Oracle® Fusion Middleware

Connectivity and Knowledge Modules Guide for Oracle Data Integrator

11*g* Release 1 (11.1.1)

E12644-03

October 2010

ORACLE

Oracle Fusion Middleware Connectivity and Knowledge Modules Guide for Oracle Data Integrator, 11*g* Release 1 (11.1.1)

E12644-03

Primary Author: Laura Hofman Miquel

Contents

24 LDAP Directories

25 Oracle Changed Data Capture Adapters

B Oracle Data Integrator Driver for XML Reference

Preface

This book describes how work with different technologies in Oracle Data Integrator.

This preface contains the following topics:

Audience

This document is intended for developers who want to work with Knowledge Modules for their integration processes in Oracle Data Integrator.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following Oracle resources:

- *Oracle Fusion Middleware Getting Started with Oracle Data Integrator*
- *Oracle Fusion Middleware Installation Guide for Oracle Data Integrator*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Data Integrator*
- *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*
- *Oracle Fusion Middleware Knowledge Module Developer's Guide for Oracle Data Integrator*
- *Oracle Fusion Middleware Application Adapters Guide for Oracle Data Integrator*
- *Oracle Data Integrator 11g Online Help*
- *Oracle Data Integrator 11g Release Notes, included with your Oracle Data Integrator 11g installation and on Oracle Technology Network*

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.

1 Introduction

This book describes how work with different technologies in Oracle Data Integrator. This book contains the following parts:

- Part I, "Databases, Files, and XML"
- Part II, "Business Intelligence"
- Part III, "Other Technologies"

Application Adapters are covered in a separate guide. See the *Oracle Fusion Middleware Application Adapters Guide for Oracle Data Integrator* for more information.

This chapter provides an introduction to the terminology used in the Oracle Data Integrator documentation and describes the basic steps of how to use Knowledge Modules in Oracle Data Integrator.

This chapter contains the following sections:

1.1 Terminology

This section defines some common terms that are used in this document and throughout the related documents mentioned in the Preface.

Knowledge Module

Knowledge Modules (KMs) are components of Oracle Data Integrator that are used to generate the code to perform specific actions against certain technologies.

Combined with a connectivity layer such as, for example, JDBC, JMS, or JCA, Knowledge Modules allow running defined tasks against a technology, such as connecting to this technology, extracting data from it, transforming the data, checking it, integrating it, etc.

Application Adapter

Oracle Application Adapters for Data Integration provide specific software components for integrating enterprise applications data. Enterprise applications suported by Oracle Data Integrator include Oracle E-Business Suite, Siebel, SAP, etc.

An *adapter* is a group of Knowledge Modules. In some cases, this group also contains an attached technology definition for Oracle Data Integrator.

Application Adapters are covered in a separate guide. See the *Oracle Fusion Middleware Application Adapters Guide for Oracle Data Integrator* for more information.

1.2 Using This Guide

This guide provides conceptual information and processes for working with knowledge modules and technologies supported in Oracle Data Integrator.

Each chapter explains how to configure a given technology, set up a project and use the technology-specific knowledge modules to perform integration operations.

Some knowledge modules are not technology-specific and require a technology that support an industry standard. These knowledge modules are referred to as *Generic* knowledge modules. For example the knowledge modules listed in Chapter 4, "Generic SQL" and in Chapter 22, "JMS" are designed to work respectively with any ANSI SQL-92 compliant database and any JMS compliant message provider.

When these generic knowledge module can be used with a technology, the technology chapter will mention it. However, we recommend using technology-specific knowledge modules for better performances and enhanced technology-specific feature coverage.

Before using a knowledge module, it is recommended to review the knowledge module description in Oracle Data Integrator Studio for usage details, limitations and requirements. In addition, although knowledge modules options are pre-configured with default values to work out of the box, it is also recommended to review these options and their description.

The chapters in this guide will provide you with the important usage, options, limitation and requirement information attached to the technologies and knowledge modules.

Part I

Databases, Files, and XML

This part describes how to work with databases, files, and XML files in Oracle Data Integrator.

Part I contains the following chapters:

2

Oracle Database

This chapter describes how to work with Oracle Database in Oracle Data Integrator.

This chapter includes the following sections:

2.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in an Oracle Database. All Oracle Data Integrator features are designed to work best with the Oracle Database engine, including reverse-engineering, changed data capture, data quality, and integration interfaces.

2.1.1 Concepts

The Oracle Database concepts map the Oracle Data Integrator concepts as follows: An Oracle Instance corresponds to a data server in Oracle Data Integrator. Within this instance, a schema maps to an Oracle Data Integrator physical schema. A set of related objects within one schema corresponds to a data model, and each table, view or synonym will appear as an ODI datastore, with its attributes, columns and constraints.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to Oracle database instance.

2.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 2–1 for handling Oracle data. The KMs use Oracle specific features. It is also possible to use the generic SQL KMs with the Oracle Database. See Chapter 4, "Generic SQL" for more information.

Table 2–1 Oracle Database Knowledge Modules

Knowledge Module	Description
RKM Oracle	Reverse-engineers tables, views, columns, primary keys, non unique indexes and foreign keys.
JKM Oracle 10g Consistent (Streams)	Creates the journalizing infrastructure for consistent set journalizing on Oracle 10g tables, using Oracle Streams.
JKM Oracle 11g Consistent (Streams)	Creates the journalizing infrastructure for consistent set journalizing on Oracle 11g tables, using Oracle Streams.
JKM Oracle Consistent	Creates the journalizing infrastructure for consistent set journalizing on Oracle tables using triggers.
JKM Oracle Consistent (Update Date)	Creates the journalizing infrastructure for consistent set journalizing on Oracle tables using triggers based on a *Last Update Date* column on the source tables.
JKM Oracle Simple	Creates the journalizing infrastructure for simple journalizing on Oracle tables using triggers.
JKM Oracle to Oracle Consistent (OGG)	Creates and manages the ODI CDC framework infrastructure when using Oracle GoldenGate for CDC. See Chapter 26, "Oracle GoldenGate" for more information.
CKM Oracle	Checks data integrity against constraints defined on an Oracle table.
LKM File to Oracle (EXTERNAL TABLE)	Loads data from a file to an Oracle staging area using the EXTERNAL TABLE SQL Command.
LKM File to Oracle (SQLLDR)	Loads data from a file to an Oracle staging area using the SQL*Loader command line utility.
LKM MSSQL to Oracle (BCP SQLLDR)	Loads data from a Microsoft SQL Server to Oracle database (staging area) using the BCP and SQL*Loader utilities.
LKM Oracle BI to Oracle (DBLINK)	Loads data from any Oracle BI physical layer to an Oracle target database using database links. See Chapter 17, "Oracle Business Intelligence Enterprise Edition" for more information.
LKM Oracle to Oracle (DBLINK)	Loads data from an Oracle source database to an Oracle staging area database using database links.
LKM Oracle to Oracle (datapump)	Loads data from an Oracle source database to an Oracle staging area database using external tables in the datapump format.
LKM SQL to Oracle	Loads data from any ANSI SQL-92 source database to an Oracle staging area.
LKM SAP BW to Oracle (SQLLDR)	Loads data from SAP BW systems to an Oracle staging using SQL*Loader utilities. See the *Oracle Fusion Middleware Application Adapters Guide for Oracle Data Integrator* for more information.
LKM SAP ERP to Oracle (SQLLDR)	Loads data from SAP ERP systems to an Oracle staging using SQL*Loader utilities. See the *Oracle Fusion Middleware Application Adapters Guide for Oracle Data Integrator* for more information.
IKM Oracle AW Incremental Update	Integrates data in an Oracle target table in incremental update mode and is able to refresh a Cube in an Analytical Workspace. See Chapter 21, "Oracle OLAP" for more information.
IKM Oracle Incremental Update	Integrates data in an Oracle target table in incremental update mode.
IKM Oracle Incremental Update (MERGE)	Integrates data in an Oracle target table in incremental update mode, using a MERGE statement.
IKM Oracle Incremental Update (PL SQL)	Integrates data in an Oracle target table in incremental update mode using PL/SQL.

Table 2–1 (Cont.) Oracle Database Knowledge Modules

Knowledge Module	Description
IKM Oracle Multi Table Insert	Integrates data from one source into one or many Oracle target tables in append mode, using a multi-table insert statement (MTI).
IKM Oracle Slowly Changing Dimension	Integrates data in an Oracle target table used as a Type II Slowly Changing Dimension.
IKM Oracle Spatial Incremental Update	Integrates data into an Oracle (9i or above) target table in incremental update mode using the MERGE DML statement. This module supports the SDO_GEOMETRY datatype.
IKM Oracle to Oracle Control Append (DBLINK)	Integrates data from one Oracle instance into an Oracle target table on another Oracle instance in control append mode. This IKM is typically used for ETL configurations: source and target tables are on different Oracle instances and the interface's staging area is set to the logical schema of the source tables or a third schema.
SKM Oracle	Generates data access Web services for Oracle databases. See "Working with Data Services" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for information about how to use this SKM.

2.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Oracle Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

2.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

2.2.2 Technology Specific Requirements

Some of the Knowledge Modules for Oracle use specific features of this database. This section lists the requirements related to these features.

2.2.2.1 Using the SQL*Loader Utility

This section describes the requirements that must be met before using the SQL*Loader utility with Oracle database.

- The Oracle Client and the SQL*Loader utility must be installed on the machine running the Oracle Data Integrator Agent.
- The server names defined in the Topology must match the Oracle TNS name used to access the Oracle instances.

- A specific log file is created by SQL*Loader. We recommend looking at this file in case of error. Control Files (CTL), Log files (LOG), Discard Files (DSC) and Bad files (BAD) are placed in the work directory defined in the physical schema of the source files.
- Using the DIRECT mode requires that Oracle Data integrator Agent run on the target Oracle server machine. The source file must also be on that machine.

2.2.2.2 Using External Tables

This section describes the requirements that must be met before using external tables in Oracle database.

- The file to be loaded by the External Table command needs to be accessible from the Oracle instance. This file must be located on the file system of the server machine or reachable from a Unique Naming Convention path (UNC path) or stored locally.
- For performance reasons, it is recommended to install the Oracle Data Integrator Agent on the target server machine.

2.2.2.3 Using Oracle Streams

This section describes the requirements for using Oracle Streams Journalizing knowledge modules.

> **Note:** It is recommended to review first the "Changed Data Capture" chapter in the *Oracle Database Data Warehousing Guide*, which contains the comprehensive list of requirements for Oracle Streams.

The following requirements must be met before setting up changed data capture using Oracle Streams:

- Oracle Streams must be installed on the Oracle Database.
- The Oracle database must run using a SPFILE (only required for AUTO_CONFIGURATION option).
- The AQ_TM_PROCESSES option must be either left to the default value, or set to a value different from 0 and 10.
- The COMPATIBLE option should be set to 10.1 or higher.
- The database must run in ARCHIVELOG mode.
- PARALLEL_MAX_SERVERS must be increased in order to take into count the number of Apply and Capture processes. It should be increased at least by 6 for Standalone configuration, 9 for Low-Activity and 21 for High-Activity.
- UNDO_RETENTION must be set to 3600 at least.
- STREAMS_POOL_SIZE must be increased by 100MB for Standalone configuration, 236MB for Low-Activity and 548MB for High-Activity.
- All the columns of the primary key defined in the ODI Model must be part of a SUPPLEMENTAL LOG GROUP.
- When using the AUTO_CONFIGURATION knowledge module option, all the above requirements are checked and set-up automatically, except some actions that must be set manually. See "Using the Streams JKMs" for more information.

In order to run this KM without AUTO_CONFIGURATION knowledge module option, the following system privileges must be granted:

- DBA role to the connection user
- Streams Administrator to the connection user
- RESOURCE role to the work schema
- SELECT ANY TABLE to the work schema

- Asynchronous mode gives the best performance on the journalized system, but this requires extra Oracle Database initialization configuration and additional privileges for configuration.
- Asynchronous mode requires the journalized database to be in ARCHIVELOG. Before turning this option on, you should first understand the concept of asynchronous AutoLog publishing. See the Oracle Database Administrator's Guide for information about running a database in ARCHIVELOG mode. See "Asynchronous Change Data Capture" in the *Oracle Database Data Warehousing Guide* for more information on supplemental logging. This will help you to correctly manage the archives and avoid common issues, such as hanging the Oracle instance if the archive files are not removed regularly from the archive repository.
- When using asynchronous mode, the user connecting to the instance must be granted admin authorization on Oracle Streams. This is done using the DMBS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE procedure when logged in with a user already having this privilege (for example the SYSTEM user).
- The work schema must be granted the SELECT ANY TABLE privilege to be able to create views referring to tables stored in other schemas.

For detailed information on all other prerequisites, see the "Change Data Capture" chapter in the *Oracle Database Data Warehousing Guide*.

2.2.3 Connectivity Requirements

This section lists the requirements for connecting to an Oracle Database.

JDBC Driver

Oracle Data Integrator is installed with a default version of the Oracle Type 4 JDBC driver. This drivers directly uses the TCP/IP network layer and requires no other installed component or configuration.

It is possible to connect an Oracle Server through the Oracle JDBC OCI Driver, or even using ODBC. For performance reasons, it is recommended to use the Type 4 driver.

Connection Information

You must ask the Oracle DBA the following information:

- Network Name or IP address of the machine hosting the Oracle Database.
- Listening port of the Oracle listener.
- Name of the Oracle Instance (SID).
- TNS alias of the connected instance.
- Login and password of an Oracle User.

2.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating an Oracle Data Server
2. Creating an Oracle Physical Schema

2.3.1 Creating an Oracle Data Server

An Oracle data server corresponds to an Oracle Database Instance connected with a specific Oracle user account. This user will have access to several schemas in this instance, corresponding to the physical schemas in Oracle Data Integrator created under the data server.

2.3.1.1 Creation of the Data Server

Create a data server for the Oracle technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining an Oracle data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator.
   - **Instance/dblink**: TNS Alias used for this Oracle instance. It will be used to identify the Oracle instance when using database links and SQL*Loader.
   - **User/Password**: Oracle user (with its password), having select privileges on the source schemas, select/insert privileges on the target schemas and select/insert/object creation privileges on the work schemas that will be indicated in the Oracle physical schemas created under this data server.
2. In the JDBC tab:
   - **JDBC Driver**: `oracle.jdbc.driver.OracleDriver`
   - **JDBC URL**: `jdbc:oracle:thin:@<network name or ip address of the Oracle machine>:<port of the Oracle listener (1521)>:<name of the Oracle instance>`

     To connect an Oracle RAC instance with the Oracle JDBC thin driver, use an Oracle RAC database URL as shown in the following example:

     ```
     jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)
     (ADDRESS=(PROTOCOL=TCP)(HOST=host1) (PORT=1521))
     (ADDRESS=(PROTOCOL=TCP)(HOST=host2) (PORT=1521))
     (CONNECT_DATA=(SERVICE_NAME=service)))
     ```

2.3.2 Creating an Oracle Physical Schema

Create an Oracle physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

2.4 Setting Up an Integration Project

Setting up a project using the Oracle Database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with Oracle Database:

- RKM Oracle
- CKM Oracle
- LKM SQL to Oracle
- LKM File to Oracle (SQLLDR)
- LKM File to Oracle (EXTERNAL TABLE)
- IKM Oracle Incremental Update

2.5 Creating and Reverse-Engineering an Oracle Model

This section contains the following topics:

2.5.1 Create an Oracle Model

Create an Oracle Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

2.5.2 Reverse-engineer an Oracle Model

Oracle supports both Standard reverse-engineering - which uses only the abilities of the JDBC driver - and Customized reverse-engineering, which uses a RKM to retrieve the structure of the objects directly from the Oracle dictionary.

In most of the cases, consider using the standard JDBC reverse engineering for starting. Standard reverse-engineering with Oracle retrieves tables, views, columns, primary keys, and references.

Consider switching to customized reverse-engineering for retrieving more metadata. Oracle customized reverse-engineering retrieves the table and view structures, including columns, primary keys, alternate keys, indexes, check constraints, synonyms, and references.

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on Oracle use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on Oracle with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Oracle technology:

In the Reverse tab of the Oracle Model, select the `KM: RKM Oracle.<project name>`.

2.6 Setting up Changed Data Capture

The ODI Oracle Knowledge Modules support the Changed Data Capture feature. See Chapter "Working with Changed Data Capture" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details on how to set up journalizing and how to use captured changes.

Oracle Journalizing Knowledge Modules support Simple Journalizing and Consistent Set Journalizing. The Oracle JKMs use either triggers or Oracle Streams to capture data changes on the source tables.

Oracle Data Integrator provides the Knowledge Modules listed in Table 2–2 for journalizing Oracle tables.

Table 2–2 Oracle Journalizing Knowledge Modules

KM	Notes
JKM Oracle 10g Consistent (Streams)	Creates the journalizing infrastructure for consistent set journalizing on Oracle 10g tables, using Oracle Streams.
JKM Oracle 11g Consistent (Streams)	Creates the journalizing infrastructure for consistent set journalizing on Oracle 11g tables, using Oracle Streams.
JKM Oracle Consistent	Creates the journalizing infrastructure for consistent set journalizing on Oracle tables using triggers.
JKM Oracle Consistent (Update Date)	Creates the journalizing infrastructure for consistent set journalizing on Oracle tables using triggers based on a *Last Update Date* column on the source tables.
JKM Oracle Simple	Creates the journalizing infrastructure for simple journalizing on Oracle tables using triggers.

Note that it is also possible to use Oracle GoldenGate to consume changed records from an Oracle database. See Chapter 26, "Oracle GoldenGate" for more information.

Using the Streams JKMs

The Streams KMs work with the default values. The following are the recommended settings:

- By default, the AUTO_CONFIGURATION KM option is set to Yes. If set to Yes, the KM provides automatic configuration of the Oracle database and ensures that all prerequisites are met. As this option automatically changes the database initialization parameters, it is not recommended to use it in a production environment. You should check the Create Journal step in the Oracle Data Integrator execution log to detect configurations tasks that have not been performed correctly (Warning status).
- By default, the CONFIGURATION_TYPE option is set to `Low Activity`. Leave this option if your database is having a low transactional activity.

  Set this option to `Standalone` for installation on a standalone database such as a development database or on a laptop.

  Set this option to `High Activity` if the database is intensively used for transactional processing.

- By default, the STREAMS_OBJECT_GROUP option is set to `CDC`. The value entered is used to generate object names that can be shared across multiple CDC sets journalized with this JKM. If the value of this option is CDC, the naming rules listed in Table 2–3 will be applied.

  Note that this option can only take upper case ASCII characters and must not exceed 15 characters.

Table 2–3 Naming Rules Example for the CDC Group Name

Capture Process	ODI_CDC_C
Queue	ODI_CDC_Q
Queue Table	ODI_CDC_QT
Apply Process	ODI_CDC_A

- VALIDATE enables extra steps to validate the correct use of the KM. This option checks various requirements without configuring anything (for configuration steps, please see AUTO_CONFIGURATION option). When a requirement is not met, an error message is written to the log and the execution of the JKM is stopped in error.

  By default, this option is set to Yes in order to provide an easier use of this complex KM out of the box

Using the Update Date JKM

This JKM assumes that a column containing the last update date exists in all the journalized tables. This column name is provided in the UPDATE_DATE_COL_NAME knowledge module option.

2.7 Setting up Data Quality

Oracle Data Integrator provides the CKM Oracle for checking data integrity against constraints defined on an Oracle table. See "Set up Flow Control and Post-Integration Control" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details.

Oracle Data Integrator provides the Knowledge Module listed in Table 2–4 to perform a check on Oracle. It is also possible to use the generic SQL KMs. See Chapter 4, "Generic SQL" for more information.

Table 2–4 Check Knowledge Modules for Oracle Database

Recommended KM	Notes
CKM Oracle	Uses Oracle's Rowid to identify records

2.8 Designing an Interface

You can use Oracle as a source, staging area or a target of an integration interface. It is also possible to create ETL-style integration interfaces based on the Oracle technology.

The KM choice for an interface or a check determines the abilities and performance of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning an Oracle data server.

2.8.1 Loading Data from and to Oracle

Oracle can be used as a source, target or staging area of an interface. The LKM choice in the Interface Flow tab to load data between Oracle and another type of data server is essential for the performance of an interface.

2.8.1.1 Loading Data from Oracle

The following KMs implement optimized methods for loading data from an Oracle database to a target or staging area database. In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technology involved.

Target or Staging Area Technology	KM	Notes
Oracle	LKM Oracle to Oracle (dblink)	Creates a view on the source server, and synonyms on this view on the target server.
Oracle	LKM Oracle to Oracle (datapump)	Uses external tables in the datapump format.

2.8.1.2 Loading Data to Oracle

The following KMs implement optimized methods for loading data from a source or staging area into an Oracle database. In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technology involved.

Source or Staging Area Technology	KM	Notes
Oracle	LKM Oracle to Oracle (dblink)	Views created on the source server, synonyms on the target
SAP BW	LKM SAP BW to Oracle (SQLLDR)	Uses Oracle's bulk loader. File cannot be Staging Area.
SAP ERP	LKM SAP ERP to Oracle (SQLLDR)	Uses Oracle's bulk loader. File cannot be Staging Area.
Files	LKM File to Oracle (EXTERNAL TABLE)	Loads file data using external tables.
Files	LKM File to Oracle (SQLLDR)	Uses Oracle's bulk loader. File cannot be Staging Area.
Oracle	LKM Oracle to Oracle (datapump)	Uses external tables in the datapump format.
Oracle BI	LKM Oracle BI to Oracle (DBLINK)	Creates synonyms for the target staging table and uses the OBIEE populate command.
MSSQL	LKM MSSQL to Oracle (BCP/SQLLDR)	Unloads data from SQL Server using BCP, loads data into Oracle using SQL*Loader.
All	LKM SQL to Oracle	Faster than the Generic LKM (Uses Statistics)

2.8.2 Integrating Data in Oracle

The data integration strategies in Oracle are numerous and cover several modes. The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

The following KMs implement optimized methods for integrating data into an Oracle target. In addition to these KMs, you can also use the Generic SQL KMs.

Mode	KM	Note
Update	IKM Oracle Incremental Update	Optimized for Oracle.
Update	IKM Oracle Spatial Incremental Update	Supports SDO_GEOMETRY datatypes
Update	IKM Oracle Incremental Update (MERGE)	Recommended for very large volumes of data because of bulk set-based MERGE feature.
Update	IKM Oracle Incremental Update (PL SQL)	Use PL/SQL and supports long and blobs in incremental update mode.
Specific	IKM Oracle Slowly Changing Dimension	Supports type 2 Slowly Changing Dimensions
Specific	IKM Oracle Multi Table Insert	Supports multi-table insert statements.
Append	IKM Oracle to Oracle Control Append (DBLINK)	Optimized for Oracle using DB*Links

Using Slowly Changing Dimensions

For using slowly changing dimensions, make sure to set the *Slowly Changing Dimension* value for each column of the Target datastore. This value is used by the IKM Oracle Slowly Changing Dimension to identify the Surrogate Key, Natural Key, Overwrite or Insert Column, Current Record Flag and Start/End Timestamps columns.

Using Multi Table Insert

The IKM Oracle Multi Table Insert is used to integrate data from one source into one to many Oracle target tables with a multi-table insert statement. This IKM must be used in integration interfaces that are sequenced in a Package. This Package must meet the following conditions:

- The first interface of the Package must have a temporary target and the KM option *DEFINE_QUERY* set to `YES`.

  This first interface defines the structure of the SELECT clause of the multi-table insert statement (that is the source flow).

- Subsequent integration interfaces must source from this temporary datastore and have the KM option *IS_TARGET_TABLE* set to `YES`.

- The last interface of the Package must have the KM option *EXECUTE* set to `YES` in order to run the multi-table insert statement.

- Do not set *Use Temporary Interface as Derived Table (Sub-Select)* to `true` on any of the interfaces.

If large amounts of data are appended, consider to set the KM option *OPTIMIZER_HINT* to `/*+ APPEND */`.

Using Spatial Datatypes

To perform incremental update operations on Oracle Spatial datatypes, you need to declare the SDO_GEOMETRY datatype in the Topology and use the IKM Oracle Spatial Incremental Update. When comparing two columns of SDO_GEOMETREY datatype, the GEOMETRY_TOLERANCE option is used to define the error margin inside which the geometries are considered to be equal.

See the *Oracle Spatial User's Guide and Reference* for more information.

2.8.3 Designing an ETL-Style Interface

See "Working with Integration Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for generic information on how to design integration interfaces. This section describes how to design an ETL-style interface where the staging area is Oracle database or any ANSI-92 compliant database and the target on Oracle database.

In an ETL-style interface, ODI processes the data in a staging area, which is different from the target. Oracle Data Integrator provides two ways for loading the data from an Oracle staging area to an Oracle target:

- Using a Multi-connection IKM
- Using an LKM and a mono-connection IKM

Depending on the KM strategy that is used, flow and static control are supported.

Using a Multi-connection IKM

A multi-connection IKM allows updating a target where the staging area and sources are on different data servers.

Oracle Data Integrator provides the following multi-connection IKM for handling Oracle data: IKM Oracle to Oracle Control Append (DBLINK). You can also use the generic SQL multi-connection IKMs. See Chapter 4, "Generic SQL" for more information.

See Table 2–5 for more information on when to use a multi-connection IKM.

To use a multi-connection IKM in an ETL-style interface:

1. Create an integration interface with the staging area on Oracle or an ANSI-92 compliant technology and the target on Oracle using the standard procedure as described in "Creating an Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. In the Definition tab of the Interface Editor, select **Staging Area different from Target** and select the logical schema of the source tables or a third schema.
3. In the Flow tab, select one of the Source Sets, by clicking its title. The Property Inspector opens for this object.
4. Select an LKM from the LKM Selector list to load from the source(s) to the staging area. See Table 2–5 to determine the LKM you can use.
5. Optionally, modify the KM options.
6. In the Flow tab, select the Target by clicking its title. The Property Inspector opens for this object.

   In the Property Inspector, select an ETL multi-connection IKM from the IKM Selector list to load the data from the staging area to the target. See Table 2–5 to determine the IKM you can use.

Note the following when setting the KM options:

- For IKM Oracle to Oracle Control Append (DBLINK)
  - If large amounts of data are appended, set the KM option OPTIMIZER_HINT to `/*+ APPEND */`.

- Set AUTO_CREATE_DB_LINK to `true` to create automatically db link on the target schema. If AUTO_CREATE_DB_LINK is set to `false` (default), the link with this name should exist in the target schema.
- If you set the options FLOW_CONTROL and STATIC_CONTROL to Yes, select a CKM in the Controls tab. If FLOW_CONTROL is set to Yes, the flow table is created on the target.

Using an LKM and a mono-connection IKM

If there is no dedicated multi-connection IKM, use a standard exporting LKM in combination with a standard mono-connection IKM. The exporting LKM is used to load the flow table from the staging area to the target. The mono-connection IKM is used to integrate the data flow into the target table.

Oracle Data Integrator supports any ANSI SQL-92 standard compliant technology as a source of an ETL-style interface. Staging area and the target are Oracle.

See Table 2–5 for more information on when to use the combination of a standard exporting LKM and a mono-connection IKM.

To use an LKM and a mono-connection IKM in an ETL-style interface:

1. Create an integration interface with the staging area and target on Oracle using the standard procedure as described in "Creating an Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. In the Definition tab of the Interface Editor, select **Staging Area different from Target** and select the logical schema of the source tables or a third schema.
3. In the Flow tab, select one of the Source Sets.
4. In the Property Inspector, select an LKM from the LKM Selector list to load from the source(s) to the staging area. See Table 2–5 to determine the LKM you can use.
5. Optionally, modify the KM options.
6. Select the Staging Area. In the Property Inspector, select an LKM from the LKM Selector list to load from the staging area to the target. See Table 2–5 to determine the LKM you can use.
7. Optionally, modify the options.
8. Select the Target by clicking its title. The Property Inspector opens for this object.

   In the Property Inspector, select a standard mono-connection IKM from the IKM Selector list to update the target. See Table 2–5 to determine the IKM you can use.

