenGate based replication. This represents the work schema where objects can be created for processing.
- **Replication**: This represents the data area where replicated data is stored. Each source system is associated with a replication schema on the target database.
- **Target**: All star schema objects are stored in this area.
- **Metadata**: The metadata configurations utilized to manage the solution are stored in this data area.
- **Master**: The Oracle Data Integrator master repository is maintained in a separate schema associated with the master logical schema.
- **Repository**: This schema is used to store the work repository of Oracle Data Integrator.
- **Configuration**: The configuration schema is associated with the file store location used for export or import of the metadata.

Physical Schema

A physical schema is the actual schema or database user where data is retrieved or stored.

Model

A model is an Oracle Data Integrator entity, which stores the list of objects, their structures, constraints and relationships. The model also allows creation of validation rules that can be applied on the data.

The model folders in the solution are organized based on the logical schemas, which also form the different stages of data in the data processing pipeline.

The source models are not part of the base Oracle Data Integrator work repository that is installed in a fresh install. As each source instance is added the associated source model(s) are created and actions executed on it. Each source instance may have one or more models associated. The model codes are by suffixing two character alphabets to the context code.

Interface

An interface is a declarative mapping between the source to target, and it utilizes Loading Knowledge Modules (LKM), Integration Knowledge Module (IKM) and Check Knowledge Module (CKM) as required for execution. Joins, Aggregations, filters and data transformations are performed on an interface.

Package

A Package is made up of a sequence of steps organized into an execution diagram.

Agent

An agent is a component that executes the Oracle Data Integrator jobs. Multiple agents can be created and utilized for load balancing. The solution uses WebLogic enterprise agents for stability and scalability.

Reverse Engineering

It is the process of identifying the data structures, constraints and relations from an existing data store. The data store can be any database management system, files or any other Oracle Data Integrator supported data store.

Journalization

It is the process of keeping track of data changes enabling incremental data access, transport and transformation.

Change Data Capture

It is the process of identifying and tracking data that has changed since the last known state. Change data capture mechanisms are implemented by the journalizing knowledge module.

Scenario

A scenario is a self contained prepackaged executable version of a package, interface or procedure in Oracle Data Integrator. The logic within the scenario cannot be viewed using the Oracle Data Integrator client.

Knowledge Module

Knowledge modules (KMs) in Oracle Data Integrator are components that implement reusable transformation and ELT (Extract, Load, and Transform) strategies across different technologies.

Knowledge Modules (KMs) are code templates. Each KM is dedicated to an individual task in the overall data integration process.

These are based on logical tasks that are performed and do not contain references to physical objects (data stores, columns, physical paths, etc.).

A KM is reused across several interfaces or models. The benefit of knowledge modules is that you make a change once and it is instantly propagated to hundreds of transformations.

These can be written with a mix and match of different programming languages, types, and styles (native RDBMS SQL, scripting languages such as Jython or JavaScript, or even Java).

Types of knowledge modules are as follows:

- Reverse-engineering Knowledge Module (RKM)
- Check Knowledge Module (CKM)
- Loading Knowledge Module (LKM)
- Integration Knowledge Module (IKM)
- Journalizing Knowledge Module (JKM)
- Service Knowledge Module (SKM)

Note: Refer to Oracle Data Integrator documents for more details.