Table 2–5 KM Guidelines for ETL-Style Interfaces with Oracle Data

Source	Staging Area	Target	Exporting LKM	IKM	KM Strategy	Comment
ANSI SQL-92 standard compliant	Oracle	Oracle	NA	IKM Oracle to Oracle Control Append (DBLINK)	Multi-connection IKM	Use this KM strategy to: ▪ Perform control append ▪ Use DB*Links for performance reasons Supports flow and static control.
ANSI SQL-92 standard compliant	Oracle or any ANSI SQL-92 standard compliant database	Oracle or any ANSI SQL-92 standard compliant database	NA	IKM SQL to SQL Incremental Update	Multi-connection IKM	Allows an incremental update strategy with no temporary target-side objects. Use this KM if it is not possible to create temporary objects in the target server. The application updates are made without temporary objects on the target, the updates are made directly from source to target. The configuration where the flow table is created on the staging area and not in the target should be used only for small volumes of data. Supports flow and static control
Oracle	Oracle	Oracle	LKM to Oracle to Oracle (DBLINK)	IKM Oracle Slowly Changing Dimension	LKM + standard IKM	
Oracle	Oracle	Oracle	LKM to Oracle to Oracle (DBLINK)	IKM Oracle Incremental Update	LKM + standard IKM	
Oracle	Oracle	Oracle	LKM to Oracle to Oracle (DBLINK)	IKM Oracle Incremental Update (MERGE)	LKM + standard IKM	

2.9 Troubleshooting

This section provides information on how to troubleshoot problems that you might encounter when using Oracle Knowledge Modules. It contains the following topics:

- Troubleshooting Oracle Database Errors
- Common Problems and Solutions

2.9.1 Troubleshooting Oracle Database Errors

Errors appear often in Oracle Data Integrator in the following way:

```
java.sql.SQLException: ORA-01017: invalid username/password; logon denied
at ...
at ...
...
```

the `java.sql.SQLExceptioncode` simply indicates that a query was made to the database through the JDBC driver, which has returned an error. This error is frequently a database or driver error, and must be interpreted in this direction.

Only the part of text in bold must first be taken in account. It must be searched in the Oracle documentation. If its contains an error code specific to Oracle, like here (in red), the error can be immediately identified.

If such an error is identified in the execution log, it is necessary to analyze the SQL code send to the database to find the source of the error. The code is displayed in the description tab of the erroneous task.

2.9.2 Common Problems and Solutions

This section describes common problems and solutions.

- `ORA-12154 TNS:could not resolve service name`

  TNS alias resolution. This problem may occur when using the OCI driver, or a KM using database links. Check the configuration of the TNS aliases on the machines.

- `ORA-02019 connection description for remote database not found`

  You use a KM using non existing database links. Check the KM options for creating the database links.

- `ORA-00900 invalid SQL statement`

  `ORA-00923 FROM Keyword not found where expected`

  The code generated by the interface, or typed in a procedure is invalid for Oracle. This is usually related to an input error in the mapping, filter of join. The typical case is a missing quote or an unclosed bracket.

  A frequent cause is also the call made to a non SQL syntax, like the call to an Oracle stored procedure using the syntax

  ```
  EXECUTE SCHEMA.PACKAGE.PROC(PARAM1, PARAM2).
  ```

  The valid SQL call for a stored procedure is:

  ```
  BEGIN
  SCHEMA.PACKAGE.PROC(PARAM1, PARAM2);
  END;
  ```

The syntax `EXECUTE SCHEMA.PACKAGE.PROC(PARAM1, PARAM2)` is specific to SQL*PLUS, and do not work with JDBC.

- `ORA-00904 invalid column name`

  Keying error in a mapping/join/filter. A string which is not a column name is interpreted as a column name, or a column name is misspelled.

  This error may also appear when accessing an error table associated to a datastore with a recently modified structure. It is necessary to impact in the error table the modification, or drop the error tables and let Oracle Data Integrator recreate it in the next execution.

- `ORA-00903 invalid table name`

  The table used (source or target) does not exist in the Oracle schema. Check the mapping logical/physical schema for the context, and check that the table physically exists on the schema accessed for this context.

- `ORA-00972 Identifier is too Long`

  There is a limit in the object identifier in Oracle (usually 30 characters). When going over this limit, this error appears. A table created during the execution of the interface went over this limit. and caused this error (see the execution log for more details).

  Check in the topology for the oracle technology, that the maximum lengths for the object names (tables and columns) correspond to your Oracle configuration.

- `ORA-01790 expression must have same datatype as corresponding expression`

  You are trying to connect two different values that can not be implicitly converted (in a mapping, a join...). Use the explicit conversion functions on these values.

3 Files

This chapter describes how to work with Files in Oracle Data Integrator.

This chapter includes the following sections:

3.1 Introduction

Oracle Data Integrator supports fixed or delimited files containing ASCII or EBCDIC data.

3.1.1 Concepts

The File technology concepts map the Oracle Data Integrator concepts as follows: A File server corresponds to an Oracle Data Integrator data server. In this File server, a directory containing files corresponds to a physical schema. A group of flat files within a directory corresponds to an Oracle Data Integrator model, in which each file corresponds to a datastore. The fields in the files correspond to the datastore columns.

Oracle Data Integrator provides a built-in driver for Files and knowledge modules for integrating Files using this driver, using the metadata declared in the File data model and in the topology.

Most technologies also have specific features for interacting with flat files, such as database loaders, utilities, and external tables. Oracle Data Integrator can also benefit from these features by using technology-specific Knowledge Modules. In terms of performance, it is most of the time recommended to use database utilities when handling flat files.

Note that the *File* technology concerns flat files (fixed and delimited). XML files are covered in Chapter 5, "XML Files".

3.1.2 Knowledge Modules

Oracle Data Integrator provides the knowledge modules (KM) listed in this section for handling File data using the File driver.

Note that the KMs listed in Table 3–1 are generic and can be used with any technology. Technology-specific KMs, using features such as loaders or external tables, are listed in the corresponding technology chapter.

Table 3–1 Knowledge Modules to read from a File

Knowledge Module	Description
LKM File to SQL	Loads data from an ASCII or EBCDIC File to any ANSI SQL-92 compliant database used as a staging area.
IKM SQL to File Append	Integrates data in a target file from any ANSI SQL-92 compliant staging area in replace mode.
RKM File (FROM EXCEL)	Retrieves file metadata from a Microsoft Excel spreadsheet. Consider using this KM if you plan to maintain the definition of your files structure in a dedicated Excel spreadsheet.

3.2 Installation and Configuration

Make sure you have read the information in this section before you start working with the File technology:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

3.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

3.2.2 Technology Specific Requirements

Some of the knowledge modules for File data use specific features of the database. This section lists the requirements related to these features.

Database Utilities

Most database technologies have their own utilities for interacting with flat files. All require that the database client software is accessible from the Agent that runs the interface that is using the utility. Some examples are:

- Oracle: SQL*Loader
- Microsoft SQL Server: bcp
- Teradata: FastLoad, MultiLoad, TPump, FastExport

You can benefit from these utilities in Oracle Data Integrator by using the technology-specific knowledge modules. See the technology-specific chapter in this guide for more information about the knowledge modules and the requirements for using the database utilities.

3.2.3 Connectivity Requirements

This section lists the requirements for connecting to flat files.

JDBC Driver

Oracle Data Integrator includes a built-in driver for flat files. This driver is installed with Oracle Data Integrator and does not require additional configuration.

3.3 Setting up the Topology

Setting up the topology consists in:

1. Creating a File Data Server
2. Creating a File Physical Schema

3.3.1 Creating a File Data Server

A File data server is a container for a set of file folders (each file folder corresponding to a physical schema).

Oracle Data Integrator provides the default FILE_GENERIC data server. This data server suits most of the needs. In most cases, it is not required to create a File data server, and you only need to create a physical schema under the FILE_GENERIC data server.

3.3.1.1 Creation of the Data Server

Create a data server for the File technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a File data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator.
   - **User/Password**: These fields are not used for File data servers.
2. In the JDBC tab, enter the following values:
   - **JDBC Driver**: `com.sunopsis.jdbc.driver.file.FileDriver`
   - **JDBC URL**: `jdbc:snps:dbfile?<property=value>&<property=value>&...`

     You can use in the URL the properties listed in Table 3–2.

Table 3–2 JDBC File Driver Properties

Property	Value	Description
ENCODING	`<encoding_code>`	File encoding. The list of supported encoding is available at `http://java.sun.com/j2se/1.4.2/docs/guide/intl/encoding.doc.html`. The default encoding value is `ISO8859_1`.
TRUNC_FIXED_STRINGS	`TRUE|FALSE`	Truncates strings to the field size for fixed files. Default value is `FALSE`.
TRUNC_DEL_STRINGS	`TRUE|FALSE`	Truncates strings to the field size for delimited files. Default value is `FALSE`.

Table 3–2 (Cont.) JDBC File Driver Properties

Property	Value	Description
OPT	`TRUE|FALSE`	Optimizes file access on multiprocessor machines for better performance. Using this option on single processor machines may actually decrease performance. Default value is `FALSE`.

JDBC URL example:

```
jdbc:snps:dbfile?ENCODING=ISO8859_1&TRUNC_FIXED_
STRINGS=FALSE&OPT=TRUE
```

3.3.2 Creating a File Physical Schema

Create a File physical schema using the standard procedure, as described in "Creating a physical schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

In your physical schema, you must set a pair of directories:

- The **Directory (Schema)**, where Oracle Data Integrator will look for the source and target files and create error files for invalid records detected in the source files.
- A **Directory (Work Schema)**, where Oracle Data Integrator may create temporary files associated to the sources and targets contained in the Data Schema.

Notes:

- Data and Work schemas each correspond to a directory. This directory must be accessible to the component that will access the files. The directory can be an absolute path (`m:/public/data/files`) or relative to the runtime agent or Studio startup directory (`../demo/files`). It is strongly advised to use a path that is independent from the execution location.
- In UNIX in particular, the agent must have read/write permission on both these directories.
- Keep in mind that file paths are different in Windows than they are in UNIX. Take the platform used by the agent into account when setting up this information.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

3.4 Setting Up an Integration Project

Setting up a project using the File database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started:

- LKM File to SQL

- IKM SQL to File Append
- RKM File (FROM EXCEL)

In addition to these knowledge modules, you can also import file knowledge modules specific to the other technologies involved in your product.

3.5 Creating and Reverse-Engineering a File Model

This section contains the following topics:

- Create a File Model
- Reverse-engineer a File Model

3.5.1 Create a File Model

An File model is a set of datastores, corresponding to files stored in a directory. A model is always based on a logical schema. In a given context, the logical schema corresponds to one physical schema. The data schema of this physical schema is the directory containing all the files (eventually in sub-directories) described in the model.

Create a File model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

3.5.2 Reverse-engineer a File Model

Oracle Data Integrator provides specific methods for reverse-engineering files. File database supports four types of reverse-engineering:

- Delimited Files Reverse-Engineering is performed per file datastore.
- Fixed Files Reverse-engineering using the Wizard is performed per file datastore.
- COBOL Copybook reverse-engineering, which is available for fixed files, if a copybook describing the file is provided. It is performed per file datastore.
- Customized Reverse-Engineering, which uses a RKM (Reverse Knowledge Module) to get the structure of all of the files of the model from a Microsoft Excel spreadsheet. Note that if you use the RKM, you do not need to define manually the datastores by typing in each column definition because the RKM automatically defines the column definitions in the spreadsheet.

Note: The built-in file driver uses metadata from the Oracle Data Integrator models (field data type or length, number of header rows, etc.). Driver-specific tags are generated by Oracle Data Integrator and passed to the driver along with regular SQL commands. These tags control how the driver reads or writes the file.

Similarly, when Oracle Data Integrator uses database loaders and utilities, it uses the model metadata to control these loaders and utilities.

It is important to pay close attention to the file definition after a reverse-engineering process, as discrepancy between the file definition and file content is a source of issues at run-time.

3.5.2.1 Delimited Files Reverse-Engineering

To perform a delimited file reverse-engineering:

1. In the Models accordion, right click your File Model and select **New Datastore**. The Datastore Editor opens.
2. In the Definition Tab, enter the following fields:
   - **Name**: Name of this datastore
   - **Resource Name**: Sub-directory (if needed) and name of the file. You can browse for the file using the (...) button
3. Go to the Files tab to describe the type of file. Set the fields as follows:
   - **File Format**: Delimited
   - **Header (Number of Lines)**: Enter the number of lines of the header. Note that if there is a header, the first line of the header will be used by Oracle Data Integrator to name the columns in the file.
   - Select a **Record Separator**.
   - Select or enter the character used as a **Field Separator**.
   - Enter a **Text Delimiter** if your file uses one.
   - Enter a **Decimal Separator** if your file contains decimals.
4. From the File main menu, select **Save**.
5. In the Datastore Editor, go to the Columns tab.
6. In the editor toolbar, click **Reverse-Engineer**.
7. Verify the datatype and length for the reverse engineered columns. Oracle Data Integrator infers the fields datatypes and lengths from the file content, but may set default values (for example 50 for the strings field length) or incorrect data types in this process.
8. From the File main menu, select **Save**.

3.5.2.2 Fixed Files Reverse-engineering using the Wizard

Oracle Data Integrator provides a wizard to graphically define the columns of a fixed file.

To reverse-engineer a fixed file using the wizard:

1. In the Models accordion, right click your File Model and select **New Datastore**. The Datastore Editor opens.
2. In the Definition Tab, enter the following fields:
   - **Name**: Name of this datastore
   - **Resource Name**: Sub-directory (if needed) and name of the file. You can browse for the file using the (...) button
3. Go to the Files tab to describe the type of file. Set the fields as follows:
   - **File Format**: Fixed
   - **Header (Number of Lines)**: Enter the number of lines of the header.
   - Select a **Record Separator**.
4. From the File main menu, select **Save**.
5. In the Datastore Editor, go to the Columns tab.

6. In the editor toolbar, click **Reverse-Engineer**.The Columns Setup Wizard is launched. The Columns Setup Wizard displays the first records of your file.
7. Click on the ruler (above the file contents) to create markers delimiting the columns. You can right-click within the ruler to delete a marker.
8. Columns are created with pre-generated names (C1, C2, and so on). You can edit the column name by clicking in the column header line (below the ruler).
9. In the properties panel (on the right), you can edit all the parameters of the selected column. You should set at least the Column Name, Datatype, and Length for each column.
10. Click **OK** when the columns definition is complete.
11. From the File main menu, select **Save**.

3.5.2.3 COBOL Copybook reverse-engineering

COBOL Copybook reverse-engineering allows you to retrieve a legacy file structure from its description contained in a COBOL Copybook file.

To reverse-engineer a fixed file using a COBOL Copybook:

1. In the Models accordion, right click your File Model and select **New Datastore**. The Datastore Editor opens.
2. In the Definition Tab, enter the following fields:
   - **Name**: Name of this datastore
   - **Resource Name**: Sub-directory (if needed) and name of the file. You can browse for the file using the (...) button
3. Go to the Files tab to describe the type of file. Set the fields as follows:
   - **File Format**: Fixed
   - **Header (Number of Lines)**: Enter the number of lines of the header.
   - Select a **Record Separator**.
4. From the File main menu, select **Save**.
5. In the Datastore Editor, go to the Columns tab.
6. Create or open a File datastore that has a fixed format.
7. In the Datastore Editor, go to the Columns tab.
8. In the toolbar menu, click **Reverse Engineer COBOL CopyBook**.
9. In the Reverse Engineer Cobol CopyBook Dialog, enter the following fields:
   - **File**: Location of the Copybook file
   - **Character set**: Copybook file charset.
   - **Description format** (`EBCDIC | ASCII`): Copybook file format
   - **Data format** (`EBCDIC | ASCII`): Data file format
10. Click **OK**.

The columns described in the Copybook are reverse-engineered and appear in the column list.

Note: If a field has a data type declared in the Copybook with no corresponding datatype in Oracle Data Integrator File technology, then this column will appear with no data type.

3.5.2.4 Customized Reverse-Engineering

In this reverse-engineering method, Oracle Data Integrator uses a Microsoft Excel spreadsheet that contains the description of the group of files. This file has a specific format. A sample file (`file_repository.xls`) is provided in the Oracle Data Integrator demo in the `/demo/excel` sub-directory.

The following steps assume that you have modified this file with the description of the structure of your flat files.

To perform a customized reverse-engineering, perform the following steps:

1. Create an ODBC Datasource for the Excel Spreadsheet corresponding to the Excel Spreadsheet containing the files description.
2. Define the Data Server, Physical and Logical Schema for the Microsoft Excel Spreadsheet
3. Run the customized reverse-engineering using the RKM File from Excel RKM.

Create an ODBC Datasource for the Excel Spreadsheet

1. Launch the Microsoft ODBC Administrator.
2. Add a System Datasource.
3. Select the Microsoft Excel Driver (*.xls) driver.
4. Name the data source: `ODI_EXCEL_FILE_REPO` and select the file `/demo/excel/file_repository.xls`.

Define the Data Server, Physical and Logical Schema for the Microsoft Excel Spreadsheet

1. In Topology Navigator, add a Microsoft Excel data server with the following parameters:
   - **Name**: EXCEL_FILE_REPOSITORY
   - **JDBC Driver**: `sun.jdbc.odbc.JdbcOdbcDriver`
   - **JDBC URL**: `jdbc:odbc:ODI_EXCEL_FILE_REPO`
2. From the File main menu, select **Save**.
3. Add a physical schema to this data server. Leave the default values in the Definition tab.
1. In the Context tab of the physical schema, click **Add**.
2. In the new line, select the context that will be used for reverse engineering and enter in the logical schema column `EXCEL_FILE_REPOSITORY`. This name is mandatory.
3. From the File main menu, select **Save**.

Run the customized reverse-engineering

1. In Designer Navigator, import the RKM File From Excel Knowledge Module into your project.

2. In the Models accordion, double-click the File Model. The Model Editor opens.
3. In the **Reverse-Engineer** Tab, set the following parameters:
   - Select **Customized**
   - **Context**: Reverse Context
   - **Knowledge Module**: RKM File from Excel
4. In the toolbar menu, click **Reverse-Engineer**.
5. You can follow the reverse-engineering process in the execution log

Note:

- A Microsoft Excel logical schema must be defined. It must be named EXCEL_FILE_REPOSITORY and point to the file `file_repository.xls` or another file with a similar structure.
- The Microsoft Excel file `file_repository.xls` should be closed before running the reverse engineering.

3.6 Designing an Interface

You can use a file as a source or a target of an integration interface, but **NOT** as a staging area.

The KM choice for an interface or a check determines the abilities and performances of this interface or check. The recommendations below help in the selection of the KM for different situations concerning a File data server.

3.6.1 Loading Data From Files

Files can be used as a source of an interface. The LKM choice in the Interface Flow tab to load a File to the staging area is essential for the interface performance.

The LKM File to SQL uses the built-in file driver for loading data from a File database to a staging area. In addition to this KM, you can also use KMs that are specific to the technology of the staging area or target. Such KMs support technology-specific optimizations and use methods such as loaders or external tables.

This knowledge module, as well as other KMs relying on the built-in driver, support the following two features attached to the driver:

- Erroneous Records Handling
- Multi-Record Files Support

Erroneous Records Handling

Oracle Data Integrator built-in driver provides error handling at column level for the File technology. When loading a File, Oracle Data Integrator performs several controls. One of them verifies if the data in the file is consistent with the datastore definition. If one value from the row is inconsistent with the column description, the *On Error* option - on the Control tab of the Column Editor - defines the action to perform. The On Error option can take the following values:

- **Reject Error**: The row containing the error is moved to a .BAD file, and a reason of the error is written to a .ERROR file.

The .BAD and .ERROR files are located in the same directory as the file being read and are named after this file, with a .BAD and .ERROR extension.

- **Null if error (inactive trace)**: The row is kept in the flow and the erroneous value is replaced by null.
- **Null if error (active trace)**: The row is kept in the flow, the erroneous value is replaced by null, and an reason of the error is written to the .ERROR file.

Multi-Record Files Support

Oracle Data Integrator is able to handle files that contain multiple record formats. For example, a file may contain records representing *orders* (these records have 5 columns) and other records representing *order lines* (these records having 8 columns with different datatypes).

The approach in Oracle Data Integrator consists in considering each specific record format as a different datastore.

Example 3–1 Multi Record File

This example uses the multi record file `orders.txt`. It contains two different record types: *orders* and *order lines*.

Order records have the following format:

```
REC_CODE,ORDER_ID,CUSTOMER_ID,ORDER_DATE
```

Order lines records have the following format

```
REC_CODE,ORDER_ID,LINE_ID,PRODUCT_ID,QTY
```

Order records are identified by `REC_CODE=ORD`

Order lines are identified by `REC_CODE=LIN`

To handle multi record files as a source interface of an integration interface:

1. Create a File Model using a logical schema that points to the directory containing the source file.
2. Identify the different record formats and structures of the flat file. In Example 3–1 two record formats can be identified: one for the *orders* and one for the *order lines*.
3. For each record format identified, do the following:
    1. Create a datastore in the File Model for each type of record.

       For Example 3–1 create two datastores.
    2. In the Definition tab of the Datastore Editor, enter a unique name in the Name field and enter the flat file name in the Resource Name field. Note that the resource name is identical for all datastores of this model.

       For Example 3–1 you can use `ORDERS` and `ORDER_LINES` as the name of your datastores. Enter `orders.txt` in the Resource Name field for both datastores.
    3. In the Files tab, select, depending on the format of your flat file, **Fixed** or **Delimited** from the File Format list and specify the record and field separators.
    4. In the Columns tab, enter the column definitions for this record type.
    5. One or more columns can be used to identify the record type. The record code is the field value content that is used as distinguishing element to be found in the file. The record code must be unique and allows files with several record

patterns to be processed. In the Record Codes field, you can specify several values separated by the semicolon (;) character.

In the Column Editor, assign a record code for each record type in the **Record Codes** field.

In Example 3–1, enter `ORD` in the Record Codes field of the CODE_REC column of the ORDERS datastore and enter `LIN` in the Record Codes field of the CODE_REC column of the ORDER_LINES datastore.

With such definition, when reading data from the ORDERS datastore, the file driver will filter only those of the records where the first column contains the value ORD. The same applies to the ORDER_LINES datastore (only the records with the first column containing the value LIN will be returned).

3.6.2 Integrating Data in Files

Files can be used as a target of an interface. The data integration strategies in Files concern loading from the staging area to Files. The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

The IKM SQL to File Append uses the file driver for integrating data into a Files target from a staging area in truncate-insert mode.

This KM has the following options:

- INSERT automatically attempts to insert data into the target datastore of the interface.
- CREATE_TARG_TABLE creates the target table.
- TRUNCATE deletes the content of the target file and creates it if it does not exist.
- GENERATE_HEADER creates the header row for a delimited file.

In addition to this KM, you can also use IKMs that are specific to the technology of the staging area. Such KMs support technology-specific optimizations and use methods such as loaders or external tables.

4

Generic SQL

This chapter describes how to work with technologies supporting the ANSI SQL-92 syntax in Oracle Data Integrator.

> **Note:** This is a generic chapter. The information described in this chapter can be applied to technologies supporting the ANSI SQL-92 syntax, including Oracle, Microsoft SQL Server, Sybase ASE, IBM DB2, Teradata, PostgreSQL, MySQL, Derby and so forth.
>
> Some of the ANSI SQL-92 compliant technologies are covered in a separate chapter in this guide. Refer to the dedicated technology chapter for specific information on how to leverage the ODI optimizations and database utilities of the given technology.

This chapter includes the following sections:

4.1 Introduction

Oracle Data Integrator supports ANSI SQL-92 standard compliant technologies.

4.1.1 Concepts

The mapping of the concepts that are used in ANSI SQL-92 standard compliant technologies and the Oracle Data Integrator concepts are as follows: a data server in Oracle Data Integrator corresponds to a data processing resource that stores and serves data in the form of tables. Depending on the technology, this resource can be named for example, database, instance, server and so forth. Within this resource, a sub-division maps to an Oracle Data Integrator physical schema. This sub-division can be named schema, database, catalog, library and so forth. A set of related objects

within one schema corresponds to a data model, and each table, view or synonym will appear as an ODI datastore, with its attributes, columns, and constraints

4.1.2 Knowledge Modules

Oracle Data Integrator provides a wide range of Knowledge Modules for handling data stored in ANSI SQL-92 standard compliant technologies. The Knowledge Modules listed in Table 4–1 are generic SQL Knowledge Modules and apply to the most popular ANSI SQL-92 standard compliant databases.

Oracle Data Integrator also provides specific Knowledge Modules for some particular databases to leverage the specific utilities. Technology-specific KMs, using features such as loaders or external tables, are listed in the corresponding technology chapter.

Table 4–1 Generic SQL Knowledge Modules

Knowledge Module	Description
CKM SQL	Checks data integrity against constraints defined on a Datastore. Rejects invalid records in the error table created dynamically. Can be used for static controls as well as for flow controls. Consider using this KM if you plan to check data integrity on an ANSI SQL-92 compliant database. Use specific CKMs instead if available for your database.
IKM SQL Control Append	Integrates data in an ANSI SQL-92 compliant target table in replace/append mode. When flow data needs to be checked using a CKM, this IKM creates a temporary staging table before invoking the CKM. Consider using this IKM if you plan to load your SQL compliant target table in replace mode, with or without data integrity check. To use this IKM, the staging area must be on the same data server as the target.
IKM SQL Incremental Update	Integrates data in an ANSI SQL-92 compliant target table in incremental update mode. This KM creates a temporary staging table to stage the data flow. It then compares its content to the target table to idetinfythe records to insert and the records to update. It also allows performing data integrity check by invoking the CKM. This KM is therefore not recommended for large volumes of data. Consider using this KM if you plan to load your ANSI SQL-92 compliant target table to insert missing records and to update existing ones. Use technology-specific incremental update IKMs whenever possible as they are more optimized for performance. To use this IKM, the staging area must be on the same data server as the target.
IKM SQL to File Append	Integrates data in a target file from an ANSI SQL-92 compliant staging area in replace mode. Consider using this IKM if you plan to transform and export data to a target file. If your source datastores are located on the same data server, we recommend using this data server as staging area to avoid extra loading phases (LKMs) To use this IKM, the staging area must be different from the target.
IKM SQL to SQL Control Append	Integrates data into a ANSI-SQL92 target database from any ANSI-SQL92 compliant staging area. This IKM is typically used for ETL configurations: source and target tables are on different databases and the interface's staging area is set to the logical schema of the source tables or a third schema.
IKM SQL to SQL Incremental Update	Integrates data from any AINSI-SQL92 compliant database into any any AINSI-SQL92 compliant database target table in incremental update mode. This IKM is typically used for ETL configurations: source and target tables are on different databases and the interface's staging area is set to the logical schema of the source tables or a third schema.

Table 4–1 (Cont.) Generic SQL Knowledge Modules

Knowledge Module	Description
LKM File to SQL	Loads data from an ASCII or EBCDIC File to an ANSI SQL-92 compliant database used as a staging area. This LKM uses the Agent to read selected data from the source file and write the result in the staging temporary table created dynamically. Consider using this LKM if one of your source datastores is an ASCII or EBCDIC file. Use technology-specific LKMs for your target staging area whenever possible as they are more optimized for performance. For example, if you are loading to an Oracle database, use the LKM File to Oracle (SQLLDR) or the LKM File to Oracle (EXTERNAL TABLE) instead.
LKM SQL to SQL	Loads data from an ANSI SQL-92 compliant database to an ANSI SQL-92 compliant staging area. This LKM uses the Agent to read selected data from the source database and write the result into the staging temporary table created dynamically. Consider using this LKM if your source datastores are located on a SQL compliant database different from your staging area. Use technology-specific LKMs for your source and target staging area whenever possible as they are more optimized for performance. For example, if you are loading from an Oracle source server to an Oracle staging area, use the LKM Oracle to Oracle (dblink) instead.
LKM SQL to SQL (row by row)	Loads data from any ISO-92 database to any ISO-92 compliant target database. This LKM uses a Jython script to read selected data from the database and write the result into the target temporary table, which is created dynamically. It loads data from a staging area to a target and indicates the state of each processed row. The following options are used for the logging mechanism: ■ LOG_LEVEL: This option is used to set the granularity of the data logged. The following log levels can be set: ■ 0: nothing to log ■ 1: any JDBC action will be indicated such as 'select action', 'delete action', 'insert action'… ■ 2: in addition to level 1, all records that generate an error will be logged ■ 3: in addition to level 2, all processed records will be logged ■ LOG_FILE_NAME: Full path to the log file used. ■ MAX_ERRORS:Specify the maximum number of errors. The LKM process stops when the maximum number of errors specified in this option is reached. This Knowledge Module is NOT RECOMMENDED when using LARGE VOLUMES. Other specific modules using Bulk utilities (SQL*LOADER, BULK INSERT...) or direct links (DBLINKS, Linked Servers...) are usually more efficient.

Table 4–1 (Cont.) Generic SQL Knowledge Modules

Knowledge Module	Description
LKM SQL to SQL (JYTHON)	Loads data from an ANSI SQL-92 compliant database to an ANSI SQL-92 compliant staging area. This LKM uses Jython scripting to read selected data from the source database and write the result into the staging temporary table created dynamically. This LKM allows you to modify the default JDBC data type binding between the source database and the target staging area by editing the underlying Jython code provided. Consider using this LKM if your source datastores are located on an ANSI SQL-92 compliant database different from your staging area and if you plan to specify your own data type binding method. Use technology-specific LKMs for your source and target staging area whenever possible as they are more optimized for performance. For example, if you are loading from an Oracle source server to an Oracle staging area, use the LKM Oracle to Oracle (dblink) instead.
RKM SQL (JYTHON)	Retrieves JDBC metadata for tables, views, system tables and columns from an ANSI SQL-92 compliant database. This RKM may be used to specify your own strategy to convert JDBC metadata into Oracle Data Integrator metadata. Consider using this RKM if you encounter problems with the standard JDBC reverse-engineering process due to some specificities of your JDBC driver. This RKM allows you to edit the underlying Jython code to make it match the specificities of your JDBC driver.
SKM SQL	Generates data access Web services for ANSI SQL-92 compliant databases. Data access services include data manipulation operations such as adding, removing, updating or filtering records as well as changed data capture operations such as retrieving changed data. Data manipulation operations are subject to integrity check as defined by the constraints of your datastores. Consider using this SKM if you plan to generate and deploy data manipulation or changed data capture web services to your Service Oriented Architecture infrastructure. Use specific SKMs instead if available for your database

4.2 Installation and Configuration

Make sure you have read the information in this section before you start using the generic SQL Knowledge Modules:

- System Requirements and Certifications
- Technology-Specific Requirements
- Connectivity Requirements

4.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

4.2.2 Technology-Specific Requirements

See the Technology Specific Requirements section of the specific technology chapter for more information.

If your technology does not have a dedicated chapter in this guide, see the documentation of your technology for any technology-specific requirements.

4.2.3 Connectivity Requirements

See the Connectivity Requirements section of the specific technology chapter for more information.

The Java Database Connectivity (JDBC) is the standard for connecting to a database and other data sources. If your technology does not have a dedicated chapter in this guide, see the documentation of your technology for the JDBC configuration information, including the required driver files, the driver name, and the JDBC URL format.

4.3 Setting up the Topology

Setting up the Topology consists in:

1. Creating a Data Server
2. Creating a Physical Schema

4.3.1 Creating a Data Server

Create a data server under the ANSI SQL-92 compliant technology listed in the Physical Architecture accordion using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

If your technology has a dedicated chapter in this guide, see this chapter for more information. For other technologies, see the documentation of your technology for the JDBC driver name and JDBC URL format.

4.3.2 Creating a Physical Schema

Create a Physical Schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

If your technology has a dedicated chapter in this guide, see this chapter for more information.

4.4 Setting up an Integration Project

Setting up a Project using an ANSI SQL-92 compliant database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The recommended knowledge modules to import into your project for getting started depend on the corresponding technology. If your technology has a dedicated chapter in this guide, see this chapter for more information.

4.5 Creating and Reverse-Engineering a Model

This section contains the following topics:

- Create a Data Model

- Reverse-engineer a Data Model

4.5.1 Create a Data Model

Create a data model based on the ANSI SQL-92 compliant technology using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

If your technology has a dedicated chapter in this guide, see this chapter for more information.

4.5.2 Reverse-engineer a Data Model

ANSI SQL-92 standard compliant technologies support both types of reverse-engineering, the Standard reverse-engineering, which uses only the abilities of the JDBC driver, and the Customized reverse-engineering, which uses a RKM which provides logging features.

In most of the cases, consider using the standard JDBC reverse engineering instead of the RKM SQL (Jython). However, you can use this RKM as a starter if you plan to enhance it by adding your own metadata reverse-engineering behavior.

Standard Reverse-Engineering

To perform a Standard Reverse- Engineering on ANSI SQL-92 technologies use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

If your technology has a dedicated chapter in this guide, see this chapter for more information.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on ANSI SQL-92 technologies with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the usage of the RKM SQL (Jython):

This RKM provides two logging options:

- USE_LOG: Set to Yes if you want the reverse-engineering to process log details in a log file.
- LOG_FILE_NAME: Enter the name for the log file.

4.6 Setting up Changed Data Capture

Oracle Data Integrator does not provide journalizing Knowledge Modules for ANSI SQL-92 compliant technologies.

4.7 Setting up Data Quality

Oracle Data Integrator provides the CKM SQL for checking data integrity against constraints defined on an ANSI SQL-92 compliant table. See "Set up Flow Control and Post-Integration Control" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details.

4.8 Designing an Interface

You can use ANSI SQL-92 compliant technologies as a source, staging area or a target of an integration interface. It is also possible to create ETL-style integration interfaces based on an ANSI SQL-92 compliant technology.

The KM choice for an interface or a check determines the abilities and performances of this interface or check. The recommendations below help in the selection of the KM for different situations concerning a data server based on an ANSI SQL-92 compliant technology.

4.8.1 Loading Data From and to an ANSI SQL-92 Compliant Technology

ANSI SQL-92 compliant technologies can be used as a source, target or staging area of an interface. The LKM choice in the Interface Flow tab to load data between an ANSI SQL-92 compliant technology and another type of data server is essential for the performance of an interface.

4.8.1.1 Loading Data from an ANSI SQL-92 Compliant Technology

The generic KMs that are listed in Table 4–2 implement methods for loading data from an ANSI SQL-92 compliant database to a target or staging area database. In addition to these KMS, Oracle Data Integrator provides KMs specific to the target or staging area database. If your technology has a dedicated chapter in this guide, see this chapter for more information.

Table 4–2 KMs to Load from an ANSI SQL-92 Compliant Technology

Source or Staging Area	KM	Notes
ANSI SQL-92 compliant technology	LKM SQL to SQL	Standard KM for SQL-92 to SQL-92 transfers
ANSI SQL-92 compliant technology	LMK SQL to SQL (Jython)	This LKM uses Jython scripting to read selected data from the source database and write the result into the staging temporary table created dynamically. This LKM allows you to modify the default JDBC data types binding between the source database and the target staging area by editing the underlying Jython code provided.
ANSI SQL-92 compliant technology	LMK SQL to SQL (row by row)	This LKM uses row by row logging.

4.8.1.2 Loading Data to an ANSI SQL-92 Compliant Technology

The generic KMs that are listed in Table 4–3 implement methods for loading data from a source or staging area into an ANSI SQL-92 compliant database. In addition to these KMs, Oracle Data Integrator provides KMs specific to the source or staging area database. If your technology has a dedicated chapter in this guide, see this chapter for more information.

Table 4–3 KMs to Load to an ANSI SQL-92 Compliant Technology

Source or Staging Area	KM	Notes
File	LKM File to SQL	Standard KM
ANSI SQL-92 compliant technology	LKM SQL to SQL	Standard KM

Table 4–3 (Cont.) KMs to Load to an ANSI SQL-92 Compliant Technology

Source or Staging Area	KM	Notes
ANSI SQL-92 compliant technology	LMK SQL to SQL (Jython)	This LKM uses Jython scripting to read selected data from the source database and write the result into the staging temporary table created dynamically. This LKM allows you to modify the default JDBC data types binding between the source database and the target staging area by editing the underlying Jython code provided.
ANSI SQL-92 compliant technology	LMK SQL to SQL (row by row)	This LKM uses row by row logging.

4.8.2 Integrating Data in an ANSI SQL-92 Compliant Technology

An ANSI SQL-92 compliant technology can be used as a target of an interface. The IKM choice in the Interface Flow tab determines the performance and possibilities for integrating.

The KMs listed in Table 4–4 implement methods for integrating data into an ANSI SQL-92 compliant target. In addition to these KMs, Oracle Data Integrator provides KMs specific to the source or staging area database. See the corresponding technology chapter for more information.

Table 4–4 KMs to Integrate Data in an ANSI SQL-92 Compliant Technology

Source or Staging Area	KM	Notes
ANSI SQL-92 compliant technology	IKM SQL Control Append	Uses Bulk data movement inside data server
ANSI SQL-92 compliant technology	IKM SQL Incremental Update	Uses Bulk data movement inside data server
ANSI SQL-92 compliant technology	IKM SQL to File Append	Uses agent for data movement
ANSI SQL-92 compliant technology	IKM SQL to SQL Incremental Update	Uses agent or JYTHON for data movement
ANSI SQL-92 compliant technology	IKM SQL to SQL Control Append	Uses agent for control append strategies

4.8.3 Designing an ETL-Style Interface

See "Working with Integration Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for generic information on how to design integration interfaces. This section describes how to design an ETL-style interface where the staging area and target are ANSI SQL-92 compliant.

In an ETL-style interface, ODI processes the data in a staging area, which is different from the target. Oracle Data Integrator provides two ways for loading the data from an ANSI SQL-92 compliant staging area to an ANSI SQL-92 compliant target:

- Using a Multi-connection IKM
- Using a LKM and a mono-connection IKM

Depending on the KM strategy that is used, flow and static control are supported.

Using a Multi-connection IKM

A multi-connection IKM allows updating a target where the staging area and sources are on different data servers.

Oracle Data Integrator provides the following multi-connection IKMs for ANSI SQL-92 compliant technologies: IKM SQL to SQL Incremental Update and IKM SQL to SQL Control Append.

See Table 4–5 for more information on when to use a multi-connection IKM.

To use a multi-connection IKM in an ETL-style interface:

1. Create an integration interface with an ANSI SQL-92 compliant staging area and target using the standard procedure as described in "Creating an Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. In the Definition tab of the Interface Editor, select **Staging Area different from Target** and select the logical schema of the source tables or a third schema.
3. In the Flow tab, select one of the Source Sets, by clicking its title. The Property Inspector opens for this object.
4. Select an LKM from the LKM Selector list to load from the source(s) to the staging area. See Table 4–5 to determine the LKM you can use.
5. Optionally, modify the KM options.
6. In the Flow tab, select the Target by clicking its title. The Property Inspector opens for this object.

   In the Property Inspector, select an ETL multi-connection IKM from the IKM Selector list to load the data from the staging area to the target. See Table 4–5 to determine the IKM you can use.

Note the following when setting the KM options:

- For IKM SQL to SQL Incremental Update
  - If you do not want to create any tables on the target system, set `FLOW_CONTROL=false` and `FLOW_TABLE_LOCATION=STAGING`.

    Please note that this will lead to row-by-row processing and therefore significantly lower performance.
  - If you set the options FLOW_CONTROL or STATIC_CONTROL to `true`, select a CKM in the Controls tab. Note that if FLOW_CONTROL is set to `true`, the flow table is created on the target, regardless of the value of FLOW_TABLE_LOCATION.
  - The FLOW_TABLE_LOCATION option can take the following values:

Value	Description	Comment
TARGET	Objects are created on the target.	Default value.
STAGING	Objects are created only on the staging area, not on the target.	Cannot be used with flow control. Leads to to row-by-row processing and therefore loss of performance.
NONE	No objects are created on staging area nor target.	Cannot be used with flow control. Leads to to row-by-row processing and therefore loss of performance. Requires to read source data twice in case of journalized data sources

Using a LKM and a mono-connection IKM

If there is no dedicated multi-connection IKM, use a standard exporting LKM in combination with a standard mono-connection IKM. The exporting LKM is used to load the flow table from the staging area to the target. The mono-connection IKM is used to integrate the data flow into the target table.

Oracle Data Integrator supports any ANSI SQL-92 standard compliant technology as a source, staging area, and target of an ETL-style interface.

See Table 4–5 for more information on when to use the combionation of a standard LKM and a mono-connection IKM.

To use an LKM and a mono-connection IKM in an ETL-style interface:

1. Create an integration interface with an ANSI SQL-92 compliant staging area and target using the standard procedure as described in "Creating an Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. In the Definition tab of the Interface Editor, select **Staging Area different from Target** and select the logical schema of the source tables or a third schema.
3. In the Flow tab, select one of the Source Sets.
4. In the Property Inspector, select an LKM from the LKM Selector list to load from the source(s) to the staging area. See Table 4–5 to determine the LKM you can use.
5. Optionally, modify the KM options.
6. Select the Staging Area. In the Property Inspector, select an LKM from the LKM Selector list to load from the staging area to the target. See Table 4–5 to determine the LKM you can use.
7. Optionally, modify the options.
8. Select the Target by clicking its title. The Property Inspector opens for this object.

   In the Property Inspector, select a standard mono-connection IKM from the IKM Selector list to update the target. See Table 4–5 to determine the IKM you can use.

Table 4–5 KM Guidelines for ETL-Style Interfaces based on an ANSI SQL-92 standard compliant technology

Source	Staging Area	Target	Exporting LKM	IKM	KM Strategy	Comment
ANSI SQL-92 standard compliant	ANSI SQL-92 standard compliant database	ANSI SQL-92 standard compliant database	NA	IKM SQL to SQL Incremental Update	Multi-connection IKM	Allows an incremental update strategy with no temporary target-side objects. Use this KM if it is not possible to create temporary objects in the target server. The application updates are made without temporary objects on the target, the updates are made directly from source to target. The configuration where the flow table is created on the staging area and not in the target should be used only for small volumes of data. Supports flow and static control
ANSI SQL-92 standard compliant	ANSI SQL-92 standard compliant database	ANSI SQL-92 standard compliant database	NA	IKM SQL to SQL Control Append	Multi-connection IKM	Use this KM strategy to perform control append. Supports flow and static control.
ANSI SQL-92 standard compliant	ANSI SQL-92 standard compliant database	ANSI SQL-92 standard compliant database	any standard KM loading from an ANSI SQL-92 standard compliant technology to an ANSI SQL-92 standard compliant technology	IKM SQL Incremetal Update	Mono-connection IKM	Allows an incremental update strategy

5

XML Files

This chapter describes how to work with XML files in Oracle Data Integrator.

This chapter includes the following sections:

5.1 Introduction

Oracle Data Integrator supports XML files integration through the Oracle Data Integrator Driver for XML.

5.1.1 Concepts

The XML concepts map the Oracle Data Integrator concepts as follows: An XML file corresponds to a data server in Oracle Data Integrator. Within this data server, a single schema maps the content of the XML file.

The Oracle Data Integrator Driver for XML (XML driver) loads the hierarchical structure of the XML file into a relational schema. This relational schema is a set of tables located in the schema that can be queried or modified using SQL. The XML driver is also able to unload the relational schema back in the XML file.

The relational schema is reverse-engineered as a data model in ODI, with tables, columns, and constraints. This model is used like a normal relational data model in ODI. If the modified data within the relational schema needs to be written back to the XML file, the XML driver provides the capability to *synchronize* the relational schema into the file.

See Appendix B, "Oracle Data Integrator Driver for XML Reference" for more information on this driver.

5.1.2 Knowledge Modules

Oracle Data Integrator provides the IKM XML Control Append for handling XML data. This Knowledge Module is a specific XML Knowledge Module. It has a specific option to synchronize the data from the relational schema to the file.

In addition to this KM, you can also use an XML data server as any SQL data server. XML data servers support both the technology-specific KMs sourcing or targeting SQL data servers, as well as the generic KMs. See Chapter 4, "Generic SQL" or the technology chapters for more information on these KMs.

5.2 Installation and Configuration

Make sure you have read the information in this section before you start using the XML Knowledge Module:

- System Requirements
- Technologic Specific Requirements
- Connectivity Requirements

5.2.1 System Requirements

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

5.2.2 Technologic Specific Requirements

There are no technology-specific requirements for using XML Files in Oracle Data Integrator.

5.2.3 Connectivity Requirements

This section lists the requirements for connecting to XML database.

Oracle Data Integrator Driver for XML

XML files are accessed through the Oracle Data Integrator Driver for XML. This JDBC driver is installed with Oracle Data Integrator and requires no other installed component or configuration.

You must ask the system administrator for the following connection information:

- The location of the DTD or XSD file associated with your XML file
- The location of the XML file

5.3 Setting up the Topology

Setting up the topology consists in:

1. Creating an XML Data Server

2. Creating a Physical Schema for XML

5.3.1 Creating an XML Data Server

An XML data server corresponds to one XML file that is accessible to Oracle Data Integrator.

5.3.1.1 Creation of the Data Server

Create a data server for the XML technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a File data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator.
   - **User/Password**: These fields are not used for XML data servers.
2. In the JDBC tab, enter the values according to the driver used:
   - **JDBC Driver**: `com.sunopsis.jdbc.driver.xml.SnpsXmlDriver`
   - **JDBC URL**: `jdbc:snps:xml?[property=value&property=value...]`

   Table 5–1 lists the key properties of the Oracle Data Integrator Driver for XML. These properties can be specified in the JDBC URL.

   See Appendix B, "Oracle Data Integrator Driver for XML Reference" for a detailed description of these properties and for a comprehensive list of all properties.

Table 5–1 JDBC Driver Properties

Property	Value	Notes
f	<XML File location>	XML file name. Use slash "/" in the path name instead of back slash "\". It is possible to use an HTTP, FTP or File URL to locate the file. Files located by URL are read-only.
d	<DTD/XSD File location>	Description file: This file may be a DTD or XSD file. It is possible to use an HTTP, FTP or File URL to locate the file. Files located by URL are read-only. Note that when no DTD or XSD file is present, the relational schema is built using only the XML file content. It is not recommended to reverse-engineer the data model from such a structure as one XML file instance may not contain all the possible elements described in the DTD or XSD, and data model may be incomplete.
re	<Root element>	Name of the element to take as the root table of the schema. This value is case sensitive. This property can be used for reverse-engineering for example a specific message definition from a WSDL file, or when several possible root elements exist in a XSD file.
ro	true | false	If true, the XML file is opened in read only mode.
s	<schema name>	Name of the relational schema where the XML file will be loaded. If this property is missing, a schema named after the five first letters of the XML file name will automatically be created. This schema will be selected when creating the physical schema under the XML data server.

Table 5–1 (Cont.) JDBC Driver Properties

Property	Value	Notes
cs	true | false	Load the XML file in case sensitive or insensitive mode. For case insensitive mode, all element names in the DTD file should be distinct (For example: Abc and abc in the same file will result in name collisions).

The following examples illustrate these properties:

Connects to the `PROD20100125_001.xml` file described by `products.xsd` in the PRODUCTS schema.

```
jdbc:snps:xml?f=/xml/PROD20100125_001.xml&d=/xml/products.xsd&s=PRODUCTS
```

Connects in read-only mode to the `staff_internal.xml` file described by `staff_internal.dtd` in read-only mode. The schema name will be `staff`.

```
jdbc:snps:xml?f=/demo/xml/staff_internal.xml&d=/demo/xml/staff_
internal.dtd&ro=true&s=staff
```

5.3.2 Creating a Physical Schema for XML

Create an XML physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The schema name that you have set on the URL will be preset. Select this schema for both the Data Schema and Work Schema.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

5.4 Setting Up an Integration Project

Setting up a Project using the XML database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The recommended knowledge modules to import into your project for getting started with XML are the following:

- LKM SQL to SQL
- LKM File to SQL
- IKM XML Control Append

5.5 Creating and Reverse-Engineering a XML File

This section contains the following topics:

- Create an XML Model
- Reverse-Engineering an XML Model

5.5.1 Create an XML Model

An XML file model groups a set of datastores. Each datastore typically represents an element in the XML file.

Create an XML Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. Select the XML technology and the XML logical schema created when configuring the topology.

5.5.2 Reverse-Engineering an XML Model

XML supports standard reverse-engineering, which uses only the abilities of the XML driver.

It is recommended to reference a DTD or XSD file in the dtd or d parameters of the URL to reverse-engineer the structure from a generic description of the XML file structure. Reverse-engineering can use an XML instance file if no XSD or DTD is available. In this case, the relational schema structure will be inferred from the data contained in the XML file.

Standard Reverse-Engineering

To perform a Standard Reverse- Engineering on XML use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The standard reverse-engineering process will automatically reverse-engineer the table from the relational schema generated by the XML driver. Note that these tables automatically include:

- Primary keys (PK columns) to preserve parent-child elements relationships
- Foreign keys (FK columns) to preserve parent-child elements relationships
- Order identifier (ORDER columns) to preserve the order of elements in the XML file

These extra columns enable the mapping of the hierarchical XML structure into the relational schema. See XML to SQL Mapping in the Appendix B, "Oracle Data Integrator Driver for XML Reference" for more information.

5.6 Designing an Interface

You can use XML as a source or a target of an integration interface.

The KM choice for an interface or a check determines the abilities and performances of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning an XML data server.

5.6.1 Notes about XML Interfaces

Read carefully these notes before working with XML in integration interfaces.

5.6.1.1 Targeting an XML Structure

When using a datastore of an XML model as a target of an interface, you must make sure to load the driver-generated columns that are used for preserving the parent-child relationships and the order in the XML hierarchy. For example, if filling records for the `region` element into an XML structure as shown in Example 5–1, that correspond to a REGION table in the relational schema, you should load the columns

REGION_ID and REGION_NAME of the REGION table. These two columns correspond to XML attributes.

Example 5–1 XML Structure

```
<country COUNTRY_ID="6" COUNTRY_NAME="Australia">
    <region REGION_ID="72" REGION_NAME="Queensland">
</country>
```

In Example 5–1 you must also load the following additional columns that are automatically created by the XML Driver in the REGION table:

- REGIONPK: This column enables you to identify each `<region>` element.
- REGIONORDER: This column enables you to order the `<region>` elements in the XML file (records are not ordered in a relational schema, whereas XML elements are ordered).
- COUNTRYFK: This columns enables you to put the `<region>` element in relation with the `<country>` parent element. This value is equal to the COUNTRY.COUNTRYPK value for the *Australia* record in the COUNTRY table.

5.6.1.2 Synchronizing XML File and Schema

To ensure a perfect synchronization of the data in an XML file and the data in the XML schema, the following commands have to be called:

- Before using the tables of an XML model, either to read or update data, it is recommended that you use the `SYNCHRONIZE FROM FILE` command on the XML logical schema. This operation reloads the XML hierarchical data in the relational XML schema. The schema is loaded in the built-in or external database storage when first accessed. Subsequent changes made to the file are not automatically synchronized into the schema unless you issue this command.
- After performing changes in the relational schema, you must unload this schema into the XML hierarchical data by calling the `SYNCHRONIZE ALL` or `SYNCHRONIZE FROM DATABASE` commands on the XML Logical Schema. The IKM XML Control Append implements this synchronize command.

These commands must be executed in procedures in the packages before (and after) the interfaces and procedures manipulating the XML schema.

See Appendix B, "Oracle Data Integrator Driver for XML Reference" for more information on these commands.

5.6.1.3 Handling Large XML Files

Large XML files can be handled with high performance with Oracle Data Integrator.

The default driver configuration stores the relational schema in a *built-in engine* in memory. It is recommended to consider the use of *external database* storage for handling large XML files.

See Section B.2.3.1, "Schema Storage" for more information on these commands.

5.6.2 Loading Data from and to XML

An XML file can be used as an interface's source or target. The LKM choice in the Interface Flow tab that is used to load data between XML files and other types of data servers is essential for the performance of the interface.

5.6.2.1 Loading Data from an XML Schema

Use the Generic SQL KMs or the KMs specific to the other technology involved to load data from an XML database to a target or staging area database.

Table 5–2 lists some examples of KMs that you can use to load from an XML source to a staging area:

Table 5–2 KMs to Load from XML to a Staging Area

Staging Area	KM	Notes
Microsoft SQL Server	LKM SQL to MSSQL (BULK)	Uses SQL Server's bulk loader.
Oracle	LKM SQL to Oracle	Faster than the Generic LKM (Uses Statistics)
Sybase	LKM SQL to Sybase ASE (BCP)	Uses Sybase's bulk loader.
All	LKM SQL to SQL	Generic KM to load data between an ANSI SQL-92 source and an ANSI SQL-92 staging area.

5.6.2.2 Loading Data to an XML Schema

It is not advised to use an XML schema as a staging area, except if XML is the target of the interface and you wish to use the target as a staging area. In this case, it might be required to load data to an XML schema.

Use the Generic SQL KMs or the KMs specific to the other technology involved to load data from a source or staging area into an XML schema.

Table 5–3 lists some examples of KMs that you can use to load from a source to an XML staging area.

Table 5–3 KMs to Load to an XML Schema

Source	KM	Notes
File	LKM File to SQL	Generic KM to load a file in a ANSI SQL-92 staging area.
All	LKM SQL to SQL	Generic KM to load data between an ANSI SQL-92 source and an ANSI SQL-92 staging area.

5.6.3 Integrating Data in XML

XML can be used as a target of an interface. The data integration strategies in XML concern loading from the staging area to XML. The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

The IKM XML Control Append integrates data into the XML schema and has an option to synchronize the data to the file. In addition to this KM, you can also use the Generic SQL KMs or the KMs specific to the other technology involved. Note that if using generic or technology-specific KMs, you must manually perform the synchronize operation to write the changes made in the schema to the XML file.

Table 5–4 lists some examples of KMs that you can use to integrate data:

- From a staging area to an XML target
- From an XML staging area to an XML target. Note that in this case the staging area is on the XML target.

Table 5–4 KMs to Integrate Data in an XML File

Mode	Staging Area	KM	Notes
Update	XML	IKM SQL Incremental Update	Generic KM
Append	XML	IKM SQL Control Append	Generic KM
Append	All RDBMS	IKM SQL to SQL Append	Generic KM

5.7 Troubleshooting

This section provides information on how to troubleshoot problems that you might encounter when using XML in Oracle Data Integrator. It contains the following topics:

- Detect the Errors Coming from XML
- Common Errors

5.7.1 Detect the Errors Coming from XML

Errors appear often in Oracle Data Integrator in the following way:

```
java.sql.SQLException: No suitable driver
at ...
at ...
...
```

the `java.sql.SQLExceptioncode` simply indicates that a query was made through the JDBC driver, which has returned an error. This error is frequently a database or driver error, and must be interpreted in this direction.

Only the part of text in bold must first be taken in account. It must be searched in the XML driver documentation. If it contains a specific error code, like here, the error can be immediately identified.

If such an error is identified in the execution log, it is necessary to analyze the SQL code send to the database to find the source of the error. The code is displayed in the description tab of the task in error.

5.7.2 Common Errors

This section describes the most common errors with XML along with the principal causes. It contains the following topics:

- `No suitable driver`

  The JDBC URL is incorrect. Check that the URL syntax is valid.

- `File <XML file> is already locked by another instance of the XML driver.`

  The XML file is locked by another user/application. Close all application that might be using the XML file. If such an application has crashed, then remove the .lck file remaining in the XML file's directory.

- `The DTD file "xxxxxxx.dtd" doesn't exist`

  This exception may occur when trying to load an XML file by the command LOAD FILE. The error message can have two causes:

  - The path of the DTD file is incorrect.

- The corresponding XML file was already opened by another schema (during connection for instance).

- `Table not found: S0002 Table not found: <table name> in statement [<SQL statement>]`

  The table you are trying to access does not exist in the schema.

- `Column not found: S0022 Column not found: <column name> in statement [<SQL statement>]`

  The column you are trying to access does not exist in the tables specified in the statement.

6

Microsoft SQL Server

This chapter describes how to work with Microsoft SQL Server in Oracle Data Integrator.

This chapter includes the following sections:

6.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in Microsoft SQL Server. Oracle Data Integrator features are designed to work best with Microsoft SQL Server, including reverse-engineering, changed data capture, data integrity check, and integration interfaces.

6.1.1 Concepts

The Microsoft SQL Server concepts map the Oracle Data Integrator concepts as follows: A Microsoft SQL Server server corresponds to a data server in Oracle Data Integrator. Within this server, a database/owner pair maps to an Oracle Data Integrator physical schema. A set of related objects within one database corresponds to a data model, and each table, view or synonym will appear as an ODI datastore, with its attributes, columns and constraints.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to Microsoft SQL Server.

6.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 6–1 for handling Microsoft SQL Server data. In addition to these specific Microsoft SQL Server Knowledge Modules, it is also possible to use the generic SQL KMs with Microsoft SQL Server. See Chapter 4, "Generic SQL" for more information.

Table 6–1 Microsoft SQL Server Knowledge Modules

Knowledge Module	Description
IKM MSSQL Incremental Update	Integrates data in a Microsoft SQL Server target table in incremental update mode.
IKM MSSQL Slowly Changing Dimension	Integrates data in a Microsoft SQL Server target table used as a Type II Slowly Changing Dimension in your Data Warehouse.
JKM MSSQL Consistent	Creates the journalizing infrastructure for consistent journalizing on Microsoft SQL Server tables using triggers.
JKM MSSQL Simple	Creates the journalizing infrastructure for simple journalizing on Microsoft SQL Server tables using triggers.
LKM File to MSSQL (BULK)	Loads data from a File to a Microsoft SQL Server staging area database using the BULK INSERT SQL command.
LKM MSSQL to MSSQL (BCP)	Loads data from a Microsoft SQL Server source database to a Microsoft SQL Server staging area database using the native BCP out/BCP in commands.
LKM MSSQL to MSSQL (LINKED SERVERS)	Loads data from a Microsoft SQL Server source database to a Microsoft SQL Server staging area database using the native linked servers feature.
LKM MSSQL to ORACLE (BCP SQLLDR)	Loads data from a Microsoft SQL Server to an Oracle database (staging area) using the BCP and SQLLDR utilities.
LKM SQL to MSSQL (BULK)	Loads data from any ANSI SQL-92 source database to a Microsoft SQL Server staging area database using the native BULK INSERT SQL command.
LKM SQL to MSSQL	Loads data from any ANSI SQL-92 source database to a Microsoft SQL Server staging area. This LKM is similar to the standard LKM SQL to SQL described in Chapter 4, "Generic SQL" except that you can specify some additional specific Microsoft SQL Server parameters.
RKM MSSQL	Retrieves metadata for Microsoft SQL Server objects: tables, views and synonyms, as well as columns and constraints.

6.2 Installation and Configuration

Make sure you have read the information in this section before you start working with the Microsoft SQL Server technology:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

6.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

6.2.2 Technology Specific Requirements

Some of the Knowledge Modules for Microsoft SQL Server use specific features of this database. The following restrictions apply when using these Knowledge Modules. See the Microsoft SQL Server documentation for additional information on these topics.

6.2.2.1 Using the BULK INSERT Command

This section describes the requirements that must be met before using the BULK INSERT command with Microsoft SQL Server:

- The file to be loaded by the BULK INSERT command needs to be accessible from the Microsoft SQL Server instance machine. It could be located on the file system of the server or reachable from a UNC (Unique Naming Convention) path.
- UNC file paths are supported but not recommended as they may decrease performance.
- For performance reasons, it is often recommended to install Oracle Data Integrator Agent on the target server machine.

6.2.2.2 Using the BCP Command

This section describes the requirements that must be met before using the BCP command with Microsoft SQL Server:

- The BCP utility as well as the Microsoft SQL Server Client Network Utility must be installed on the machine running the Oracle Data Integrator Agent.
- The server names defined in the Topology must match the Microsoft SQL Server Client connect strings used for these servers.
- White spaces in server names defined in the Client Utility are not supported.
- UNC file paths are supported but not recommended as they may decrease performance.
- The target staging area database must have the option *select into/bulk copy*.
- Execution can remain pending if the file generated by the BCP program is empty.
- For performance reasons, it is often recommended to install Oracle Data Integrator Agent on the target server machine.

6.2.2.3 Using Linked Servers

This section describes the requirements that must be met before using linked servers with Microsoft SQL Server:

- The user defined in the Topology to connect to the Microsoft SQL Server instances must have the following privileges:
  - The user must be the db_owner of the staging area databases
  - The user must have db_ddladmin role
  - For automatic link server creation, the user must have sysdamin privileges
- The MSDTC Service must be started on both SQL Server instances (source and target). The following hints may help you configure this service:
  - The Log On As account for the MSDTC Service is a Network Service account (and not the 'LocalSystem' account).
  - MSDTC should be enabled for network transactions.

- Windows Firewall should be configured to allow the MSDTC service on the network. By default, the Windows Firewall blocks the MSDTC program.
- The Microsoft SQL Server must be started after MSDTC has completed its startup.

See the following links for more information about configuring the MSDTC Service:

- http://support.microsoft.com/?kbid=816701
- http://support.microsoft.com/?kbid=839279

6.2.3 Connectivity Requirements

This section lists the requirements for connecting to a Microsoft SQL Server database.

JDBC Driver

Oracle Data Integrator is installed with a default Microsoft SQL Server Datadirect Driver. This drivers directly uses the TCP/IP network layer and requires no other installed component or configuration. You can alternatively use the drivers provided by Microsoft for SQL Server.

6.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating a Microsoft SQL Server Data Server
2. Creating a Microsoft SQL Server Physical Schema

6.3.1 Creating a Microsoft SQL Server Data Server

A Microsoft SQL Server data server corresponds to a Microsoft SQL Server server connected with a specific user account. This user will have access to several databases in this server, corresponding to the physical schemas in Oracle Data Integrator created under the data server.

6.3.1.1 Creation of the Data Server

Create a data server for the Microsoft SQL Server technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Microsoft SQL data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator
   - **Server**: Physical name of the data server
   - **User/Password**: Microsoft SQLServer user with its password
2. In the JDBC tab:
   - **JDBC Driver**: `weblogic.jdbc.sqlserver.SQLServerDriver`
   - **JDBC URL**: jdbc:weblogic:sqlserver://hostname:port[;property=value[;...]]

6.3.2 Creating a Microsoft SQL Server Physical Schema

Create a Microsoft SQL Server physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The work schema and data schema in this physical schema correspond each to a database/owner pair. The work schema should point to a temporary database and the data schema should point to the database hosting the data to integrate.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

6.4 Setting Up an Integration Project

Setting up a project using the Microsoft SQL Server database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with Microsoft SQL Server:

- IKM MSSQL Incremental Update
- IKM MSSQL Slowly Changing Dimension
- JKM MSSQL Consistent
- JKM MSSQL Simple
- LKM File to MSSQL (BULK)
- LKM MSSQL to MSSQL (BCP)
- LKM MSSQL to MSSQL (LINKED SERVERS)
- LKM MSSQL to ORACLE (BCP SQLLDR)
- LKM SQL to MSSQL (BULK)
- LKM SQL to MSSQL
- CKM SQL. This generic KM is used for performing integrity check for SQL Server.
- RKM MSSQL

6.5 Creating and Reverse-Engineering a Microsoft SQL Server Model

This section contains the following topics:

6.5.1 Create a Microsoft SQL Server Model

Create a Microsoft SQL Server Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

6.5.2 Reverse-engineer a Microsoft SQL Server Model

Microsoft SQL Server supports both Standard reverse-engineering - which uses only the abilities of the JDBC driver - and Customized reverse-engineering, which uses a RKM to retrieve the metadata.

In most of the cases, consider using the standard JDBC reverse engineering for starting. Standard reverse-engineering with Microsoft SQL Server retrieves tables, views, and columns.

Consider switching to customized reverse-engineering for retrieving more metadata. Microsoft SQL Server customized reverse-engineering retrieves the tables, views, and synonyms. The RKM MSSQL also reverse-engineers columns that have a user defined data type and translates the user defined data type to the native data type.

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on Microsoft SQL Server use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on Microsoft SQL Server with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Microsoft SQL Server technology:

1. In the Reverse tab of the Microsoft SQL Server Model, select the `KM: RKM MSSQL.<project name>`.
2. In the COMPATIBLE option, enter the Microsoft SQL Server version. This option decides whether to enable reverse synonyms. Note that only Microsoft SQLServer version 2005 and above support synonyms.

Note the following information when using this RKM:

- The connection user must have SELECT privileges on any INFORMATION_ SCHEMA views.
- Only native data type will be saved for the column with user defined data type in the repository and model.
- User defined data types implemented through a class of assembly in the Microsoft .NET Framework common language runtime (CLR) will not be reversed.

6.6 Setting up Changed Data Capture

The ODI Microsoft SQL Server Knowledge Modules support the Changed Data Capture feature. See Chapter "Working with Changed Data Capture" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details on how to set up journalizing and how to use captured changes.

Microsoft SQL Server Journalizing Knowledge Modules support Simple Journalizing and Consistent Set Journalizing. The Microsoft SQL Server JKMs use triggers to capture data changes on the source tables.

Oracle Data Integrator provides the Knowledge Modules listed in Table 6–2 for journalizing Microsoft SQL Server tables.

Table 6–2 Microsoft SQL Server Journalizing Knowledge Modules

KM	Notes
JKM MSSQL Consistent	Creates the journalizing infrastructure for consistent journalizing on Microsoft SQL Server tables using triggers.
JKM MSSQL Simple	Creates the journalizing infrastructure for simple journalizing on Microsoft SQL Server tables using triggers.

Log-based changed data capture is possible with Microsoft SQL Server using the Oracle Changed Data Capture Adapters. See Chapter 25, "Oracle Changed Data Capture Adapters" for more information.

6.7 Setting up Data Quality

Oracle Data Integrator provides the generic CKM SQL for checking data integrity against constraints defined on a Microsoft SQL Server table. See "Set up Flow Control and Post-Integration Control" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details.

See Chapter 4, "Generic SQL" for more information.

6.8 Designing an Interface

You can use Microsoft SQL Server as a source, staging area or a target of an integration interface.

The KM choice for an interface or a check determines the abilities and performance of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning a Microsoft SQL Server data server.

6.8.1 Loading Data from and to Microsoft SQL Server

Microsoft SQL Server can be used as a source, target or staging area of an interface. The LKM choice in the Interface Flow tab to load data between Microsoft SQL Server and another type of data server is essential for the performance of an interface.

6.8.1.1 Loading Data from Microsoft SQL Server

Oracle Data Integrator provides Knowledge Modules that implement optimized methods for loading data from Microsoft SQL Server to a target or staging area database. These optimized Microsoft SQL Server KMs are listed in Table 6–3.

In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technology involved to load data from Microsoft SQL Server to a target or staging area database.

Table 6–3 KMs for loading data from Microsoft SQL Server

Source or Staging Area Technology	KM	Notes
Microsoft SQL Server	LKM MSSQL to MSSQL (BCP)	Loads data from a Microsoft SQL Server source database to a Microsoft SQL Server staging area database using the native BCP out/BCP in commands.
Microsoft SQL Server	LKM MSSQL to MSSQL (LINKED SERVERS)	Loads data from a Microsoft SQL Server source database to a Microsoft SQL Server staging area database using the native linked servers feature.
Oracle	LKM MSSQL to ORACLE (BCP SQLLDR)	Loads data from a Microsoft SQL Server to an Oracle database (staging area) using the BCP and SQLLDR utilities.

6.8.1.2 Loading Data to Microsoft SQL Server

Oracle Data Integrator provides Knowledge Modules that implement optimized methods for loading data from a source or staging area into a Microsoft SQL Server database. These optimized Microsoft SQL Server KMs are listed in Table 6–4.

In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technology involved.

Table 6–4 KMs for loading data to Microsoft SQL Server

Source or Staging Area Technology	KM	Notes
File	LKM File to MSSQL (BULK)	Loads data from a File to a Microsoft SQL Server staging area database using the BULK INSERT SQL command.
Microsoft SQL Server	LKM MSSQL to MSSQL (BCP)	Loads data from a Microsoft SQL Server source database to a Microsoft SQL Server staging area database using the native BCP out/BCP in commands.
Microsoft SQL Server	LKM MSSQL to MSSQL (LINKED SERVERS)	Loads data from a Microsoft SQL Server source database to a Microsoft SQL Server staging area database using the native linked servers feature.

Table 6–4 (Cont.) KMs for loading data to Microsoft SQL Server

Source or Staging Area Technology	KM	Notes
SQL	LKM SQL to MSSQL (BULK)	Loads data from any ANSI SQL-92 source database to a Microsoft SQL Server staging area database using the native BULK INSERT SQL command.
SQL	LKM SQL to MSSQL	Loads data from any ANSI SQL-92 source database to a Microsoft SQL Server staging area.

6.8.2 Integrating Data in Microsoft SQL Server

Oracle Data Integrator provides Knowledge Modules that implement optimized data integration strategies for Microsoft SQL Server. These optimized Microsoft SQL Server KMs are listed in Table 6–5. I

In addition to these KMs, you can also use the Generic SQL KMs.

The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

Table 6–5 KMs for integrating data to Microsoft SQL Server

KM	Notes
IKM MSSQL Incremental Update	Integrates data in a Microsoft SQL Server target table in incremental update mode.
IKM MSSQL Slowly Changing Dimension	Integrates data in a Microsoft SQL Server target table used as a Type II Slowly Changing Dimension in your Data Warehouse

Using Slowly Changing Dimensions

For using slowly changing dimensions, make sure to set the *Slowly Changing Dimension* value for each column of the target datastore. This value is used by the IKM MSSQL Slowly Changing Dimension to identify the Surrogate Key, Natural Key, Overwrite or Insert Column, Current Record Flag and Start/End Timestamps columns.

7

Microsoft Excel

This chapter describes how to work with Microsoft Excel in Oracle Data Integrator.

This chapter includes the following sections:

7.1 Introduction

Oracle Data Integrator (ODI) integrates data stored into Microsoft Excel workbooks. It allows reverse-engineering as well as read and write operations on spreadsheets.

Oracle Data Integrator uses Open Database Connectivity (ODBC) to connect to a Microsoft Excel data server. See Section 7.2.3, "Connectivity Requirements" for more details.

7.1.1 Concepts

A Microsoft Excel data server corresponds to one Microsoft Excel workbook (`.xls` file) that is accessible through your local network. A single physical schema is created under this data server.

Within this schema, a spreadsheet or a given named zone of the workbook appears as a datastore in Oracle Data Integrator.

7.1.2 Knowledge Modules

Oracle Data Integrator provides no Knowledge Module (KM) specific to the Microsoft Excel technology. You can use the generic SQL KMs to perform the data integration and transformation operations of Microsoft Excel data. See Chapter 4, "Generic SQL" for more information.

Note: Excel technology cannot be used as the staging area, does not support incremental update or flow/static check. As a consequence, the following KMs will not work with the Excel technology:

- RKM SQL (JYTHON)
- LKM File to SQL
- CKM SQL
- IKM SQL Incremental Update
- IKM SQL Control Append
- LKM SQL to SQL (JYTHON)

7.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Microsoft Excel Knowledge Module:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

7.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

7.2.2 Technology Specific Requirements

There are no technology-specific requirements for using Microsoft Excel files in Oracle Data Integrator.

7.2.3 Connectivity Requirements

This section lists the requirements for connecting to a Microsoft Excel workbook.

To be able to access Microsoft Excel data, you need to:

- Install the Microsoft Excel ODBC Driver
- Declare a Microsoft Excel ODBC Data Source

Install the Microsoft Excel ODBC Driver

Microsoft Excel workbooks can only be accessed through ODBC connectivity. The ODBC Driver for Excel must be installed on your system.

Declare a Microsoft Excel ODBC Data Source

An ODBC data source must be defined for each Microsoft Excel workbook (`.xls` file) that will be accessed from ODI. ODBC datasources are created with the Microsoft ODBC Data Source Administrator. Refer to your Microsoft Windows operating system documentation for more information on datasource creation.

7.3 Setting up the Topology

Setting up the Topology consists in:

1. Creating a Microsoft Excel Data Server
2. Creating a Microsoft Excel Physical Schema

7.3.1 Creating a Microsoft Excel Data Server

A Microsoft Excel data server corresponds to one Microsoft Excel workbook (`.xls` file) that is accessible through your local network.

Create a data server for the Microsoft Excel technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Microsoft Excel Data Server:

1. In the Definition tab:
   - **Array Fetch Size**: `1`
   - **Batch Update Size**: `1`
2. In the JDBC tab:
   - **JDBC Driver**: `sun.jdbc.odbc.JdbcOdbcDriver`
   - **JDBC URL**: `jdbc:odbc:<odbc_dsn_alias>`

     where <odbc_dsn_alias> is the name of your ODBC data source.

WARNING: To access a Microsoft Excel workbook via ODBC, you must first ensure that this workbook is not currently open in a Microsoft Excel session. This can lead to unexpected results.

7.3.2 Creating a Microsoft Excel Physical Schema

Create a Microsoft Excel Physical Schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Note that Oracle Data Integrator needs only one physical schema for each Microsoft Excel data server. If you wish to connect a different workbook, a different data server must be created to connect a ODBC datasource corresponding to this other workbook.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator and associate it in a given context.

Note: An Excel physical schema only has a data schema, and no work schema. Microsoft Excel cannot be used as the staging area of an interface.

7.4 Setting Up an Integration Project

Setting up a Project using the Microsoft Excel follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Import the following generic SQL KMs into your project for getting started with Microsoft Excel:

- LKM SQL to SQL
- IKM SQL to SQL Append

See Chapter 4, "Generic SQL" for more information about these KMs.

7.5 Creating and Reverse-Engineering a Microsoft Excel Model

This section contains the following topics:

- Create a Microsoft Excel Model
- Reverse-engineer a Microsoft Excel Model

7.5.1 Create a Microsoft Excel Model

A Microsoft Excel Model is a set of datastores that correspond to the tables contained in a Microsoft Excel workbook.

Create a Microsoft Excel Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

7.5.2 Reverse-engineer a Microsoft Excel Model

Microsoft Excel supports only the Standard reverse-engineering, which uses only the abilities of the ODBC driver.

Oracle Data Integrator reverse-engineers:

- *Spreadsheets*: Spreadsheets appear as *system tables*. Such a table is named after the spreadsheet name, followed with a dollar sign ($). This table's columns are named after the first line of the spreadsheet. Note that new records are added at the end of the spreadsheet.
- *Named Cell Ranges* in a spreadsheet. These will appear as *tables* named after the cell range name. Depending on the scope of a name, the table name may be prefixed by the name of the spreadsheet (in the following format: `<spreadsheet_name>$<zone_name>`). The columns for such a table are named after the first line of the cell range. Note that new records are added automatically below the named cell. It is possible to create a blank named cell range that will be loaded using ODI by naming a cell range that contains only the first header line.

  In most Microsoft Excel versions, you can simply select a cell range and use the **Name a Range...** popup menu to name this range. See the Microsoft Excel documentation for conceptual information about Names and how to define a cell range in a spreadsheet.

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on Microsoft Excel use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Note: On the Reverse Engineer tab of your Model, select in the Types of objects to reverse-engineer section **Table** and **System Table** to reverse-engineer spreadsheets and named cell ranges.

7.6 Designing an Interface

You can use a Microsoft Excel file as a source or a target of an integration interface, but **NOT** as the staging area

The KM choice for an interface or a check determines the abilities and performances of this interface or check. The recommendations below help in the selection of the KM for different situations concerning a Microsoft Excel server.

7.6.1 Loading Data From and to Microsoft Excel

Microsoft Excel can be used as a source or a target of an interface. The LKM choice in the Interface Flow tab to load data between Microsoft Excel and another type of data server is essential for the performance of an interface.

7.6.1.1 Loading Data from Microsoft Excel

Oracle Data Integrator does not provide specific knowledge modules for Microsoft Excel. Use the Generic SQL KMs or the KMs specific to the technology used as the staging area. The following table lists some generic SQL KMs that can be used for loading data from Microsoft Excel to any staging area.

Table 7–1 KMs to Load from Microsoft Excel

Target or Staging Area	KM	Notes
Oracle	LKM SQL to Oracle	Loads data from any ISO-92 database to an Oracle target database. Uses statistics.
SQL	LKM SQL to SQL	Loads data from any ISO-92 database to any ISO-92 compliant target database.
Sybase	LKM SQL to Sybase (bcp)	Loads data from any ISO-92 compliant database to a Sybase ASE Server database. Uses Bulk Loading.
Microsoft SQL Server	LKM SQL to MSSQL (bulk)	Loads data from any ISO-92 database to a Microsoft SQL Server target database. Uses Bulk Loading.

7.6.1.2 Loading Data to Microsoft Excel

Because Microsoft Excel cannot be used as staging area you cannot use a LKM to load data into Microsoft Excel. See Section 7.6.2, "Integrating Data in Microsoft Excel" for more information on how to integrate data into Microsoft Excel.

7.6.2 Integrating Data in Microsoft Excel

Oracle Data Integrator does not provide specific knowledge modules for Microsoft Excel. Use the Generic SQL KMs or the KMs specific to the technology used as the staging area. For integrating data from a staging area to Microsoft Excel, you can use, for example the IKM SQL to SQL Append.

7.7 Troubleshooting

This section provides information on how to troubleshoot problems that you might encounter when using the Microsoft Excel technology in Oracle Data Integrator. It contains the following topics:

- Decoding Error Messages
- Common Problems and Solutions

7.7.1 Decoding Error Messages

Errors appear often in Oracle Data Integrator in the following way:

```
java.sql.SQLException: java.sql.SQLException: [Microsoft][ODBC Driver Manager]
Data source name not found and no default driver specified RC=0xb
at ... ...
```

the java.sql.SQLException code simply indicates that a query was made through the JDBC-ODBC bridge, which has returned an error. This error is frequently a database or driver error, and must be interpreted in this direction.

Only the part of text in *italic* must first be taken in account. It must be searched in the ODBC driver or Excel documentation. If its contains a specific error code, like here in ***bold italic***, the error can be immediately identified.

If such an error is identified in the execution log, it is necessary to analyze the SQL code to find the source of the error. The code is displayed in the description tab of the task in error.

The most common errors with Excel are detailed below, with their principal causes.

7.7.2 Common Problems and Solutions

This section describes common problems and solutions.

- `UnknownDriverException`

  The JDBC driver is incorrect. Check the name of the driver.

- `[Microsoft][ODBC Driver Manager] Data source name not found and no default driver specified RC=0xb Datasource not found or driver name not specified`

  The ODBC Datasource specified in the JDBC URL is incorrect.

- `The Microsoft Jet Database engine could not find the object <object name>`

  The table you are trying to access does not exist or is not defined in the Excel spreadsheet.

- `Too few parameters. Expected 1.`

  You are trying to access an nonexisting column in the Excel spreadsheet.

- `Operation must use an updateable query.`

  This error is probably due to the fact that you have not unchecked the "read only" option when defined the Excel DSN. Unselect this option and re-execute your interface.

- DBCS or UTF-16 data is corrupted when loaded.

  This error is due to the fact that the JDBC-ODBC Bridge of the Java machine does not support UTF-16 data. This is a known issue in the Sun JVM that is solved in the later releases (1.7).

8

Microsoft Access

This chapter describes how to work with Microsoft Access in Oracle Data Integrator.

This chapter includes the following sections:

8.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in a Microsoft Access database. Oracle Data Integrator features are designed to work best with Microsoft Access, including integration interfaces.

8.2 Concepts

The Microsoft Access concepts map the Oracle Data Integrator concepts as follows: An Microsoft Access database corresponds to a data server in Oracle Data Integrator. Within this server, a schema maps to an Oracle Data Integrator physical schema.

Oracle Data Integrator uses Oracle Data Integrator uses Open Database Connectivity (ODBC) to connect to connect to a Microsoft Access database.

8.3 Knowledge Modules

Oracle Data Integrator provides the IKM Access Incremental Update for handling Microsoft Access data. This IKM integrates data in a Microsoft Access target table in incremental update mode.

The IKM Access Incremental Update creates a temporary staging table to stage the data flow and compares its content to the target table to identify the records to insert and the records to update. It also allows performing data integrity check by invoking the CKM.

Consider using this KM if you plan to load your Microsoft Access target table to insert missing records and to update existing ones.

To use this IKM, the staging area must be on the same data server as the target.

This KM uses Microsoft Access specific features. It is also possible to use the generic SQL KMs with the Microsoft Access database. See Chapter 4, "Generic SQL" for more information.

8.4 Specific Requirements

There are no specific requirements for using Microsoft Access in Oracle Data Integrator.

9

Netezza

This chapter describes how to work with Netezza in Oracle Data Integrator.

This chapter includes the following sections:

9.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in a Netezza database. Oracle Data Integrator features are designed to work best with Netezza, including reverse-engineering, data integrity check, and integration interfaces.

9.1.1 Concepts

The Netezza database concepts map the Oracle Data Integrator concepts as follows: A Netezza cluster corresponds to a data server in Oracle Data Integrator. Within this server, a database/owner pair maps to an Oracle Data Integrator physical schema.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to a Netezza database.

9.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 9–1 for handling Netezza data. These KMs use Netezza specific features. It is also possible to use the generic SQL KMs with the Netezza database. See Chapter 4, "Generic SQL" for more information.

Table 9–1 Netezza Knowledge Modules

Knowledge Module	Description
CKM Netezza	Checks data integrity against constraints defined on a Netezza table. Rejects invalid records in the error table created dynamically. Can be used for static controls as well as flow controls.

Table 9–1 (Cont.) Netezza Knowledge Modules

Knowledge Module	Description
IKM Netezza Control Append	Integrates data in a Netezza target table in replace/append mode. When flow data needs to be checked using a CKM, this IKM creates a temporary staging table before invoking the CKM.
IKM Netezza Incremental Update	Integrates data in a Netezza target table in incremental update mode.
IKM Netezza To File (EXTERNAL TABLE)	Integrates data in a target file from a Netezza staging area. It uses the native EXTERNAL TABLE feature of Netezza.
LKM File to Netezza (EXTERNAL TABLE)	Loads data from a File to a Netezza Server staging area using the EXTERNAL TABLE feature (dataobject).
LKM File to Netezza (NZLOAD)	Loads data from a File to a Netezza Server staging area using NZLOAD.
RKM Netezza	Retrieves JDBC metadata from a Netezza database. This RKM may be used to specify your own strategy to convert Netezza JDBC metadata into Oracle Data Integrator metadata. Consider using this RKM if you encounter problems with the standard JDBC reverse-engineering process due to some specificities of the Netezza JDBC driver.

9.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Netezza Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

9.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

9.2.2 Technology Specific Requirements

Some of the Knowledge Modules for Netezza use the NZLOAD utility.

The following requirements and restrictions apply for these Knowledge Modules:

- The source file must be accessible by the ODI agent executing the interface.
- The run-time agent machine must have Netezza Performance Server client installed. And the NZLOAD install directory needs to be in the PATH variable when the agent is started.
- All mappings need to be on the staging area.
- All source fields need to be mapped, and must be in the same order as the target table in Netezza.

- Date, Time, Timestamp and Numeric formats should be specified in consistent with Netezza Data Type definition.

For KMs using the EXTERNAL TABLE feature: Make sure that the file is accessible by the Netezza Server.

9.2.3 Connectivity Requirements

This section lists the requirements for connecting to a Netezza database.

JDBC Driver

Oracle Data Integrator uses the Netezza JDBC to connect to a NCR Netezza database. This driver must be installed in your Oracle Data Integrator drivers directory.

9.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating a Netezza Data Server
2. Creating a Netezza Physical Schema

9.3.1 Creating a Netezza Data Server

A Netezza data server corresponds to a Netezza cluster connected with a specific Netezza user account. This user will have access to several databases in this cluster, corresponding to the physical schemas in Oracle Data Integrator created under the data server.

9.3.1.1 Creation of the Data Server

Create a data server for the Netezza technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Netezza data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator
   - **Server**: Physical name of the data server
   - **User/Password**: Netezza user with its password
2. In the JDBC tab:
   - **JDBC Driver**: `org.netezza.Driver`
   - **JDBC URL**: `jdbc:Netezza://<host>:<port>/<database>`

> **Note:** Note that Oracle Data Integrator will have write access only on the database specified in the URL.

9.3.2 Creating a Netezza Physical Schema

Create a Netezza physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Note: When performing this configuration, the work and data databases names must match. Note also that the dollar sign ($) is an invalid character for names in Netezza. Remove the dollar sign ($) from work table and journalizing elements prefixes.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

9.4 Setting Up an Integration Project

Setting up a project using the Netezza database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with Netezza:

- CKM Netezza
- IKM Netezza Control Append
- IKM Netezza Incremental Update
- IKM Netezza To File (EXTERNAL TABLE)
- LKM File to Netezza (EXTERNAL TABLE)
- LKM File to Netezza (NZLOAD)
- RKM Netezza

9.5 Creating and Reverse-Engineering a Netezza Model

This section contains the following topics:

9.5.1 Create a Netezza Model

Create a Netezza Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

9.5.2 Reverse-engineer a Netezza Model

Netezza supports both Standard reverse-engineering - which uses only the abilities of the JDBC driver - and Customized reverse-engineering.

In most of the cases, consider using the standard JDBC reverse engineering for starting.

Consider switching to customized reverse-engineering if you encounter problems with the standard JDBC reverse-engineering process due to some specificities of the Netezza JDBC driver.

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on Netezza use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on Netezza with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Netezza technology:

1. In the Reverse tab of the Netezza Model, select the `KM: RKM Netezza.<project name>`.

The reverse-engineering process returns tables, views, columns, Keys and Foreign Keys.

9.6 Setting up Data Quality

Oracle Data Integrator provides the CKM Netezza for checking data integrity against constraints defined on a Netezza table. See "Set up Flow Control and Post-Integration Control" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details.

9.7 Designing an Interface

You can use Netezza as a source, staging area, or a target of an integration interface.

The KM choice for an interface or a check determines the abilities and performance of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning a Netezza data server.

9.7.1 Loading Data from and to Netezza

Netezza can be used as a source, target or staging area of an interface. The LKM choice in the Interface Flow tab to load data between Netezza and another type of data server is essential for the performance of an interface.

9.7.1.1 Loading Data from Netezza

Use the Generic SQL KMs or the KMs specific to the other technology involved to load data from a Netezza database to a target or staging area database.

For extracting data from a Netezza staging area to a file, use the IKM Netezza to File (EXTERNAL TABLE). See Section 9.7.2, "Integrating Data in Netezza" for more information.

9.7.1.2 Loading Data to Netezza

Oracle Data Integrator provides Knowledge Modules that implement optimized methods for loading data from a source or staging area into a Netezza database. These optimized Netezza KMs are listed in Table 9–2. In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technology involved.

Table 9–2 KMs for loading data to Netezza

Source or Staging Area Technology	KM	Notes
File	LKM File to Netezza (EXTERNAL TABLE)	Loads data from a File to a Netezza staging area database using the Netezza External table feature.
File	LKM File to Netezza (NZLOAD)	Loads data from a File to a Netezza staging area database using the NZLOAD bulk loader.

9.7.2 Integrating Data in Netezza

Oracle Data Integrator provides Knowledge Modules that implement optimized data integration strategies for Netezza. These optimized Netezza KMs are listed in Table 9–3. In addition to these KMs, you can also use the Generic SQL KMs.

The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

Table 9–3 KMs for integrating data to Netezza

KM	Notes
IKM Netezza Control Append	Integrates data in a Netezza target table in replace/append mode.
IKM Netezza Incremental Update	Integrates data in a Netezza target table in incremental update mode. This KM implements a DISTRIBUTE_ON option to define the processing distribution. It is important that the chosen column has a high cardinality (many distinct values) to ensure evenly spread data to allow maximum processing performance. Please follow Netezza's recommendations on choosing a such a column. Valid options are: ■ [PK]: Primary Key of the target table. ■ [UK]: Update key of the interface ■ [RANDOM]: Random distribution ■ <list of column>: a comma separated list of columns If no value is set (empty), no index will be created. This KM also uses an ANALYZE_TARGET option to generate statistics on the target after integration.
IKM Netezza to File (EXTERNAL TABLE)	Integrates data from a Netezza staging area to a file using external tables. This KM implements an optional BASE_TABLE option to specify the name of a table that will be used as a template for the external table.

10 Teradata

This chapter describes how to work with Teradata in Oracle Data Integrator.

This chapter includes the following sections:

10.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in an Teradata database. Oracle Data Integrator features are designed to work best with Teradata, including reverse-engineering, data integrity check, and integration interfaces.

10.1.1 Concepts

The Teradata database concepts map the Oracle Data Integrator concepts as follows: A Teradata server corresponds to a data server in Oracle Data Integrator. Within this server, a database maps to an Oracle Data Integrator physical schema.

Oracle Data Integrator uses Java Database Connectivity (JDBC) and Teradata Utilities to connect to Teradata database.

10.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 10–1 for handling Teradata data. These KMs use Teradata specific features. It is also possible to use the generic SQL KMs with the Teradata database. See Chapter 4, "Generic SQL" for more information.

Table 10–1 Teradata Knowledge Modules

Knowledge Module	Description
CKM Teradata	Checks data integrity against constraints defined on a Teradata table. Rejects invalid records in the error table created dynamically. Can be used for static controls as well as flow controls.
IKM File to Teradata (TTU)	This IKM is designed to leverage the power of the Teradata utilities for loading files directly to the target. See Section 10.8.2, "Support for Teradata Utilities" for more information.
IKM SQL to Teradata (TTU)	Integrates data from a SQL compliant database to a Teradata database target table using Teradata Utilities FastLoad, MultiLoad, TPump or Parallel Transporter. See Section 10.8.2, "Support for Teradata Utilities" for more information.
IKM Teradata Control Append	Integrates data in a Teradata target table in replace/append mode.
IKM Teradata Incremental Update	Integrates data in a Teradata target table in incremental update mode.
IKM Teradata Slowly Changing Dimension	Integrates data in a Teradata target table used as a Type II Slowly Changing Dimension in your Data Warehouse.
IKM Teradata to File (TTU)	Integrates data in a target file from a Teradata staging area in replace mode. See Section 10.8.2, "Support for Teradata Utilities" for more information.
IKM Teradata Multi Statement	Integrates data in Teradata database target table using multi statement requests, managed in one SQL transaction. See Using Multi Statement Requests for more information.
IKM SQL to Teradata Control Append	Integrates data from an ANSI-92 compliant source database into Teradata target table in truncate / insert (append) mode. This IKM is typically used for ETL configurations: source and target tables are on different databases and the interface's staging area is set to the logical schema of the source tables or a third schema.
LKM File to Teradata (TTU)	Loads data from a File to a Teradata staging area database using the Teradata bulk utilities. See Section 10.8.2, "Support for Teradata Utilities" for more information.
LKM SQL to Teradata (TTU)	Loads data from a SQL compliant source database to a Teradata staging area database using a native Teradata bulk utility. See Section 10.8.2, "Support for Teradata Utilities" for more information.
RKM Teradata	Retrieves metadata from the Teradata database using the DBC system views. This RKM supports UNICODE columns.

10.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Teradata Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

10.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

http://www.oracle.com/technology/products/oracle-data-integrator/index.html.

10.2.2 Technology Specific Requirements

Some of the Knowledge Modules for Teradata use the following *Teradata Tools and Utilities (TTU)*:

- FastLoad
- MultiLoad
- Tpump
- FastExport
- Teradata Parallel Transporter

The following requirements and restrictions apply for these Knowledge Modules:

- Teradata Utilities must be installed on the machine running the Oracle Data Integrator Agent.
- The server name of the Teradata Server defined in the Topology must match the Teradata connect string used for this server (without the `COP_n` postfix).
- It is recommended to install the Agent on a separate platform than the target Teradata host. The machine were the Agent is installed should have a very large network bandwidth to the target Teradata server.
- The IKM File to Teradata (TTU) and LKM File to Teradata (TTU) support a File Character Set Encoding option specify the encoding of the files integrated with TTU. If this option is unset, the default TTU charset is used.
  Refer to the "Getting Started: International Character Sets and the Teradata Database" Teradata guide for more information about character set encoding.

See the Teradata documentation for more information.

10.2.3 Connectivity Requirements

This section lists the requirements for connecting to a Teradata Database.

JDBC Driver

Oracle Data Integrator uses the Teradata JDBC Driver to connect to a Teradata Database. The Teradata Gateway for JDBC must be running, and this driver must be installed in your Oracle Data Integrator installation. You can find this driver at:

http://www.teradata.com/DownloadCenter/Group48.aspx

10.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating a Teradata Data Server
2. Creating a Teradata Physical Schema

10.3.1 Creating a Teradata Data Server

A Teradata data server corresponds to a Teradata Database connected with a specific Teradata user account. This user will have access to several databases in this Teradata system, corresponding to the physical schemas in Oracle Data Integrator created under the data server.

10.3.1.1 Creation of the Data Server

Create a data server for the Teradata technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Teradata data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator
   - **Server**: Physical name of the data server
   - **User/Password**: Teradata user with its password
2. In the JDBC tab:
   - **JDBC Driver**: `com.teradata.jdbc.TeraDriver`
   - **JDBC URL**: `jdbc:teradata://<host>:<port>/<server>`

     The URL parameters are:
     - `<host>`: Teradata gateway for JDBC machine network name or IP address.
     - `<port>`: gateway port number (usually 7060)
     - `<server>`: name of the Teradata server to connect

10.3.2 Creating a Teradata Physical Schema

Create a Teradata physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

10.4 Setting Up an Integration Project

Setting up a project using the Teradata database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with Teradata:

- CKM Teradata
- IKM File to Teradata (TTU)
- IKM SQL to Teradata (TTU)
- IKM Teradata Control Append
- IKM Teradata Incremental Update

- IKM Teradata Multi Statement
- IKM Teradata Slowly Changing Dimension
- IKM Teradata to File (TTU)
- IKM SQL to Teradata Control Append
- LKM File to Teradata (TTU)
- LKM SQL to Teradata (TTU)
- RKM Teradata

10.5 Creating and Reverse-Engineering a Teradata Model

This section contains the following topics:

10.5.1 Create a Teradata Model

Create a Teradata Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

10.5.2 Reverse-engineer a Teradata Model

Teradata supports both Standard reverse-engineering - which uses only the abilities of the JDBC driver - and Customized reverse-engineering, which uses a RKM to retrieve the metadata from Teradata database using the DBC system views.

In most of the cases, consider using the standard JDBC reverse engineering for starting. Standard reverse-engineering with Teradata retrieves tables and columns.

Preferably use customized reverse-engineering for retrieving more metadata. Teradata customized reverse-engineering retrieves the tables, views, columns, keys (primary indexes and secondary indexes) and foreign keys. Descriptive information (column titles and short descriptions) are also reverse-engineered.

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on Teradata use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on Teradata with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Teradata technology:

1. In the Reverse tab of the Teradata Model, select the `KM: RKM Teradata.<project name>`.
2. Set the REVERSE_FKS option to `true`, if you want to reverse-engineer existing FK constraints in the database.
3. Set the REVERSE_TABLE_CONSTRAINTS to true if you want to reverse-engineer table constrains.

The reverse-engineering process returns tables, views, columns, Keys (primary indexes and secondary indexes) and Foreign Keys. Descriptive information (Column titles and short descriptions) are also reverse-engineered

Note that Unique Indexes are reversed as follows:

- The unique primary index is considered as a primary key.
- The primary index is considered as a non unique index.
- The secondary unique primary index is considered as an alternate key
- The secondary non unique primary index is considered as a non unique index.

You can use this RKM to retrieve specific Teradata metadata that is not supported by the standard JDBC interface (such as primary indexes).

10.6 Setting up Data Quality

Oracle Data Integrator provides the CKM Teradata for checking data integrity against constraints defined on a Teradata table. See "Set up Flow Control and Post-Integration Control" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details.

Oracle Data Integrator provides the Knowledge Module listed in Table 10–2 to perform a check on Teradata.

Table 10–2 Check Knowledge Modules for Teradata Database

Recommended KM	Notes
CKM Teradata	Checks data integrity against constraints defined on a Teradata table. Rejects invalid records in the error table created dynamically. Can be used for static controls as well as flow controls. This KM supports the following Teradata optimizations: ■ Primary Indexes ■ Statistics

10.7 Designing an Interface

You can use Teradata as a source, staging area or a target of an integration interface. It is also possible to create ETL-style integration interfaces based on the Teradata technology.

The KM choice for an interface or a check determines the abilities and performance of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning a Teradata data server.

10.7.1 Loading Data from and to Teradata

Teradata can be used as a source, target or staging area of an interface. The LKM choice in the Interface Flow tab to load data between Teradata and another type of data server is essential for the performance of an interface.

10.7.1.1 Loading Data from Teradata

Use the Generic SQL KMs or the KMs specific to the other technology involved to load data from a Teradata database to a target or staging area database.

For extracting data from a Teradata staging area to a file, use the IKM File to Teradata (TTU). See Section 10.7.2, "Integrating Data in Teradata" for more information.

10.7.1.2 Loading Data to Teradata

Oracle Data Integrator provides Knowledge Modules that implement optimized methods for loading data from a source or staging area into a Teradata database. These optimized Teradata KMs are listed in Table 10–3. In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technology involved.

Table 10–3 KMs for loading data to Teradata

Source or Staging Area Technology	KM	Notes
File	LKM File to Teradata (TTU)	Loads data from a File to a Teradata staging area database using the Teradata bulk utilities. Because this method uses the native Teradata utilities to load the file in the staging area, it is more efficient than the standard LKM File to SQL when dealing with large volumes of data. Consider using this LKM if your source is a large flat file and your staging area is a Teradata database. This KM support the following Teradata optimizations: ▪ Statistics ▪ Optimized Temporary Tables Management

Table 10–3 (Cont.) KMs for loading data to Teradata

Source or Staging Area Technology	KM	Notes
SQL	LKM SQL to Teradata (TTU)	Loads data from a SQL compliant source database to a Teradata staging area database using a native Teradata bulk utility. This LKM can unload the source data in a file or Named Pipe and then call the specified Teradata utility to populate the staging table from this file/pipe. Using named pipes avoids landing the data in a file. This LKM is recommended for very large volumes. Consider using this IKM when: ■ The source data located on a SQL compliant database is large ■ You don't want to stage your data between the source and the target ■ Your staging area is a Teradata database. This KM support the following Teradata optimizations: ■ Support for Teradata Utilities ■ Support for Named Pipes ■ Optimized Temporary Tables Management

10.7.2 Integrating Data in Teradata

Oracle Data Integrator provides Knowledge Modules that implement optimized data integration strategies for Teradata. These optimized Teradata KMs are listed in Table 10–4. In addition to these KMs, you can also use the Generic SQL KMs.

The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

Table 10–4 KMs for integrating data to Teradata

KM	Notes
IKM Teradata Control Append	Integrates data in a Teradata target table in replace/append mode. When flow data needs to be checked using a CKM, this IKM creates a temporary staging table before invoking the CKM. Consider using this IKM if you plan to load your Teradata target table in replace mode, with or without data integrity check. To use this IKM, the staging area must be on the same data server as the target Teradata table. This KM support the following Teradata optimizations: ■ Primary Indexes and Statistics ■ Optimized Temporary Tables Management

Table 10–4 (Cont.) KMs for integrating data to Teradata

KM	Notes
IKM Teradata Incremental Update	Integrates data in a Teradata target table in incremental update mode. This IKM creates a temporary staging table to stage the data flow. It then compares its content to the target table to guess which records should be inserted and which others should be updated. It also allows performing data integrity check by invoking the CKM. Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your Teradata target table to insert missing records and to update existing ones. To use this IKM, the staging area must be on the same data server as the target. This KM support the following Teradata optimizations: ■ Primary Indexes and Statistics ■ Optimized Temporary Tables Management
IKM Teradata Multi Statement	Integrates data in Teradata database target table using multi statement requests, managed in one SQL transaction
IKM Teradata Slowly Changing Dimension	Integrates data in a Teradata target table used as a Type II Slowly Changing Dimension in your Data Warehouse. This IKM relies on the Slowly Changing Dimension metadata set on the target datastore to figure out which records should be inserted as new versions or updated as existing versions. Because inserts and updates are done in bulk set-based processing, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your Teradata target table as a Type II Slowly Changing Dimension. To use this IKM, the staging area must be on the same data server as the target and the appropriate Slowly Changing Dimension metadata needs to be set on the target datastore. This KM support the following Teradata optimizations: ■ Primary Indexes and Statistics ■ Optimized Temporary Tables Management This KM also includes a COMPATIBLE option. This option corresponds to the Teradata engine major version number. If this version is 12 or above, then a MERGE statement will be used instead of the standard INSERT then UPDATE statements to merge the incoming data flow into the target table.

Table 10–4 (Cont.) KMs for integrating data to Teradata

KM	Notes
IKM Teradata to File (TTU)	Integrates data in a target file from a Teradata staging area in replace mode. This IKM requires the staging area to be on Teradata. It uses the native Teradata utilities to export the data to the target file. Consider using this IKM if you plan to transform and export data to a target file from your Teradata server. To use this IKM, the staging area must be different from the target. It should be set to a Teradata location. This KM support the following Teradata optimizations: ■ Support for Teradata Utilities
IKM File to Teradata (TTU)	This IKM is designed to leverage the power of the Teradata utilities for loading files directly to the target. It is restricted to one file as source and one Teradata table as target. Depending on the utility you choose, you'll have the ability to integrate the data in either replace or incremental update mode. Consider using this IKM if you plan to load a single flat file to your target table. Because it uses the Teradata utilities, this IKM is recommended for very large volumes. To use this IKM, you have to set the staging area to the source file's schema. This KM support the following Teradata optimizations: ■ Primary Indexes and Statistics ■ Support for Teradata Utilities ■ Optimized Temporary Tables Management.

Table 10–4 (Cont.) KMs for integrating data to Teradata

KM	Notes
IKM SQL to Teradata (TTU)	Integrates data from a SQL compliant database to a Teradata database target table using Teradata Utilities TPUMP, FASTLOAD OR MULTILOAD. This IKM is designed to leverage the power of the Teradata utilities for loading source data directly to the target. It can only be used when all source tables belong to the same data server and when this data server is used as a staging area (staging area on source). Source data can be unloaded into a file or Named Pipe and then loaded by the selected Teradata utility directly in the target table. Using named pipes avoids landing the data in a file. This IKM is recommended for very large volumes. Depending on the utility you choose, you'll have the ability to integrate the data in replace or incremental update mode. Consider using this IKM when: ■ You plan to load your target table with few transformations on the source ■ All your source tables are on the same data server (used as the staging area) ■ You don't want to stage your data between the source and the target To use this IKM, you have to set the staging area to the source data server's schema. This KM support the following Teradata optimizations: ■ Primary Indexes and Statistics ■ Support for Teradata Utilities ■ Support for Named Pipes ■ Optimized Temporary Tables Management
IKM SQL to Teradata Control Append	Integrates data from an ANSI-92 compliant source database into Teradata target table in truncate / insert (append) mode. This IKM is typically used for ETL configurations: source and target tables are on different databases and the interface's staging area is set to the logical schema of the source tables or a third schema. See Section 10.7.3, "Designing an ETL-Style Interface" for more information.

Using Slowly Changing Dimensions

For using slowly changing dimensions, make sure to set the *Slowly Changing Dimension* value for each column of the target datastore. This value is used by the IKM Teradata Slowly Changing Dimension to identify the Surrogate Key, Natural Key, Overwrite or Insert Column, Current Record Flag, and Start/End Timestamps columns.

Using Multi Statement Requests

Multi statement requests are typically enable the parallel execution of simple interfaces. The Teradata performance is improved by synchronized scans and by avoiding transient journal.

Set the KM options as follows:

- Interfaces using this KM must be used within a package:

  - In the first interface of the package loading a table via the multi-statement set the INIT_MULTI_STATEMENT option to `YES`.
  - The subsequent interfaces loading a table via the multi-statement must use this KM and have the INIT_MULTI_STATEMENT option set to `NO`.
  - The last interface must have the EXECUTE option set to `YES` in order to run the generated multi-statement.
- In the STATEMENT_TYPE option, specify the type of statement (insert or update) for each interface.
- In the SQL_OPTION option, specify the additional SQL sentence that is added at the end of the query, for example QUALIFY Clause.

Note the following limitations concerning multi-statements:

- Multi-statements are only supported when they are used within a package.
- Temporary indexes are not supported.
- Updates are considered as Inserts in terms of row count.
- Updates can only have a single Dataset.
- Only executing interface (EXECUTE = YES) reports row counts.
- Journalized source data not supported.
- Neither Flow Control nor Static Control is supported.
- The SQL_OPTION option applies only to the last Dataset.

10.7.3 Designing an ETL-Style Interface

See "Working with Integration Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for generic information on how to design integration interfaces. This section describes how to design an ETL-style interface where the staging area is on a Teradata database or any ANSI-92 compliant database and the target on Teradata.

In an ETL-style interface, ODI processes the data in a staging area, which is different from the target. Oracle Data Integrator provides two ways for loading the data from a Teradata or an ANSI-92 compliant staging area to a Teradata target:

- Using a Multi-connection IKM
- Using an LKM and a mono-connection IKM

Depending on the KM strategy that is used, flow and static control are supported.

Using a Multi-connection IKM

A multi-connection IKM allows integrating data into a target when the staging area and sources are on different data servers.

Oracle Data Integrator provides the following multi-connection IKM for handling Teradata data: IKM SQL to Teradata Control Append. You can also use the generic SQL multi-connection IKMs. See Chapter 4, "Generic SQL" for more information.

See Table 10–5 for more information on when to use a multi-connection IKM.

To use a multi-connection IKM in an ETL-style interface:

1. Create an integration interface with an ANSI-92 compliant staging area and the target on Teradata using the standard procedure as described in "Creating an

Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section describes only the ETL-style specific steps.

2. In the Definition tab of the Interface Editor, select **Staging Area different from Target** and select the logical schema of the source tables or a third schema.
3. In the Flow tab, select one of the Source Sets, by clicking its title. The Property Inspector opens for this object.
4. Select an LKM from the LKM Selector list to load from the source(s) to the staging area. See Table 10–5 to determine the LKM you can use.
5. Optionally, modify the KM options.
6. In the Flow tab, select the Target by clicking its title. The Property Inspector opens for this object.

   In the Property Inspector, select an ETL multi-connection IKM from the IKM Selector list to load the data from the staging area to the target. See Table 10–5 to determine the IKM you can use.

Note the following when setting the KM options of the IKM SQL to Teradata Control Append:

- If you do not want to create any tables on the target system, set `FLOW_CONTROL=false`. If `FLOW_CONTROL=false`, the data is inserted directly into the target table.
- If `FLOW_CONTROL=true`, the flow table is created on the target or on the staging area.
- If you want to recycle data rejected from a previous control, set `RECYCLE_ERROR=true` and set an update key for your interface.

Using an LKM and a mono-connection IKM

If there is no dedicated multi-connection IKM, use a standard exporting LKM in combination with a standard mono-connection IKM. The exporting LKM is used to load the flow table from the staging area to the target. The mono-connection IKM is used to integrate the data flow into the target table.

Oracle Data Integrator supports any ANSI SQL-92 standard compliant technology as a source and staging area of an ETL-style interface. The target is Teradata.

See Table 10–5 for more information on when to use the combination of a standard LKM and a mono-connection IKM.

To use an LKM and a mono-connection IKM in an ETL-style interface:

1. Create an integration interface with an ANSI-92 compliant staging area and the target on Teradata using the standard procedure as described in "Creating an Interface" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. In the Definition tab of the Interface Editor, select **Staging Area different from Target** and select the logical schema of the source tables or a third schema.
3. In the Flow tab, select one of the Source Sets.
4. In the Property Inspector, select an LKM from the LKM Selector list to load from the source(s) to the staging area. See Table 10–5 to determine the LKM you can use.
5. Optionally, modify the KM options.

6. Select the Staging Area. In the Property Inspector, select an LKM from the LKM Selector list to load from the staging area to the target. See Table 10–5 to determine the LKM you can use.
7. Optionally, modify the options.
8. Select the Target by clicking its title. The Property Inspector opens for this object.

   In the Property Inspector, select a standard mono-connection IKM from the IKM Selector list to update the target. See Table 10–5 to determine the IKM you can use.

Table 10–5 KM Guidelines for ETL-Style Interfaces with Teradata Data

Source	Staging Area	Target	Exporting LKM	IKM	KM Strategy	Comment
ANSI SQL-92 standard compliant	ANSI SQL-92 standard compliant	Teradata	NA	IKM SQL to Teradata Control Append	Multi-connection IKM	Recommended to perform control append Supports flow control.
ANSI SQL-92 standard compliant	Teradata or any ANSI SQL-92 standard compliant database	Teradata or any ANSI SQL-92 standard compliant database	NA	IKM SQL to SQL Incremental Update	Multi-connection IKM	Allows an incremental update strategy with no temporary target-side objects. Use this KM if it is not possible to create temporary objects in the target server. The application updates are made without temporary objects on the target, the updates are made directly from source to target. The configuration where the flow table is created on the staging area and not in the target should be used only for small volumes of data. Supports flow and static control

Table 10–5 (Cont.) KM Guidelines for ETL-Style Interfaces with Teradata Data

Source	Staging Area	Target	Exporting LKM	IKM	KM Strategy	Comment
ANSI SQL-92 standard compliant	Teradata or ANSI SQL-92 standard compliant	Teradata	LKM SQL to Teradata (TTU)	IKM Teradata Incremental Update	LKM + standard IKM	
ANSI SQL-92 standard compliant	Teradata	Teradata	LKM SQL to Teradata (TTU)	IKM Teradata Slowly Changing Dimension	LKM + standard IKM	
ANSI SQL-92 standard compliant	ANSI SQL-92 standard compliant	Teradata	LKM SQL to Teradata (TTU)	IKM SQL to Teradata (TTU)	LKM + standard IKM	If no flow control, this strategy is recommended for large volumes of data

10.8 KM Optimizations for Teradata

This section describes the specific optimizations for Teradata that are included in the Oracle Data Integrator Knowledge Modules.

This section includes the following topics:

- Primary Indexes and Statistics
- Support for Teradata Utilities
- Support for Named Pipes
- Optimized Management of Temporary Tables

10.8.1 Primary Indexes and Statistics

Teradata performance heavily relies on primary indexes. The Teradata KMs support customized primary indexes (PI) for temporary and target tables. This applies to Teradata LKMs, IKMs and CKMs. The primary index for the temporary and target tables can be defined in these KMs using the PRIMARY_INDEX KM option, which takes the following values:

- `[PK]`: The PI will be the primary key of each temporary or target table. This is the default value.
- `[NOPI]`: Do not specify primary index (Teradata 13.0 & above only).
- `[UK]`: The PI will be the update key of the interface. This is the default value.
  - <Column list>: This is a free PI based on the comma-separated list of column names.
  - <Empty string>: No primary index is specified. The Teradata engine will use the default rule for the PI (first column of the temporary table).

Teradata MultiColumnStatistics should optionally be gathered for selected PI columns. This is controlled by COLLECT_STATS KM option, which is set to true by default.

10.8.2 Support for Teradata Utilities

Teradata Utilities (TTU) provide an efficient method for transferring data from and to the Teradata engine. When using a LKM or IKM supporting TTUs, it is possible to set the method for loading data using the TERADATA_UTILITY option.

This option takes the following values when pushing data to a Teradata target (IKM) or staging area (LKM):

- `FASTLOAD`: use Teradata FastLoad
- `MLOAD`: use Teradata MultiLoad
- `TPUMP`: use Teradata TPump
- `TPT-LOAD`: use Teradata Parallel Transporter (Load Operator)
- `TPT-SQL-INSERT`: use Teradata Parallel Transporter (SQL Insert Operator)

This option takes the following values when pushing data FROM Teradata to a file:

- `FEXP`: use Teradata FastExport
- `TPT`: use Teradata Parallel Transporter

When using TTU KMs, you should also take into account the following KM parameters:

- REPORT_NB_ROWS: This option allows you to report the number of lines processed by the utility in a Warning step of the integration interface.
- SESSIONS: Number of FastLoad sessions
- MAX_ALLOWED_ERRORS: Maximum number of tolerated errors. This corresponds to the ERRLIMIT command in FastLoad/MultiLoad/TPump and to the ErrorLimit attribute for TPT.
- MULTILOAD_TPUMP_TYPE: Operation performed by the MultiLoad or TPump utility. Valid values are `INSERT`, `UPSERT` and `DELETE`. For `UPSERT` and `DELETE` an update key is required in the interface.

For details and appropriate choice of utility and load operator, refer to the Teradata documentation.

10.8.3 Support for Named Pipes

When using TTU KMs to move data between a SQL source and Teradata, it is possible to increase the performances by using Named Pipes instead of files between the unload/load processes. Named Pipes can be activated by setting the NP_USE_ NAMED_PIPE option to `YES`. The following options should also be taken into account for using Named Pipes:

- NP_EXEC_ON_WINDOWS: Set this option to `YES` if the run-time agent runs on a windows platform.
- NP_ACCESS_MODULE: Access module used for Named Pipes. This access module is platform dependant.
- NP_TTU_STARTUP_TIME: This number of seconds for the TTU to be able to receive data through the pipe. This is the delay between the moment the KM starts the TTU and the moment the KM starts to push data into the named pipe. This delay is dependant on the machine workload.

10.8.4 Optimized Management of Temporary Tables

Creating and dropping Data Integrator temporary staging tables can be a resource consuming process on a Teradata engine. The ODI_DDL KM option provides a mean to control these DDL operations. It takes the following values:

- DROP_CREATE: Always drop and recreate all temporary tables for every execution (default behavior).
- CREATE_DELETE_ALL: Create temporary tables when needed (usually for the first execution only) and use DELETE ALL to drop the temporary table content. Temporary table are reused for subsequent executions.
- DELETE_ALL: Do not create temporary tables. Only submit DELETE ALL for all temporary tables.
- NONE: Do not issue any DDL on temporary tables. Temporary tables should be handled separately.

11 Hypersonic SQL

This chapter describes how to work with Hypersonic SQL in Oracle Data Integrator.

This chapter includes the following sections:

11.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in an Hypersonic SQL database. Oracle Data Integrator features are designed to work best with Hypersonic SQL, including reverse-engineering, data integrity check, and integration interfaces.

11.1.1 Concepts

The Hypersonic SQL database concepts map the Oracle Data Integrator concepts as follows: A Hypersonic SQL server corresponds to a data server in Oracle Data Integrator. Within this server, one single Oracle Data Integrator physical schema maps to the database.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to Hypersonic SQL.

11.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 11–1 for handling Hypersonic SQL data. These KMs use Hypersonic SQL specific features. It is also possible to use the generic SQL KMs with the Hypersonic SQL database. See Chapter 4, "Generic SQL" for more information.

Table 11–1 Hypersonic SQL Knowledge Modules

Knowledge Module	Description
CKM HSQL	Checks data integrity against constraints defined on a Hypersonic SQL table. Rejects invalid records in the error table created dynamically. Can be used for static controls as well as flow controls.
JKM HSQL Consistent	Creates the journalizing infrastructure for consistent journalizing on Hypersonic SQL tables using triggers. Enables consistent Changed Data Capture on Hypersonic SQL.
JKM HSQL Simple	Creates the journalizing infrastructure for simple journalizing on Hypersonic SQL tables using triggers.
SKM HSQL	Generates data access Web services for Hypersonic SQL databases.

11.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Hypersonic SQL Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

11.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

11.2.2 Technology Specific Requirements

There are no technology-specific requirements for using Hypersonic SQL in Oracle Data Integrator.

11.2.3 Connectivity Requirements

This section lists the requirements for connecting to a Hypersonic SQL Database.

JDBC Driver

Oracle Data Integrator is installed with a JDBC driver for Hypersonic SQL. This driver directly uses the TCP/IP network layer and requires no other installed component or configuration.

11.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating a Hypersonic SQL Data Server
2. Creating a Hypersonic SQL Physical Schema

11.3.1 Creating a Hypersonic SQL Data Server

A Hypersonic SQL data server corresponds to an Hypersonic SQL Database connected with a specific Hypersonic SQL user account. This user will have access to the database via a physical schema in Oracle Data Integrator created under the data server.

Create a data server for the Hypersonic SQL technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Hypersonic SQL data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator
   - **Server**: Physical name of the data server
   - **User/Password**: Hypersonic SQL user with its password (usually `sa`)
2. In the JDBC tab:
   - **JDBC Driver**: `org.hsqldb.jdbcDriver`
   - **JDBC URL**: `jdbc:hsqldb:hsql://<host>:<port>`

     The URL parameters are:
     - `<host>`: Hypersonic SQL machine network name or IP address
     - `<port>`: Port number

11.3.2 Creating a Hypersonic SQL Physical Schema

Create a physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

11.4 Setting Up an Integration Project

Setting up a project using the Hypersonic SQL database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with Hypersonic SQL:

- CKM HSQL

Import also the Generic SQL KMs into your project. See Chapter 4, "Generic SQL" for more information about these KMs.

11.5 Creating and Reverse-Engineering a Hypersonic SQL Model

This section contains the following topics:

11.5.1 Create a Hypersonic SQL Model

Create a Hypersonic SQL Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

11.5.2 Reverse-engineer a Hypersonic SQL Model

Hypersonic SQL supports Standard reverse-engineering - which uses only the abilities of the JDBC driver.

To perform a Standard Reverse- Engineering on Hypersonic SQL use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

11.6 Setting up Changed Data Capture

The ODI Hypersonic SQL Knowledge Modules support the Changed Data Capture feature. See Chapter "Working with Changed Data Capture" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details on how to set up journalizing and how to use captured changes.

Hypersonic SQL Journalizing Knowledge Modules support Simple Journalizing and Consistent Set Journalizing. The JKMs use triggers to capture data changes on the source tables.

Oracle Data Integrator provides the Knowledge Modules listed in Table 11–2 for journalizing Hypersonic SQL tables.

Table 11–2 Hypersonic SQL Journalizing Knowledge Modules

KM	Notes
JKM HSQL Consistent	Creates the journalizing infrastructure for consistent journalizing on Hypersonic SQL tables using triggers. Enables consistent Changed Data Capture on Hypersonic SQL.
JKM HSQL Simple	Creates the journalizing infrastructure for simple journalizing on Hypersonic SQL tables using triggers.

11.7 Setting up Data Quality

Oracle Data Integrator provides the CKM HSQL for checking data integrity against constraints defined on a Hypersonic SQL table. See "Set up Flow Control and Post-Integration Control" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details.

Oracle Data Integrator provides the Knowledge Module listed in Table 11–3 to perform a check on Hypersonic SQL.

Table 11–3 Check Knowledge Modules for Hypersonic SQL Database

Recommended KM	Notes
CKM HSQL	Checks data integrity against constraints defined on a Hypersonic SQL table. Rejects invalid records in the error table created dynamically. Can be used for static controls as well as flow controls.

11.8 Designing an Interface

You can use Hypersonic SQL as a source, staging area or a target of an integration interface.

The KM choice for an interface or a check determines the abilities and performance of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning a Hypersonic SQL data server.

Oracle Data Integrator does not provide specific loading or integration knowledge modules for Hypersonic SQL. Use the Generic SQL KMs or the KMs specific to the other technologies used as source, target, or staging area.

12

IBM Informix

This chapter describes how to work with IBM Informix in Oracle Data Integrator.

This chapter includes the following sections:

12.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in an IBM Informix database. Oracle Data Integrator features are designed to work best with IBM Informix, including reverse-engineering, journalizing, and integration interfaces.

12.2 Concepts

The IBM Informix concepts map the Oracle Data Integrator concepts as follows: An IBM Informix Server corresponds to a data server in Oracle Data Integrator. Within this server, an Owner maps to an Oracle Data Integrator physical schema.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to an IBM Informix database.

12.3 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 12–1 for handling IBM Informix data. These KMs use IBM Informix specific features. It is also possible to use the generic SQL KMs with the IBM Informix database. See Chapter 4, "Generic SQL" for more information.

Table 12–1 IBM Informix Knowledge Modules

Knowledge Module	Description
IKM Informix Incremental Update	Integrates data in an IBM Informix target table in incremental update mode. This IKM creates a temporary staging table to stage the data flow. It then compares its content to the target table to guess which records should be inserted and which others should be updated. It also allows performing data integrity check by invoking the CKM. Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your IBM Informix target table to insert missing records and to update existing ones. To use this IKM, the staging area must be on the same data server as the target.
JKM Informix Consistent	Creates the journalizing infrastructure for consistent journalizing on IBM Informix tables using triggers. Enables Consistent Set Changed Data Capture on IBM Informix.
JKM Informix Simple	Creates the journalizing infrastructure for simple journalizing on IBM Informix tables using triggers. Enables Simple Changed Data Capture on IBM Informix.
LKM Informix to Informix (SAME SERVER)	Loads data from a source Informix database to a target Informix staging area located inside the same server. This LKM creates a view in the source database and a synonym in the staging area database. This method if often more efficient than the standard "LKM SQL to SQL" when dealing with large volumes of data. Consider using this LKM if your source tables are located on an IBM Informix database and your staging area is on an IBM Informix database located in the same Informix server.
RKM Informix	Retrieves IBM Informix specific metadata for tables, views, columns, primary keys and non unique indexes. This RKM accesses the underlying Informix catalog tables to retrieve metadata. Consider using this RKM if you plan to extract additional metadata from your Informix catalog when it is not provided by the default JDBC reverse-engineering process.
RKM Informix SE	Retrieves IBM Informix SE specific metadata for tables, views, columns, primary keys and non unique indexes. This RKM accesses the underlying Informix SE catalog tables to retrieve metadata. Consider using this RKM if you plan to extract additional metadata from your Informix SE catalog when it is not provided by the default JDBC reverse-engineering process.
SKM Informix	Generates data access Web services for IBM Informix databases. See SKM SQL in Chapter 4, "Generic SQL" for more details.

12.4 Specific Requirements

There are no specific requirements for using IBM Informix in Oracle Data Integrator.

13

IBM DB2 for iSeries

This chapter describes how to work with IBM DB2 for iSeries in Oracle Data Integrator.

This chapter includes the following sections:

13.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in IBM DB2 for iSeries. Oracle Data Integrator features are designed to work best with IBM DB2 for iSeries, including reverse-engineering, changed data capture, data integrity check, and integration interfaces.

13.1.1 Concepts

The IBM DB2 for iSeries concepts map the Oracle Data Integrator concepts as follows: An IBM DB2 for iSeries server corresponds to a data server in Oracle Data Integrator. Within this server, a collection or schema maps to an Oracle Data Integrator physical schema. A set of related objects within one schema corresponds to a data model, and each table, view or synonym will appear as an ODI datastore, with its attributes, columns and constraints.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to IBM DB2 for iSeries.

13.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 13–1 for handling IBM DB2 for iSeries data. In addition to these specific IBM DB2 for iSeries Knowledge Modules, it is also possible to use the generic SQL KMs with IBM DB2 for iSeries. See Chapter 4, "Generic SQL" for more information.

Table 13–1 IBM DB2 for iSeries Knowledge Modules

Knowledge Module	Description
IKM DB2 400 Incremental Update	Integrates data in an IBM DB2 for iSeries target table in incremental update mode.
IKM DB2 400 Incremental Update (CPYF)	Integrates data in an IBM DB2 for iSeries target table in incremental update mode. This IKM is similar to the "IKM DB2 400 Incremental Update" except that it uses the CPYF native OS/400 command to write to the target table, instead of set-based SQL operations.
IKM DB2 400 Slowly Changing Dimension	Integrates data in an IBM DB2 for iSeries target table used as a Type II Slowly Changing Dimension in your Data Warehouse.
JKM DB2 400 Consistent	Creates the journalizing infrastructure for consistent journalizing on IBM DB2 for iSeries tables using triggers.
JKM DB2 400 Simple	Creates the journalizing infrastructure for simple journalizing on IBM DB2 for iSeries tables using triggers.
JKM DB2 400 Simple (Journal)	Creates the journalizing infrastructure for simple journalizing on IBM DB2 for iSeries tables using the journals.
LKM DB2 400 Journal to SQL	Loads data from an IBM DB2 for iSeries source to a ANSI SQL-92 compliant staging area database. This LKM can source from tables journalized with the JKM DB2 400 Simple (Journal) as it refreshes the CDC infrastructure from the journals.
LKM DB2 400 to DB2 400	Loads data from an IBM DB2 for iSeries source database to an IBM DB2 for iSeries staging area database using CRTDDMF to create a DDM file on the target and transfer data from the source to this DDM file using CPYF.
LKM SQL to DB2 400 (CPYFRMIMPF)	Loads data from an ANSI SQL-92 compliant source database to an IBM DB2 for iSeries staging area database using a temporary file loaded into the DB2 staging area with CPYFRMIPF.
RKM DB2 400	Retrieves metadata for IBM DB2 for iSeries: physical files, tables, views, foreign keys, unique keys.

13.2 Installation and Configuration

Make sure you have read the information in this section before you start working with the IBM DB2 for iSeries technology:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

13.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

http://www.oracle.com/technology/products/oracle-data-integrator/index.html.

13.2.2 Technology Specific Requirements

Some of the Knowledge Modules for IBM DB2 for iSeries use specific features of this database. The following restrictions apply when using these Knowledge Modules.

See the IBM DB2 for iSeries documentation for additional information on these topics.

Using System commands

This section describes the requirements that must be met before using iSeries specific commands in the knowledge modules for IBM DB2 for iSeries:

- Knowledge modules using system commands such as CPYF or CPYFRMIPF require that the agent runs on the iSeries runs on the iSeries system.

Using CDC with Journals

This section describes the requirements that must be met before using the Journal-based Change Data Capture with IBM DB2 for iSeries:

- This journalizing method requires that a specific program is installed and runs on the iSeries system. See Setting up Changed Data Capture for more information.

13.2.3 Connectivity Requirements

This section lists the requirements for connecting to an IBM DB2 for iSeries system.

JDBC Driver

Oracle Data Integrator is installed with a default IBM DB2 Datadirect Driver. This drivers directly uses the TCP/IP network layer and requires no other installed component or configuration. You can alternatively use the drivers provided by IBM, such as the Native Driver when installing the agent on iSeries.

13.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating a DB2/400 Data Server
2. Creating a DB2/400 Physical Schema

13.3.1 Creating a DB2/400 Data Server

An IBM DB2/400 data server corresponds to an iSeries server connected with a specific user account. This user will have access to several databases in this server, corresponding to the physical schemas in Oracle Data Integrator created under the data server.

13.3.1.1 Creation of the Data Server

Create a data server for the IBM DB2/400 technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide*

for Oracle Data Integrator. This section details only the fields required or specific for defining an IBM DB2/400 data server:

1. In the Definition tab:
   - Name: Name of the data server that will appear in Oracle Data Integrator
   - Server: Physical name of the data server
   - User/Password: DB2 user with its password
2. In the JDBC tab:
   - JDBC Driver: `weblogic.jdbc.db2.DB2Driver`
   - JDBC URL: `jdbc:weblogic:db2://hostname:port[;property=value[;...]]`

13.3.2 Creating a DB2/400 Physical Schema

Create an IBM DB2/400 physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The work schema and data schema in this physical schema correspond each to a schema (collection or library). The work schema should point to a temporary schema and the data schema should point to the schema hosting the data to integrate.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

13.4 Setting Up an Integration Project

Setting up a project using the IBM DB2 for iSeries database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with IBM DB2 for iSeries:

- IKM DB2 400 Incremental Update
- IKM DB2 400 Slowly Changing Dimension
- JKM DB2 400 Consistent
- JKM DB2 400 Simple
- RKM DB2 400
- CKM SQL

13.5 Creating and Reverse-Engineering an IBM DB2/400 Model

This section contains the following topics:

- Create an IBM DB2/400 Model
- Reverse-engineer an IBM DB2/400 Model

13.5.1 Create an IBM DB2/400 Model

Create an IBM DB2/400 Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

13.5.2 Reverse-engineer an IBM DB2/400 Model

IBM DB2 for iSeries supports both Standard reverse-engineering - which uses only the abilities of the JDBC driver - and Customized reverse-engineering, which uses a RKM to retrieve the metadata.

In most of the cases, consider using the standard JDBC reverse engineering for starting.

Consider switching to customized reverse-engineering for retrieving more metadata. IBM DB2 for iSeries customized reverse-engineering retrieves the physical files, database tables, database views, columns, foreign keys and primary and alternate keys.

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on IBM DB2 for iSeries use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on IBM DB2 for iSeries with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the IBM DB2/400 technology:

In the Reverse tab of the IBM DB2/400 Model, select the `KM: RKM DB2 400.<project name>`.

13.6 Setting up Changed Data Capture

Oracle Data Integrator handles Changed Data Capture on iSeries with two methods:

- **Trigger-based CDC** on the journalized tables. This method is set up with the JKM DB2/400 Simple or JKM DB2/400 Consistent. This CDC is not different from the CDC on other systems. See Section 13.6.1, "Setting up Trigger-Based CDC" for more information.
- **Log-based CDC by reading the native iSeries transaction journals.** This method is set up with the JKM DB2/400 Journal Simple and used by the LKM DB2/400 Journal to SQL. This method does not support Consistent Set CDC and requires a platform-specific configuration. See Section 13.6.1, "Setting up Trigger-Based CDC" for more information.

13.6.1 Setting up Trigger-Based CDC

This method support Simple Journalizing and Consistent Set Journalizing. The IBM DB2 for iSeries JKMs use triggers to capture data changes on the source tables.

Oracle Data Integrator provides the Knowledge Modules listed in Table 13–2 for journalizing IBM DB2 for iSeries tables using triggers.

See Chapter "Working with Changed Data Capture" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details on how to set up journalizing and how to use captured changes.

Table 13–2 IBM DB2 for iSeries Journalizing Knowledge Modules

KM	Notes
JKM DB2 400 Consistent	Creates the journalizing infrastructure for consistent journalizing on IBM DB2 for iSeries tables using triggers.
JKM DB2 400 Simple	Creates the journalizing infrastructure for simple journalizing on IBM DB2 for iSeries tables using triggers.

13.6.2 Setting up Log-Based CDC

This method is set up with the JKM DB2/400 Journal Simple and used by the LKM DB2/400 Journal to SQL. It uses also an RPG program to retrieve the journal content.

13.6.2.1 How does it work?

A iSeries transaction journal contains the entire history of the data changes for a given period. It is handled by the iSeries system for tables that are journaled. A journaled table is either a table from a collection, or a table for which a journal receiver and a journal have been created and journaling started.

Reading the transaction journal is performed by the a journal retriever CDCRTVJRN RPG program provided with Oracle Data Integrator. This program loads on demand the tables of the Oracle Data Integrator CDC infrastructure (J$ tables) with the contents from the transaction journal.

This program can be either scheduled on the iSeries system or called by the KMs through a stored procedure also called CDCRTVJRN. This stored procedure is automatically created by the JKM DB2/400 Journal Simple and invoked by the LKM DB2/400 Journal to SQL when data extraction is needed.

13.6.2.2 CDCRTVJRN Program Details

This program connects to the native iSeries journal for a given table, and captures changed data information into the Oracle Data Integrator Journal (J$).

The program works as follows:

1. Journalized table attributes retrieval:
   a. Table attributes retrieval: PK columns, J$ table name, last journal reading date.
   b. Attributes enrichment (short names, record size, etc.) using the `QSYS.QADBXREF` system table.
   c. Location of the iSeries journal using the `QADBRTVFD()` API.
2. PK columns information retrieval:
   a. PK columns attributes (short name, data types etc.) using the `QSYS.QADBIFLD` system table.
   b. Attributes enrichment (real physical length) using the `QUSLFLD()` API.
   c. Data preprocessing (RPG to SQL datatype conversion) for the primary key columns.
3. Extraction the native journal information into the J$ table:

a. Native journal reading using the `QJoRetrieveJournalEntries()` API.

b. Conversion of the raw data to native SQL data and capture into the J$ table.

c. Update of the changes count.

This program accepts the parameters listed in Table 13–3.

Table 13–3 CDCRTVJRN Program Parameters

Parameter	RPG Type	SQL Type	Description
SbsTName	A138	Char(138)	Full name of the subscribers table in the following format: `<Lib>.<Table>`. Example: `ODILIB.SNP_SUBSCRIBERS`
JrnTName	A138	Char(138)	Full name of the table for which the extract is done from the journal. Example: `FINANCE.MY_COMPANY_ORDERS`
JrnSubscriber	A50	Char(50)	Name of the current subscriber. It must previously have been added to the list of subscribers.
LogMessages	A1	Char(1)	Flag activating logging in a spool file. Possible values are: `Y` enable logging, and `N` to disable logging.

13.6.2.3 Installing the CDC Components on iSeries

There are two major components installed on the iSeries system to enable native journal reading:

- The CDCRTVJRN Program. This program is provided in an archive that should installed in the iSeries system. The installation process is described below.
- The CDC Infrastructure. It includes the standard CDC objects (J$ tables, views, ...) and the CDCRTVJRN Stored Procedure created by the JKM and used by the LKM to read journals. This stored procedure executes the CDCRTVJRN program.

> **Note:** The program must be set up in a library defined in the Topology as the default work library for this iSeries data server. In the examples below, this library is called `ODILIB`.

Installing the CDCRTVJRN Program

To install the CDCRTVJRN program:

1. Identify the location the program SAVF file. It is located in the `ODI_HOME/setup/manual/cdc-iseries` directory, and is also available on the Oracle Data Integrator Companion CD.
2. Connect to the iSeries system.
3. Create the default work library if it does not exist yet. You can use, for example, the following command to create an ODILIB library:

```
CRTLIB LIB(ODILIB)
```

4. Create in this library an empty save file that has the same name as the SAVF file (mandatory). For example:

```
CRTSAVF FILE(ODILIB/SAVPGM0110)
```

5. Upload the local SAVF file on the iSeries system in the library and on top of the file you have just created. Make sure that the upload process is performed in binary mode.

   An FTP command sequence performing the upload is given below as an example.

```
FTP 192.168.0.13
LCD /oracle/odi/setup/manual/cdc-iseries/
BI
CD ODILIB
PUT SAVPGM0110
BYE
```

- Restore the objects from the save file, using the RSTOBJ command. For example:

```
RSTOBJ OBJ(*ALL) SAVLIB(CDCODIRELE) DEV(*SAVF) OBJTYPE(*ALL)
SAVF(ODILIB/SAVPGM0110) RSTLIB(ODILIB)
```

- Check that the objects are correctly restored. The target library should contain a program object called `CDCRTVJRN`.

  Use the following command below to view it:

```
WRKOBJ OBJ(ODILIB/CDCRTVJRN)
```

The CDCRTVJRN Stored Procedure

This procedure is used to call the CDCRTVJRN program. It is automatically created by the JKM DB2/400 Journal Simple KM when journalizing is started. Journalizing startup is described in the Change Data Capture topic.

The syntax for the stored procedure is provided below for reference:

```
create procedure ODILIB.CDCRTVJRN(
   SbsTName char(138), /* Qualified Subscriber Table Name */
   JrnTName char(138), /* Qualified Table Name */
   Subscriber char(50) , /* Subscriber Name */
   LogMessages char(1) /* Create a Log (Y - Yes, N - No) */
)
language rpgle
external name 'ODILIB/CDCRTVJRN'
```

> **Note:** The stored procedure and the program are installed in a library defined in the Topology as the default work library for this iSeries data server

13.6.2.4 Using the CDC with the Native Journals

Once the program is installed and the CDC is setup, using the native journals consists in using the LKM DB2/400 Journal to SQL to extract journalized data from the iSeries system. The retrieval process is triggered if the RETRIEVE_JOURNAL_ENTRIES option is set to `true` for the LKM.

13.6.2.5 Problems While Reading Journals

This section list the possibly issues when using this changed data capture method.

CDCRTVJRN Program Limits

The following limits exist for the CDCRTVJRN program:

- The source table should be journaled and the iSeries journal should be readable by the user specified in the iSeries data server.
- The source table should have one PK defined in Oracle Data Integrator.
- The PK declared in Oracle Data Integrator should be in the 4096 first octets of the physical record of the data file.
- The number of columns in the PK should not exceed 16.
- The total number of characters of the PK column names added to the number of columns of the PK should not exceed 255.
- Large object datatypes are not supported in the PK. Only the following SQL types are supported in the PK: SMALLINT, INTEGER, BIGINT, DECIMAL (Packed), NUMERIC (Zoned), FLOAT, REAL, DOUBLE, CHAR, VARCHAR, CHAR VARYING, DATE, TIME, TIMESTAMP and ROWID.
- Several instances of CDCRTVJRN should not be started simultaneously on the same system.
- Reinitializing the sequence number in the iSeries journal may have a critical impact on the program (program hangs) if the journal entries consumption date (SNP_SUBSCRIBERS.JRN_CURFROMDATE) is before the sequence initialization date. To work around this problem, you should manually set a later date in SNP_SUBSCRIBERS.JRN_CURFROMDATE.

Troubleshooting the CDCRTVJRN Program

The journal reading process can be put in trace mode:

- either by calling from your query tool the CDCRTVJRN stored procedure with the LogMsg parameter set to Y,
- or by forcing the CREATE_SPOOL_FILE LKM option to 1 then restarting the interface.

The reading process logs are stored in a spool file which can be reviewed using the WRKSPLF command.

You can also review the raw contents of the iSeries journal using the DSPJRN command.

13.7 Setting up Data Quality

Oracle Data Integrator provides the generic CKM SQL for checking data integrity against constraints defined in DB2/400. See "Set up Flow Control and Post-Integration Control" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for details.

See Chapter 4, "Generic SQL" for more information.

13.8 Designing an Interface

You can use IBM DB2 for iSeries as a source, staging area or a target of an integration interface.

The KM choice for an interface or a check determines the abilities and performance of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning an IBM DB2 for iSeries data server.

13.8.1 Loading Data from and to IBM DB2 for iSeries

IBM DB2 for iSeries can be used as a source, target or staging area of an interface. The LKM choice in the Interface Flow tab to load data between IBM DB2 for iSeries and another type of data server is essential for the performance of an interface.

13.8.1.1 Loading Data from IBM DB2 for iSeries

Oracle Data Integrator provides Knowledge Modules that implement optimized methods for loading data from IBM DB2 for iSeries to a target or staging area database. These optimized IBM DB2 for iSeries KMs are listed in Table 13–4.

In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technology involved to load data from IBM DB2 for iSeries to a target or staging area database.

Table 13–4 KMs for loading data from IBM DB2 for iSeries

Source or Staging Area Technology	KM	Notes
IBM DB2 for iSeries	LKM DB2 400 to DB2 400	Loads data from an IBM DB2 for iSeries source database to an IBM DB2 for iSeries staging area database using CRTDDMF to create a DDM file on the target and transfer data from the source to this DDM file using CPYF.
IBM DB2 for iSeries	LKM DB2 400 Journal to SQL	Loads data from an IBM DB2 for iSeries source to a ANSI SQL-92 compliant staging area database. This LKM can source from tables journalized with the JKM DB2 400 Simple (Journal) as it refreshes the CDC infrastructure from the journals.

13.8.1.2 Loading Data to IBM DB2 for iSeries

Oracle Data Integrator provides Knowledge Modules that implement optimized methods for loading data from a source or staging area into an IBM DB2 for iSeries database. These optimized IBM DB2 for iSeries KMs are listed in Table 13–5.

In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technology involved.

Table 13–5 KMs for loading data to IBM DB2 for iSeries

Source or Staging Area Technology	KM	Notes
IBM DB2 for iSeries	LKM DB2 400 to DB2 400	Loads data from an IBM DB2 for iSeries source database to an IBM DB2 for iSeries staging area database using CRTDDMF to create a DDM file on the target and transfer data from the source to this DDM file using CPYF.
SQL	LKM SQL to DB2 400 (CPYFRMIMPF)	Loads data from an ANSI SQL-92 compliant source database to an IBM DB2 for iSeries staging area database using a temporary file loaded into the DB2 staging area with CPYFRMIPF.

13.8.2 Integrating Data in IBM DB2 for iSeries

Oracle Data Integrator provides Knowledge Modules that implement optimized data integration strategies for IBM DB2 for iSeries. These optimized IBM DB2 for iSeries KMs are listed in Table 13–6. I

In addition to these KMs, you can also use the Generic SQL KMs.

The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

Table 13–6 KMs for integrating data to IBM DB2 for iSeries

KM	Notes
IKM DB2 400 Incremental Update	Integrates data in an IBM DB2 for iSeries target table in incremental update mode.
IKM DB2 400 Incremental Update (CPYF)	Integrates data in an IBM DB2 for iSeries target table in incremental update mode. This IKM is similar to the "IKM DB2 400 Incremental Update" except that it uses the CPYF native OS/400 command to write to the target table, instead of set-based SQL operations.
IKM DB2 400 Slowly Changing Dimension	Integrates data in an IBM DB2 for iSeries target table used as a Type II Slowly Changing Dimension in your Data Warehouse.

Using Slowly Changing Dimensions

For using slowly changing dimensions, make sure to set the *Slowly Changing Dimension* value for each column of the target datastore. This value is used by the IKM DB2 400 Slowly Changing Dimension to identify the Surrogate Key, Natural Key, Overwrite or Insert Column, Current Record Flag and Start/End Timestamps columns.

13.9 Specific Considerations with DB2 for iSeries

This section provides specific considerations when using Oracle Data Integrator in an iSeries environment.

13.9.1 Installing the Run-Time Agent on iSeries

The Oracle Data Integrator Standalone Agent can be installed on iSeries.

See the *Oracle Fusion Middleware Installation Guide for Oracle Data Integrato*r for more information.

13.9.2 Alternative Connectivity Methods for iSeries

It is preferable to use the built-in IBM DB2 Datadirect driver in most cases. This driver directly use the TCP/IP network layer and require no other components installed on the client machine. Other methods exist to connect DB2 on iSeries.

13.9.2.1 Using Client Access

It is also possible to connect through ODBC with the IBM Client Access component installed on the machine. This method does not have very good performance and does not support the reverse engineering and some other features. It is therefore not recommended.

13.9.2.2 Using the IBM JT/400 and Native Drivers

This driver appears as a `jt400.zip` file you must copy into your Oracle Data Integrator installation drivers directory.

To connect DB2 for iSeries with a Java application installed on the iSeries machine, IBM recommends that you use the JT/400 Native driver (`jt400native.jar`) instead of the JT/400 driver (`jt400.jar`). The Native driver provides optimized access to the DB2 system, but works only from the iSeries machine.

To support seamlessly both drivers with one connection, Oracle Data Integrator has a built-in Driver Wrapper for AS/400. This wrapper connects through the Native driver if possible, otherwise it uses the JT/400 driver. It is recommended that you use this wrapper if running agents installed on AS/400 systems.

To configure a data server with the driver wrapper:

1. Change the driver and URL to your AS/400 server with the following information:
   - **Driver**: `com.sunopsis.jdbc.driver.wrapper.SnpsDriverWrapper`
   - **URL**: `jdbc:snps400:<machine_name>[;param1=value1[;param2=value2...]]`
2. Set the following java properties for the java machine the run-time agent deployed on iSeries:
   - `HOST_NAME`: comma separated list of host names identifying the current machine.
   - `HOST_IP`: IP Address of the current machine.

   The value allow the wrapper to identify whether this data server is accessed on the iSeries machine or from a remote machine.

13.10 Troubleshooting

This section provides information on how to troubleshoot problems that you might encounter when using Oracle Knowledge Modules. It contains the following topics:

- Troubleshooting Error messages
- Common Problems and Solutions

13.10.1 Troubleshooting Error messages

Errors in Oracle Data Integrator appear often in the following way:

```
java.sql.SQLException: The application server rejected the connection.(Signon was
canceled.)
at ...
at ...
...
```

the `java.sql.SQLExceptioncode` simply indicates that a query was made to the database through the JDBC driver, which has returned an error. This error is frequently a database or driver error, and must be interpreted in this direction.

Only the part of text in bold must first be taken in account. It must be searched in the DB2 or iSeries documentation. If its contains sometimes an error code specific to your system, with which the error can be immediately identified.

If such an error is identified in the execution log, it is necessary to analyze the SQL code send to the database to find the source of the error. The code is displayed in the description tab of the erroneous task.

13.10.2 Common Problems and Solutions

This section describes common problems and solutions.

13.10.2.1 Connection Errors

- `UnknownDriverException`

  The JDBC driver is incorrect. Check the name of the driver.

- `The application requester cannot establish the connection.(<name or IP address>) Cannot open a socket on host: <name or IP address>, port: 8471 (Exception: java.net.UnknownHostException:<name or IP address>)`

  Oracle Data Integrator cannot connect to the database. Either the machine name or IP address is invalid, the DB2/400 Services are not started or the TCP/IP interface on AS/400 is not started. Try to ping the AS/400 machine using the same machine name or IP address, and check with the system administrator that the appropriate services are started.

- `Datasource not found or driver name not specified`

  The ODBC Datasource specified in the JDBC URL is incorrect.

- `The application server rejected the connection.(Signon was canceled.) Database login failed, please verify userid and password. Communication Link Failure. Comm RC=8001 - CWBSY0001 - ...`

  The user profile used is not valid. This error occurs when typing an invalid user name or an incorrect password.

- `Communication Link Failure`

  An error occurred with the ODBC connectivity. Refer to the Client Access documentation for more information.

- `SQL5001 - Column qualifier or table &2 undefined. SQL5016 - Object name &1 not valid for naming convention`

  Your JDBC connection or ODBC Datasource is configured to use the wrong naming convention. Use the ODBC Administrator to change your datasource to use the proper (*SQL or *SYS) naming convention, or use the appropriate option in the JDBC URL to force the naming conversion (for instance jdbc:as400://195.10.10.13;naming=system) . Note that if using the system naming convention in the Local Object Mask of the Physical Schema, you must enter %SCHEMA/%OBJECT instead of %SCHEMA.%OBJECT.

  "*SQL" should always be used unless your application is specifically designed for *SYS. Oracle Data Integrator uses the *SQL naming convention by default.

- `SQL0204 &1 in &2 type *&3 not found`

  The table you are trying to access does not exist. This may be linked to an error in the context choice, or in the sequence of operations (E.g.: The table is a temporary table which must be created by another interface).

- `Hexadecimal characters appear in the target tables. Accentuated characters are incorrectly transferred.`

  The iSeries computer attaches a language identifier or CCSID to files, tables and even fields (columns). CCSID 65535 is a generic code that identifies a file or field as being language independent: i.e. hexadecimal data. By definition, no translation is performed by the drivers. If you do not wish to update the CCSID of the file, then translation can be forced, in the JDBC URL, thanks to the flags ccsid=<ccsid code> and convert _ccsid_65535=yes|no. See the driver's documentation for more information.

- `SQL0901 SQL system error`

  This error is an internal error of the DB2/400 system.

- `SQL0206 Column &1 not in specified tables`

  `Keying error in a mapping/join/filter`. A string which is not a column name is interpreted as a column name, or a column name is misspelled.

  This error may also appear when accessing an error table associated to a datastore with a structure recently modified. It is necessary to impact in the error table the modification, or drop the error tables and let Oracle Data Integrator recreate it in the next execution.

14

IBM DB2 UDB

This chapter describes how to work with IBM DB2 UDB in Oracle Data Integrator.

This chapter includes the following sections:

14.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in an IBM DB2 UDB database. Oracle Data Integrator features are designed to work best with IBM DB2 UDB, including journalizing, data integrity checks, and integration interfaces.

14.2 Concepts

The IBM DB2 UDB concepts map the Oracle Data Integrator concepts as follows: An IBM DB2 UDB database corresponds to a data server in Oracle Data Integrator. Within this server, a schema maps to an Oracle Data Integrator physical schema.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to an IBM DB2 UDB database.

14.3 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 14–1 for handling IBM DB2 UDB data. These KMs use IBM DB2 UDB specific features. It is also possible to use the generic SQL KMs with the IBM DB2 UDB database. See Chapter 4, "Generic SQL" for more information

Table 14–1 IBM DB2 UDB Knowledge Modules

Knowledge Module	Description
IKM DB2 UDB Incremental Update	Integrates data in an IBM DB2 UDB target table in incremental update mode. This IKM creates a temporary staging table to stage the data flow. It then compares its content to the target table to identify which records should be inserted and which others should be updated. It also allows performing data integrity check by invoking the CKM. Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your IBM DB2 UDB target table to insert missing records and to update existing ones. To use this IKM, the staging area must be on the same data server as the target.
IKM DB2 UDB Slowly Changing Dimension	Integrates data in an IBM DB2 UDB target table used as a Type II Slowly Changing Dimension in your Data Warehouse. This IKM relies on the Slowly Changing Dimension metadata set on the target datastore to figure out which records should be inserted as new versions or updated as existing versions. Because inserts and updates are done in bulk set-based processing, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your IBM DB2 UDB target table as a Type II Slowly Changing Dimension. To use this IKM, the staging area must be on the same data server as the target and the appropriate Slowly Changing Dimension metadata needs to be set on the target datastore.
JKM DB2 UDB Consistent	Creates the journalizing infrastructure for consistent journalizing on IBM DB2 UDB tables using triggers. Enables Consistent Changed Data Capture on IBM DB2 UDB.
JKM DB2 UDB Simple	Creates the journalizing infrastructure for simple journalizing on IBM DB2 UDB tables using triggers. Enables Simple Changed Data Capture on IBM DB2 UDB.
LKM DB2 UDB to DB2 UDB (EXPORT_IMPORT)	Loads data from an IBM DB2 UDB source database to an IBM DB2 UDB staging area database using the native EXPORT / IMPORT commands. This module uses the EXPORT CLP command to extract data in a temporary file. Data is then loaded in the target staging DB2 UDB table using the IMPORT CLP command. This method if often more efficient than the standard LKM SQL to SQL when dealing with large volumes of data. Consider using this LKM if your source tables are located on a DB2 UDB database and your staging area is on a different DB2 UDB database.
LKM File to DB2 UDB (LOAD)	Loads data from a File to a DB2 UDB staging area database using the native CLP LOAD Command. Depending on the file type (Fixed or Delimited) this LKM will generate the appropriate LOAD script in a temporary directory. This script is then executed by the CLP and automatically deleted at the end of the execution. Because this method uses the native IBM DB2 loaders, it is more efficient than the standard LKM File to SQL when dealing with large volumes of data. Consider using this LKM if your source is a large flat file and your staging area is an IBM DB2 UDB database.

Table 14–1 (Cont.) IBM DB2 UDB Knowledge Modules

Knowledge Module	Description
LKM SQL to DB2 UDB	Loads data from any ANSI SQL-92 standard compliant source database to an IBM DB2 UDB staging area. This LKM is similar to the standard LKM SQL to SQL described in Chapter 4, "Generic SQL" except that you can specify some additional specific IBM DB2 UDB parameters.
LKM SQL to DB2 UDB (LOAD)	Loads data from any ANSI SQL-92 standard compliant source database to an IBM DB2 UDB staging area using the CLP LOAD command. This LKM unloads the source data in a temporary file and calls the IBM DB2 native loader using the CLP LOAD command to populate the staging table. Because this method uses the native IBM DB2 loader, it is often more efficient than the LKM SQL to SQL or LKM SQL to DB2 UDB methods when dealing with large volumes of data. Consider using this LKM if your source data located on a generic database is large, and when your staging area is an IBM DB2 UDB database.
SKM IBM UDB	Generates data access Web services for IBM DB2 UDB databases. See SKM SQL in Chapter 4, "Generic SQL" for more information.

14.4 Specific Requirements

Some of the Knowledge Modules for IBM DB2 UDB use operating system calls to invoke the IBM CLP command processor to perform efficient loads. The following restrictions apply when using such Knowledge Modules:

- The IBM DB2 UDB Command Line Processor (CLP) as well as the DB2 UDB Connect Software must be installed on the machine running the Oracle Data Integrator Agent.
- The server names defined in the Topology must match the IBM DB2 UDB connect strings used for these servers.
- Some DB2 UDB JDBC drivers require DB2 UDB Connect Software to be installed on the machine running the ODI Agent.

See the IBM DB2 documentation for more information.

15

Sybase AS Enterprise

This chapter describes how to work with Sybase AS Enterprise in Oracle Data Integrator.

This chapter includes the following sections:

15.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in a Sybase AS Enterprise database. Oracle Data Integrator features are designed to work best with Sybase AS Enterprise, including journalizing and integration interfaces.

15.2 Concepts

The Sybase AS Enterprise concepts map the Oracle Data Integrator concepts as follows: An Sybase AS Enterprise database corresponds to a data server in Oracle Data Integrator. Within this server, a database/owner pair maps to an Oracle Data Integrator physical schema.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to a Sybase AS Enterprise database.

15.3 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 15–1 for handling Sybase AS Enterprise data. These KMs use Sybase AS Enterprise specific features. It is also possible to use the generic SQL KMs with the Sybase AS Enterprise database. See Chapter 4, "Generic SQL" for more information.

Table 15–1 Sybase ASE Knowledge Modules

Knowledge Module	Description
IKM Sybase ASE Incremental Update	Integrates data in a Sybase Adaptive Server Enterprise target table in incremental update mode. This IKM creates a temporary staging table to stage the data flow. It then compares its content to the target table to guess which records should be inserted and which others should be updated. It also allows performing data integrity check by invoking the CKM. Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your Sybase Adaptive Server Enterprise target table to insert missing records and to update existing ones. To use this IKM, the staging area must be on the same data server as the target.
IKM Sybase ASE Slowly Changing Dimension	Integrates data in a Sybase Adaptive Server Enterprise target table used as a Type II Slowly Changing Dimension in your Data Warehouse. This IKM relies on the Slowly Changing Dimension metadata set on the target datastore to figure out which records should be inserted as new versions or updated as existing versions. Because inserts and updates are done in bulk set-based processing, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your Sybase Adaptive Server Enterprise target table as a Type II Slowly Changing Dimension. To use this IKM, the staging area must be on the same data server as the target and the appropriate Slowly Changing Dimension metadata needs to be set on the target datastore.
JKM Sybase ASE Consistent	Creates the journalizing infrastructure for consistent journalizing on Sybase Adaptive Server Enterprise tables using triggers. Enables Consistent Set Changed Data Capture on Sybase Adaptive Server Enterprise.
JKM Sybase ASE Simple	Creates the journalizing infrastructure for simple journalizing on Sybase Adaptive Server Enterprise tables using triggers. Enables Simple Changed Data Capture on Sybase Adaptive Server Enterprise.
LKM SQL to Sybase ASE	Loads data from any SQL compliant database to Sybase Adaptive Server Enterprise. This KM uses the ODI Agent to read selected data from the database and write the result into the target temporary table created dynamically. When using this KM on a journalized source table, the Journalizing table is first updated to flag the records consumed and then cleaned from these records at the end of the interface. This Knowledge Module is NOT RECOMMENDED when using LARGE VOLUMES. Other specific modules using Bulk utilities (SQL*LOADER, BULK INSERT...) or direct links (DBLINKS, Linked Servers...) are usually more efficient.

Table 15–1 (Cont.) Sybase ASE Knowledge Modules

Knowledge Module	Description
LKM SQL to Sybase ASE (BCP)	Loads data from any SQL compliant database to a Sybase Adaptive Server Enterprise staging area database using the BCP (Bulk Copy Program) utility. This LKM unloads the source data in a temporary file and calls the Sybase BCP utility to populate the staging table. Because this method uses the native BCP utility, it is often more efficient than the "LKM SQL to SQL" method when dealing with large volumes of data. Consider using this LKM if your source data located on a generic database is large, and when your staging area is a Sybase Adaptive Server Enterprise database.
LKM Sybase ASE to Sybase ASE (BCP)	Loads data from a Sybase Adaptive Server Enterprise source database to a Sybase Adaptive Server Enterprise staging area database using the native BCP out/BCP in commands. This module uses the native BCP (Bulk Copy Program) command to extract data in a temporary file. Data is then loaded in the target staging Sybase Adaptive Server Enterprise table using the native BCP command again. This method if often more efficient than the standard "LKM SQL to SQL" when dealing with large volumes of data. Consider using this LKM if your source tables are located on a Sybase Adaptive Server Enterprise instance and your staging area is on a different Sybase Adaptive Server Enterprise instance.

15.4 Specific Requirements

Some of the Knowledge Modules for Sybase Adaptive Server Enterprise use the BCP specific loading utility. The following restrictions apply when using such Knowledge Modules:

- The BCP utility as well as the Sybase Adaptive Server Enterprise Client must be installed on the machine running the Oracle Data Integrator Agent.
- The server names defined in the Topology must match the Sybase Adaptive Server Enterprise Client connect strings used for these servers.
- White spaces in server names defined on the Client are not supported.
- The target staging area database must have option "select into/bulk copy"
- Execution can remain pending if the file generated by the BCP program is empty.
- For performance reasons, it is often recommended to install Oracle Data Integrator Agent on the target server machine.

See the Sybase Adaptive Server Enterprise documentation for more information.

16

Sybase IQ

This chapter describes how to work with Sybase IQ in Oracle Data Integrator.

This chapter includes the following sections:

16.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in a Sybase IQ database. Oracle Data Integrator features are designed to work best with Sybase IQ, including data integrity check and integration interfaces.

16.2 Concepts

The Sybase IQ concepts map the Oracle Data Integrator concepts as follows: A Sybase IQ server corresponds to a data server in Oracle Data Integrator. Within this server, a schema maps to an Oracle Data Integrator physical schema.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to a Sybase IQ database.

16.3 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 16–1 for handling Sybase IQ data. These KMs use Sybase IQ specific features. It is also possible to use the generic SQL KMs with the Sybase IQ database. See Chapter 4, "Generic SQL" for more information.

Table 16–1 Sybase IQ Knowledge Modules

Knowledge Module	Description
CKM Sybase IQ	Checks data integrity against constraints defined on a Sybase IQ table. Rejects invalid records in the error table created dynamically. Can be used for static controls as well as flow controls. Consider using this KM if you plan to check data integrity on a Sybase IQ database.

Table 16–1 (Cont.) Sybase IQ Knowledge Modules

Knowledge Module	Description
IKM Sybase IQ Incremental Update	Integrates data in a Sybase IQ target table in incremental update mode. This IKM creates a temporary staging table to stage the data flow. It then compares its content to the target table to guess which records should be inserted and which others should be updated. It also allows performing data integrity check by invoking the CKM. Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your Sybase IQ target table to insert missing records and to update existing ones. To use this IKM, the staging area must be on the same data server as the target.
IKM Sybase IQ Slowly Changing Dimension	Integrates data in a Sybase IQ target table used as a Type II Slowly Changing Dimension in your Data Warehouse. This IKM relies on the Slowly Changing Dimension metadata set on the target datastore to figure out which records should be inserted as new versions or updated as existing versions. Because inserts and updates are done in bulk set-based processing, this IKM is optimized for large volumes of data. Consider using this IKM if you plan to load your Sybase IQ target table as a Type II Slowly Changing Dimension. To use this IKM, the staging area must be on the same data server as the target and the appropriate Slowly Changing Dimension metadata needs to be set on the target datastore.
LKM File to Sybase IQ (LOAD TABLE)	Loads data from a File to a Sybase IQ staging area database using the LOAD TABLE SQL command. Because this method uses the native LOAD TABLE command, it is more efficient than the standard "LKM File to SQL" when dealing with large volumes of data. However, the loaded file must be accessible from the Sybase IQ machine. Consider using this LKM if your source is a large flat file and your staging area is a Sybase IQ database.
LKM SQL to Sybase IQ (LOAD TABLE)	Loads data from any ANSI SQL-92 standard compliant source database to a Sybase IQ staging area database using the native LOAD TABLE SQL command. This LKM unloads the source data in a temporary file and calls the Sybase IQ LOAD TABLE SQL command to populate the staging table. Because this method uses the native LOAD TABLE, it is often more efficient than the LKM SQL to SQL method when dealing with large volumes of data. Consider using this LKM if your source data located on a generic database is large, and when your staging area is a Sybase IQ database.

16.4 Specific Requirements

Some of the Knowledge Modules for Sybase IQ use the LOAD TABLE specific command. The following restrictions apply when using such Knowledge Modules.

- The file to be loaded by the LOAD TABLE command needs to be accessible from the Sybase IQ machine. It could be located on the file system of the server or reachable from a UNC (Unique Naming Convention) path or mounted from a remote file system.

- UNC file paths are supported but not recommended as they may decrease performance.
- For performance reasons, it is often recommended to install Oracle Data Integrator Agent on the target server machine.

See the Sybase IQ documentation for more information.

Part II

Business Intelligence

This part describes how to work with Business Intelligence in Oracle Data Integrator.

Part II contains the following chapters:

17

Oracle Business Intelligence Enterprise Edition

This chapter describes how to work with Oracle Business Intelligence Enterprise Edition in Oracle Data Integrator.

This chapter includes the following sections:

17.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data from Oracle Business Intelligence Enterprise Edition (Oracle BI).

Oracle Data Integrator provides specific methods for reverse-engineering and extracting data from ADF View Objects (ADF-VOs) via the Oracle BI Physical Layer using integration interfaces.

17.1.1 Concepts

The Oracle Business Intelligence Enterprise Edition concepts map the Oracle Data Integrator concepts as follows: An Oracle BI Server corresponds to a data server in Oracle Data Integrator. Within this server, a catalog/owner pair maps to an Oracle Data Integrator physical schema.

Oracle Data Integrator connects to this server to access, via a bypass connection pool, the physical sources that support ADF View Objects.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to an Oracle BI Server.

17.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 17–1 for handling Oracle BI data. These KMs use Oracle BI specific features.

Table 17–1 Oracle BI Knowledge Modules

Knowledge Module	Description
RKM Oracle BI (Jython)	Retrieves the table structure in Oracle BI (columns and primary keys).
LKM Oracle BI to Oracle (DBLink)	Loads data from an Oracle BI source to an Oracle database area using dblinks.
LKM Oracle BI to SQL	Loads data from an Oracle BI source to any ANSI SQL-92 compliant database.
IKM Oracle BI to SQL Append	Integrates data into a ANSI-SQL92 target database from an Oracle BI source.

17.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Oracle BI Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

17.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

17.2.2 Technology Specific Requirements

There are no technology-specific requirements for using Oracle BI in Oracle Data Integrator.

17.2.3 Connectivity Requirements

This section lists the requirements for connecting to an Oracle BI Server.

JDBC Driver

Oracle Data Integrator uses the Oracle BI native driver to connect to the Oracle BI Server. This driver must be installed in your Oracle Data Integrator drivers directory.

Bypass Connection Pool

In Oracle BI, a sqlbypass database connection must be setup to bypass the ADF layer and directly fetch data from the underlying database. The name of this connection pool is required for creating the Oracle BI data server in Oracle Data Integrator.

17.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating an Oracle BI Data Server
2. Creating an Oracle BI Physical Schema

17.3.1 Creating an Oracle BI Data Server

A data server corresponds to a Oracle BI Server. Oracle Data Integrator connects to this server to access, via a bypass connection pool, the physical sources that support ADF View Objects. These physical objects are located under the view objects that are exposed in this server. This server is connected with a user who has access to several catalogs/schemas. Catalog/schemas pairs correspond to the physical schemas that are created under the data server.

17.3.1.1 Creation of the Data Server

Create a data server for the Oracle BI technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Oracle BI data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator
   - **Server**: Leave this field empty.
   - **User/Password**: Oracle BI user with its password
2. In the JDBC tab:
   - **JDBC Driver**: `oracle.bi.jdbc.AnaJdbcDriver`
   - **JDBC URL**: `jddbc:oraclebi://<host>:<port>`

     `<host>` is the server on which Oracle BI server is installed. By default the `<port>` number is `9703`.
3. In the Properties tab, add a JDBC property with the following key/value pair.
   - Key: `NQ_SESSION.SELECTPHYSICAL`
   - Value: `Yes`

   Note: This option is required for accessing the physical data. Using this option makes the Oracle BI connection read-only.
4. In the Flexfield tab, set the name of the bypass connection pool in the CONNECTION_POOL flexfield.
   - Name: `CONNECTION_POOL`
   - Value: `<connection pool name>`

   Note: Note this bypass connection pool must also be defined in the Oracle BI server itself.

17.3.2 Creating an Oracle BI Physical Schema

Create a Oracle BI physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

In the physical schema the Data and Work Schemas correspond each to an Oracle BI Catalog/schema pair.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

17.4 Setting Up an Integration Project

Setting up a project using an Oracle BI Server follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with Oracle BI:

- RKM Oracle BI (Jython)
- LKM Oracle BI to Oracle (DBLink)
- LKM Oracle BI to SQL
- IKM Oracle BI to SQL Append

Import also the knowledge modules (IKM, CKM) required for the other technologies involved in your project.

17.5 Creating and Reverse-Engineering an Oracle BI Model

This section contains the following topics:

- Create an Oracle BI Model
- Reverse-engineer an Oracle BI Model

17.5.1 Create an Oracle BI Model

Create an Oracle BI Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

17.5.2 Reverse-engineer an Oracle BI Model

Oracle BI supports Customized reverse-engineering.

To perform a Customized Reverse-Engineering on Oracle BI with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Oracle BI technology:

1. In the Reverse tab of the Oracle BI Model, select the `KM: RKM Oracle BI (Jython).<project name>`.

This KM implements the USE_LOG and LOG_FILE_NAME logging options to trace the reverse-engineering process.

17.6 Setting up Data Quality

Data integrity check is not supported in an Oracle BI Server. You can check data extracted Oracle BI in a staging area using another technology.

17.7 Designing an Interface

You can use Oracle BI as a source of an integration interface.

The KM choice for an interface determines the abilities and performance of this interface. The recommendations in this section help in the selection of the KM for different situations concerning an Oracle BI server.

17.7.1 Loading Data from and to Oracle BI

The LKM choice in the Interface Flow tab to load data between Oracle BI and another type of data server is essential for the performance of an interface.

17.7.1.1 Loading Data from Oracle BI

Use the knowledge modules listed in Table 17–2 to load data from an Oracle BI server to a target or staging area database.

Table 17–2 KMs for loading data From Oracle BI

Staging Area/Target Technology	KM	Notes
Oracle	LKM Oracle BI to Oracle (Dblink)	Loads data from an Oracle BI source to an Oracle Database staging area using DBLinks. To use this knowledge module, a DBLink must be manually created from the source Fusion Transaction DB (that is the database storing the underlying data tables) to the Oracle staging area. This DBLink name must be the one specified in the Oracle staging area data server connection.
SQL	LKM Oracle BI to SQL	Loads data from an Oracle BI Source to an ANSI SQL-92 compliant staging area database via the agent.
SQL	IKM Oracle BI to SQL Append	Loads and Integrates data from an Oracle BI Source to an ANSI SQL-92 compliant staging area database via the agent. To use this KM, you must set the staging are of your interface on the source Oracle BI server. In this configuration, no temporary table is created and data is loaded and integrated directly from the source to the target tables.

17.7.1.2 Loading Data to Oracle BI

Oracle BI cannot be used as a staging area. No LKM targets Oracle BI.

17.7.2 Integrating Data in Oracle BI

Oracle BI cannot be used as a target or staging area. It is not possible to integrate data into Oracle BI with the knowledge modules.

18

Oracle Hyperion Essbase

This chapter describes how to work with Oracle Hyperion Essbase in Oracle Data Integrator.

This chapter includes the following sections:

18.1 Introduction

Oracle Data Integrator Adapter for Oracle's Hyperion Essbase enables you to connect and integrate Essbase with virtually any source or target using Oracle Data Integrator. The adapter provides a set of Oracle Data Integrator Knowledge Modules (KMs) for loading and extracting metadata and data and calculating data in Essbase applications.

18.1.1 Integration Process

You can use Oracle Data Integrator Adapter for Essbase to perform these data integration tasks on an Essbase application:

- Load metadata and data
- Extract metadata and data

Using the adapter to load or extract metadata or data involves the following tasks:

- Setting up an environment: defining data servers and schemas.

  See Section 18.3, "Setting up the Topology".

- Reverse-engineering an Essbase application using the Reverse-engineering Knowledge Module (RKM)

  See Section 18.4, "Creating and Reverse-Engineering an Essbase Model".

- Extracting metadata and data using Load Knowledge Modules (LKM).

  See Section 18.5, "Designing an Interface"

- Integrating the metadata and data into the Essbase application using the Integration Knowledge Modules (IKM).

  See Section 18.5, "Designing an Interface"

18.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 18–1 for handling Hyperion Essbase data. These KMs use Hyperion Essbase specific features. It is also possible to use the generic SQL KMs with the Hyperion Essbase database. See Chapter 4, "Generic SQL" for more information.

Table 18–1 Hyperion Essbase Knowledge Modules

Knowledge Module	Description
RKM Hyperion Essbase	Reverse-engineers Essbase applications and creates data models to use as targets or sources in Oracle Data Integrator interfaces
IKM SQL to Hyperion Essbase (DATA)	Integrates data into Essbase applications.
IKM SQL to Hyperion Essbase (METADATA)	Integrates metadata into Essbase applications
LKM Hyperion Essbase DATA to SQL	Loads data from an Essbase application to any SQL compliant database used as a staging area.
LKM Hyperion Essbase METADATA to SQL	Loads metadata from an Essbase application to any SQL compliant database used as a staging area.

18.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Oracle Data Integrator Adapter for Essbase:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

18.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

18.2.2 Technology Specific Requirements

There are no technology-specifc requirements for using the Oracle Data Integrator Adapter for Essbase.

18.2.3 Connectivity Requirements

There are no connectivity-specific requirements for using the Oracle Data Integrator Adapter for Essbase.

18.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating an Hyperion Essbase Data Server
2. Creating an Hyperion Essbase Physical Schema

18.3.1 Creating an Hyperion Essbase Data Server

Create a data server for the Hyperion Essbase technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Hyperion Essbase data server:

1. In the Definition tab:
   - Name: Enter a name for the data server definition.
   - Server (Data Server): Enter the Essbase server name.

   Note: If the Essbase server is running on a port other than the default port (1423), then provide the Essbase server details in this format, `<Essbase Server hostname>:<port>`.

2. Under Connection, enter a user name and password for connecting to the Essbase server.

   Note: The Test button does not work for an Essbase data server connection. This button works only for relational technologies that have a JDBC Driver.

18.3.2 Creating an Hyperion Essbase Physical Schema

Create a Hyperion Essbase physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Under Application (Catalog) and Application (Work Catalog), specify an Essbase application and under Database (Schema) and Database (Work Schema), specify an Essbase database associated with the application you selected.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

18.4 Creating and Reverse-Engineering an Essbase Model

This section contains the following topics:

- Create an Essbase Model
- Reverse-engineer an Essbase Model

18.4.1 Create an Essbase Model

Create an Essbase Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*

18.4.2 Reverse-engineer an Essbase Model

Reverse-engineering an Essbase application creates an Oracle Data Integrator model that includes a datastore for each dimension in the application and a datastore for data.

To perform a Customized Reverse-Engineering on Hyperion Essbase with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Hyperion Essbase technology.

1. In the Reverse tab of the Essbase Model, select the RKM Hyperion Essbase.
2. Set the KM options as indicated in Table 18–2.

Table 18–2 RKM Hyperion Essbase Options

Option	Possible Values	Description
MULTIPLE_DATA_COLUMNS	▪ No (Default) ▪ Yes	If this option is set to `No`, then the datastore created for the data extract / load model contains one column for each of the standard dimensions and a single data column. If this option is set to `Yes`, then the datastore created for the data extract / load model contains one column for each of the standard dimensions excluding the dimension specified by the DATA_COLUMN_DIMENSION option and as many data columns as specified by the comma separated list for the DATA_COLUMN_MEMBERS option.
DATA_COLUMN_DIMENSION	Account	This option is only applicable if MULTIPLE_DATA_COLUMNS is set to `Yes`. Specify the data column dimension name. For example, data columns are spread across the dimension Account or Time, and so on.

Table 18–2 (Cont.) RKM Hyperion Essbase Options

Option	Possible Values	Description
DATA_COLUMN_ MEMBERS	Account	This option is only applicable if MULTIPLE_ DATA_COLUMNS is set to `Yes`. Separate the required data column members with, (Comma). For example, if the data column dimension is set to Account and members are set to `Sales,COGS` then the datastore for data extract/load contains one column for each of the dimension except the data column dimension and one column for each of the data column member specified in the comma separated value. For example. Assuming that the dimensions in the Essbase application are Account, Scenario, Product, Market, and Year and the data column dimension is specified as Account and Data Column Members as `Sales, COGS`, the datastore will have the following columns: ■ Scenario (String) ■ Product (String) ■ Market (String) ■ Year (String) ■ Sales (Numeric) ■ COGS (Numeric)
EXTRACT_ ATTRIBUTE_ MEMBERS	■ No (Default) ■ Yes	If this option is set to `No`, then the datastore created for the data extract / load model contains one column for each of the standard dimensions and a single data column. Attribute dimensions are not included. If this option is set to `Yes`, then the data model contains these columns. ■ One column is created for each of the standard dimensions ■ One or more Data column(s) are created depending upon the value of the MULTIPLE_DATA_COLUMN option ■ One column is created for each of the associated attribute dimension

The RKM connects to the application (which is determined by the logical schema and the context) and imports some or all of these datastores, according to the dimensions in the application.

18.5 Designing an Interface

After reverse-engineering an Essbase application as a model, you can use the datastores in this model in these ways:

- Targets of interfaces for loading data and metadata into the application
- Sources of interfaces for extracting metadata and data from the application.

The KM choice for an interface determines the abilities and performance of this interface. The recommendations in this section help in the selection of the KM for different situations concerning Hyperion Essbase.

This section contains the following topics:

- Loading Metadata
- Loading Data
- Extracting Data

18.5.1 Loading Metadata

Oracle Data Integrator provides the IKM SQL to Hyperion Essbase (METADATA) for loading metadata into an Essbase application.

Metadata consists of dimension members. You must load members, or metadata, before you load data values for the members.

You can load members only to dimensions that exist in Essbase. You must use a separate interface for each dimension that you load. You can chain interfaces to load metadata into several dimensions at once.

Note: The metadata datastore can also be modified by adding or delete columns to match the dimension build rule that will be used to perform the metadata load. For example, the default datastore would have columns for ParentName and ChildName, if the rule is a generational dimension build rule, you can modify the metadata datastore to match the columns within your generational dimension build rule. The loadMarkets interface within the samples is an example of performing a metadata load using a generational dimension build rule.

Table 18–3 lists the options of the IKM SQL to Hyperion Essbase (METADATA). These options define how the adapter loads metadata into an Essbase application.

Table 18–3 IKM SQL to Hyperion Essbase (METADATA) Options

Option	Values	Description
RULES_FILE	Blank (Default)	Specify the rules file for loading or building metadata. If the rules file is present on the Essbase server, then, only specify the file name, otherwise, specify the fully qualified file name with respect to the Oracle Data Integrator Agent.
RULE_SEPARATOR	, (Default)	(Optional) Specify a rule separator in the rules file. These are the valid values: ■ Comma ■ Tab ■ Space ■ Custom character; for example, @, #, ^

Table 18–3 (Cont.) IKM SQL to Hyperion Essbase (METADATA) Options

Option	Values	Description
RESTRUCTURE_DATABASE	■ KEEP_ALL_DATA (Default) ■ KEEP_INPUT_DATA ■ KEEP_LEVEL0_DATA ■ DISCARD_ALL_DATA	Restructure database after loading metadata in the Essbasecube. These are the valid values: ■ KEEP_ALL_DATA- Keep all the data ■ KEEP_INPUT_DATA Keep only ■ input data ■ KEEP_LEVEL0_DATA-Keep only ■ level 0 data ■ DISCARD_ALL_DATA-Discard all ■ data Note: This option is applicable for the Essbase Release 9.3 and later. For the Essbase releases prior to 9.3, this option is ignored.
PRE_LOAD_MAXL_SCRIPT	Blank (Default)	Enable this option to execute a MAXL script before loading metadata to the Essbase cube. Specify a fully qualified path name (without blank spaces) for the MAXL script file. Note: To successfully execute this option, the Essbase client must be installed and configured on the machine where the Oracle Data Integrator Agent is running.
POST_LOAD_MAXL_SCRIPT	Blank (Default)	Enable this option to execute a MAXL script after loading metadata to the Essbase cube. Specify a fully qualified path name (without blank spaces) for the MAXL script file. Note: To successfully execute this option, the Essbase client must be installed and configured on the machine where the Oracle Data Integrator Agent is running.
ABORT_ON_PRE_MAXL_ERROR	■ No (Default) ■ Yes	This option is only applicable if you are enabling the PRE_LOAD_MAXL_SCRIPT option. If you set the ABORT_ON_PRE_MAXL_ERROR option to Yes, then the load process is aborted on encountering any error while executing the pre-MAXL script.
LOG_ENABLED	■ No (Default) ■ Yes	If this option is set to Yes, during the IKM process, logging is done to the file specified in the LOG_FILE_NAME option.
LOG_FILE_NAME	<?=java.lang.System.getProperty(“java.io.tmpdir”)?>/Extract_<%=snpRef.getFrom()%>.log (Default)	Specify a file name to log events of the IKM process.

Table 18–3 (Cont.) IKM SQL to Hyperion Essbase (METADATA) Options

Option	Values	Description
ERROR_LOG_FILENAME	<?=java.lang.System.getProperty("java.io.tmpdir")?>/Extract_<%=snpRef.getFrom()%>.log (Default)	Specify a file name to log the error records of the IKM process.

18.5.2 Loading Data

Oracle Data Integrator provides the IKM SQL to Hyperion Essbase (DATA) for loading data into an Essbase application.

You can load data into selected dimension members that are already created in Essbase. For a successful data load, all the standard dimension members are required and they should be valid members. You must set up the Essbase application before you can load data into it.

You can also create a custom target to match a load rule.

Before loading data, ensure that the members (metadata) exist in the Essbase dimension. The data load fails for records that have missing members and this information is logged (if logging is enabled) as an error record and the data load process will continue until the maximum error threshold is reached.

Note: The data datastore can also be modified by adding or delete columns to match the data load rule that will be used to perform the data load.

Table 18–4 lists the options of the IKM SQL to Hyperion Essbase (DATA). These options define how the adapter loads and consolidates data in an Essbase application.

Table 18–4 IKM SQL to Hyperion Essbase (DATA)

Option	Values	Description
RULES_FILE	Blank (Default)	(Optional) Specify a rules file to enhance the performance of data loading. Specify a fully qualified file name if the rules file is not present on the Essbase server. If the rules file option is not specified, then the API-based data load is used. However, you cannot specify the API.
RULE_SEPARATOR	, (Default)	(Optional) Specify a rule separator in the rules file. These are the valid values: ▪ Comma ▪ Tab ▪ Space ▪ Custom character; for example, @, #, ^

Table 18–4 (Cont.) IKM SQL to Hyperion Essbase (DATA)

Option	Values	Description
GROUP_ID	Integer	When performing multiple data loads in parallel, many interfaces can be set to use the same GROUP_ID. This GROUP _ID is used to manage parallel loads allowing the data load to be committed when the final interface for the GROUP_ID is complete. For more information on loading to parallel ASO cubes, refer to the Essbase Database Administrators guide.
BUFFER_ID	1–1000000	Multiple data load buffers can exist on an aggregate storage database. To save time, you can load data into multiple data load buffers at the same time. Although only one data load commit operation on a database can be active at any time, you can commit multiple data load buffers in the same commit operation, which is faster than committing buffers individually. For more information on loading to parallel ASO cubes, refer to the Essbase Database Administrators guide.
BUFFER_SIZE	0-100	When performing an incremental data load, Essbase uses the aggregate storage cache for sorting data. You can control how much of the cache a data load buffer can use by specifying the percentage (between 0 and 100% inclusive). By default, the resource usage of a data load buffer is set to 100, and the total resource usage of all data load buffers created on a database cannot exceed 100. For example, if a buffer of 90 exists, you cannot create another buffer of a size greater than 10. A value of 0 indicates to Essbase to use a self-determined, default load buffer size.
CLEAR_DATABASE	■ None (Default) ■ All ■ Upper Blocks ■ Non-input Blocks	Enable this option to clear data from the Essbase cube before loading data into it. These are the valid values: ■ None—Clear database will not happen ■ All—Clears all data blocksinput data ■ Upper Blocks—Clears all consolidated level blocks ■ Non-Input Blocks—Clears blocks containing values derived from calculations Note: For ASO applications, the Upper Blocks and Non-Input Blocks options will not be applicable.
CALCULATION_ SCRIPT	Blank (Default)	(Optional) Specify the calculation script that you want to run after loading data in the Essbase cube. Provide a fully qualified file name if the calculation script is not present on the Essbase server.
RUN_CALC_ SCRIPT_ONLY	■ No (Default) ■ Yes	This option is only applicable if you have specified a calculation script in the CALCULATION_SCRIPT option. If you set the RUN_CALC_SCRIPT_ONLY option to Yes, then only the calculation script is executed without loading the data into the target Essbase cube.

Table 18–4 (Cont.) IKM SQL to Hyperion Essbase (DATA)

Option	Values	Description
PRE_LOAD_MAXL_SCRIPT	Blank (Default)	Enable this option to execute a MAXL script before loading data to the Essbase cube. Specify a fully qualified path name (without blank spaces) for the MAXL script file. Note: Essbase client must be installed and configured on the machine where the Oracle Data Integrator Agent is running.
POST_LOAD_MAXL_SCRIPT	Blank (Default)	Enable this option to execute a MAXL script after loading data to the Essbase cube. Specify a fully qualified path name (without blank spaces) for the MAXL script file. Note: Essbase client must be installed and configured on the machine where the Oracle Data Integrator Agent is running.
ABORT_ON_PRE_MAXL_ERROR	■ No (Default) ■ Yes	This option is only applicable if you are enabling the PRE_LOAD_MAXL_SCRIPT option. If you set the ABORT_ON_PRE_MAXL_ERROR option to Yes, then the load process is aborted on encountering any error while executing pre-MAXL script.
MAXIMUM_ERRORS_ALLOWED	1 (Default)	Enable this option to set the maximum number of errors to be ignored before stopping a data load. The value that you specify here is the threshold limit for error records encountered during a data load process. If the threshold limit is reached, then the data load process is aborted. For example, the default value 1 means that the data load process stops on encountering a single error record. If value 5 is specified, then data load process stops on encountering the fifth error record. If value 0 (== infinity) is specified, then the data load process continues even after error records are encountered.
COMMIT_INTERVAL	1000 (Default)	Commit Interval is the chunk size of records that are loaded in the Essbase cube in a complete batch. Enable this option to set the Commit Interval for the records in the Essbase cube. Changing the Commit Interval can increase performance of data load based on design of the Essbase database.
LOG_ENABLED	■ No (Default) ■ Yes	If this option is set to Yes, during the IKM process, logging is done to the file specified in the LOG_FILENAME option.
LOG_FILENAME	<?=java.lang.System.getProperty (“java.io.tmpdir”)?/<% =snpRef.getTargetTable ("RES_NAME")%>.log (Default)	Specify a file name to log events of the IKM process.

Table 18–4 (Cont.) IKM SQL to Hyperion Essbase (DATA)

Option	Values	Description
LOG_ERRORS	■ No (Default) ■ Yes	If this option is set to Yes, during the IKM process, details of error records are logged to the file specified in the ERROR_LOG_FILENAME option.
ERROR_LOG_FILENAME	<?=java.lang.System.getProperty (java.io.tmpdir")?>/<% =snpRef.getTargetTable ("RES_NAME")%>.err	Specify a file name to log error record details of the IKM process.
ERR_LOG_HEADER_ROW	■ No (Default) ■ Yes	If this option is set to Yes, then the header row containing the column names are logged to the error records file.
ERR_COL_DELIMITER	, (Default)	Specify the column delimiter to be used for the error records file.
ERR_ROW_DELIMITER	\r\n (Default)	Specify the row delimiter to be used for the error records file.
ERR_TEXT_DELIMITER	' (Default)	Specify the text delimiter to be used for the column data in the error records file. For example, if the text delimiter is set as ' " ' (double quote), then all the columns in the error records file will be delimited by double quotes.

18.5.3 Extracting Data

This section includes the following topics:

- Data Extraction Methods for Essbase
- Extracting Essbase Data
- Extracting Members from Metadata

18.5.3.1 Data Extraction Methods for Essbase

The Oracle Data Integrator Adapter for Essbase supports querying and scripting for data extraction. To extract data, as a general process, create an extraction query and provide the extraction query to the adapter. Before the adapter parses the output of the extraction query and populates the staging area, a column validation is done. The adapter executes the extraction query based on the results of the metadata output query during the validation. The adapter does the actual parsing of the output query only when the results of the column validation are successful.

After the extraction is complete, validate the results—make sure that the extraction query has extracted data for all the output columns.

You can extract data with these Essbase-supported query and scripts:

- Data Extraction Using Report Scripts
- Data Extraction Using MDX Queries
- Data Extraction Using Calculation Scripts

Data Extraction Using Report Scripts

Data can be extracted by parsing the reports generated by report scripts. The report scripts can exist on the client computer as well as server, where Oracle Data Integrator is running on the client computer and Essbase is running on the server. The column validation is not performed when extracting data using report scripts. So, the output columns of a report script is directly mapped to the corresponding connected column in the source model. However, before you begin data extract using report scripts, you must complete these tasks:

- Suppress all formatting in the report script. Include this line as the first line in the report script—{ROWREPEAT SUPHEADING SUPFORMAT SUPBRACKETS SUPFEED SUPCOMMAS NOINDENTGEN TABDELIMIT DECIMAL 15}.
- The number of columns produced by a report script must be greater than or equal to the connected columns from the source model.
- The column delimiter value must be set in the LKM option.

Data Extraction Using MDX Queries

An MDX query is an XML-based data-extraction mechanism. You can specify the MDX query to extract data from an Essbase application. However, before you begin data extract using MDX queries, you must complete these tasks:

- The names of the dimension columns must match with the dimensions in the Essbase cube.
- For Type 1 data extraction, all the names of data columns must be valid members of a single standard dimension.
- For Type 1 data extraction, it is recommended that the data dimension exists in the lower level axis, that is, axis (0) of columns. If it is not specified in the lowest level axis then the memory consumption would be high.
- If columns are connected with the associated attribute dimension from the source model, then, the same attribute dimension must be selected in the MDX query.
- The script of the MDX query can be present on the client computer or the server.

Data Extraction Using Calculation Scripts

Calculation scripts provide a faster option to extract data from an Essbase application. However, before you extract data using the calculation scripts, take note of these restrictions:

- Data extraction using calculation scripts is supported ONLY for BSO applications.
- Data extraction using calculation scripts is supported ONLY for the Essbase Release 9.3 and later.
- Set the DataExportDimHeader option to ON.
- (If used) Match the DataExportColHeader setting to the data column dimension (in case of multiple data columns extraction).
- The Oracle Data Integrator Agent, which is used to extract data, must be running on the same machine as the Essbase server.
- When accessing calculation scripts present on the client computer, a fully qualified path to the file must be provided, for example, C:\Essbase_Samples\Calc_Scripts \calcall.csc, where as, to access calculation scripts present on the server, only the file name is sufficient.

18.5.3.2 Extracting Essbase Data

Oracle Data Integrator provides the LKM Hyperion Essbase DATA to SQL for extracting data from an Essbase application.

You can extract data for selected dimension members that exist in Essbase. You must set up the Essbase application before you can extract data from it.

Table 18–5 provides the options of the LKM Hyperion Essbase Data to SQL. These options define how Oracle Data Integrator Adapter for Essbase extracts data.

Table 18–5 LKM Hyperion Essbase DATA to SQL Options

Option	Values	Description
PRE_CALCULATION_SCRIPT	Blank (Default)	(Optional) Specify the calculation script that you want to run before extracting data from the Essbase cube.
EXTRACTION_QUERY_TYPE	■ ReportScript (Default) ■ MDXQuery ■ CalcScript	Specify an extraction query type—report script, MDX query, or calculation script. Provide a valid extraction query, which fetches all the data to fill the output columns. The first record (first two records in case of calculation script) contains the meta information of the extracted data.
EXTRACTION_QUERY_FILE	Blank (Default)	Specify a fully qualified file name of the extraction query.
EXT_COL_DELIMITER	\t (Default)	Specify the column delimiter for the extraction query. If no value is specified for this option, then space (“ “) is considered as column delimiter.
EXTRACT_DATA_FILE_IN_CALC_SCRIPT	Blank (Default)	This option is only applicable if the query type in the EXTRACTION_QUERY_TYPE option is specified as CalcScript. Specify a fully qualified file location where the data is extracted through the calculation script..
PRE_EXTRACT_MAXL	Blank (Default)	Enable this option to execute a MAXL script before extracting data from the Essbase cube.
POST_EXTRACT_MAXL	Blank (Default)	Enable this option to execute a MAXL script after extracting data from the Essbase cube.
ABORT_ON_PRE_MAXL_ERROR	■ No (Default) ■ Yes	This option is only applicable if the PRE_EXTRACT_MAXL option is enabled. If the ABORT_ON_PRE_MAXL_ERROR option is set to Yes, while executing pre-MAXL script, the load process is aborted on encountering any error.
LOG_ENABLED	■ No (Default) ■ Yes	If this option is set to Yes, during the LKM process, logging is done to the file specified in the LOG_FILE_NAME option.
LOG_FILENAME	<?=java.lang.System.getProperty (“java.io.tmpdir”)?/<%=snpRef.getTargetTable("RES_NAME")%>.log (Default)	Specify a file name to log events of the LKM process.

Table 18–5 (Cont.) LKM Hyperion Essbase DATA to SQL Options

Option	Values	Description
MAXIMUM_ERRORS_ALLOWED	1 (Default	Enable this option to set the maximum number of errors to be ignored before stopping extract.
LOG_ERRORS	■ No (Default) ■ Yes	If this option is set to Yes, during the LKM process, details of error records are logged to the file specified in the ERROR_LOG_FILENAME option.
ERROR_LOG_FILENAME	<?=java.lang.System.getProperty(java.io.tmpdir")?>/<%=snpRef.getTargetTable("RES_NAME")%>.err	Specify a file name to log error record details of the LKM process.
ERR_LOG_HEADER_ROW	■ No (Default) ■ Yes	If this option is set to Yes, then the header row containing the column names are logged to the error records file.
ERR_COL_DELIMITER	, (Default)	Specify the column delimiter to be used for the error records file.
ERR_ROW_DELIMITER	\r\n (Default)	Specify the row delimiter to be used for the error records file.
ERR_TEXT_DELIMITER	' (Default)	Specify the text delimiter to be used for the column data in the error records file. For example, if the text delimiter is set as ' " ' (double quote), then all the columns in the error records file are delimited by double quotes.
DELETE_TEMPORARY_OBJECTS	■ No (Default) ■ Yes	Set this option to No, in order to retain temporary objects (tables, files, and scripts) after integration. This option is useful for debugging.

18.5.3.3 Extracting Members from Metadata

Oracle Data Integrator provides the LKM Hyperion Essbase METADATA to SQL for extracting members from a dimension in an Essbase application.

To extract members from selected dimensions in an Essbase application, you must set up the Essbase application and load metadata into it before you can extract members from a dimension.

Before extracting members from a dimension, ensure that the dimension exists in the Essbase database. No records are extracted if the top member does not exist in the dimension.

Table 18–6 lists the options of the LKM Hyperion Essbase METADATA to SQL. These options define how Oracle Data Integrator Adapter for Oracle's Hyperion Essbase extracts dimension members.

Table 18–6 LKM Hyperion Essbase METADATA to SQL

Option	Values	Description
MEMBER_FILTER_CRITERIA	IDescendants, (Default)	Enable this option to select members from the dimension hierarchy for extraction. You can specify these selection criteria: ■ IDescendants ■ Descendants ■ IChildren ■ Children ■ Member_Only ■ Level0 ■ UDA
MEMBER_FILTER_VALUE	Blank (Default)	Enable this option to provide the member name for applying the specified filter criteria. If no member is specified, then the filter criteria is applied on the root dimension member. If the MEMBER_FILTER_CRITERIA value is MEMBER_ONLY or UDA, then the MEMBER_FILTER_VALUE option is mandatory and cannot be an empty string.
LOG_ENABLED	■ No (Default) ■ Yes	If this option is set to Yes, during the LKM process, logging is done to the file specified by the LOG_FILE_NAME option.
LOG_FILE_NAME	<?=java.lang.System.getProperty (java.io.tmpdir")?>/Extract_<% =snpRef.getFrom()%>.log	Specify a file name to log events of the LKM process.
MAXIMUM_ERRORS_ALLOWED	1 (Default)	Enable this option to set the maximum number of errors to be ignored before stopping extract.
LOG_ERRORS	■ No (Default) ■ Yes	If this option is set to Yes, during the LKM process, details of error records are logged to the file specified in the ERROR_LOG_FILENAME option.
ERROR_LOG_FILENAME	<?=java.lang.System.getProperty (java.io.tmpdir")?>/Extract_<% =snpRef.getFrom()%>.err	Specify a file name to log error record details of the LKM process.
ERR_LOG_HEADER_ROW	■ No (Default) ■ Yes	If this option is set to Yes, then the header row containing the column names are logged to the error records file.
ERR_COL_DELIMITER	, (Default)	Specify the column delimiter to be used for the error records file.
ERR_ROW_DELIMITER	\r\n (Default)	Specify the row delimiter to be used for the error records file.

Table 18–6 (Cont.) LKM Hyperion Essbase METADATA to SQL

Option	Values	Description
ERR_TEXT_DELIMITER	■ Blank (Default) ■ \" ■ \"	Specify the text delimiter to be used for the data column in the error records file. For example, if the text delimiter is set as ' " ' (double quote), then all the columns in the error records file are delimited by double quotes.
DELETE_TEMPORARY_OBJECTS	■ No (Default) ■ Yes	Set this option to No, in order to retain temporary objects (tables, files, and scripts) after integration. This option is useful for debugging.

19

Oracle Hyperion Financial Management

This chapter describes how to work with Oracle Hyperion Financial Management in Oracle Data Integrator.

This chapter includes the following sections:

19.1 Introduction

Oracle Data Integrator Adapter for Hyperion Financial Management enables you to connect and integrate Hyperion Financial Management with any database through Oracle Data Integrator. The adapter provides a set of Oracle Data Integrator Knowledge Modules (KMs) for loading and extracting metadata and data and consolidating data in Financial Management applications.

19.1.1 Integration Process

You can use Oracle Data Integrator Adapter for Hyperion Financial Management to perform these data integration tasks on a Financial Management application:

- Load metadata and data
- Extract data
- Consolidate data
- Enumerate members of member lists

Using the adapter to load or extract data involves these tasks:

- Setting up an environment: defining data servers and schemas

  See Section 19.3, "Setting up the Topology".

- Reverse-engineering a Financial Management application using the Reverse-engineering Knowledge Module (RKM)

  See Section 19.4, "Creating and Reverse-Engineering a Financial Management Model".

- Loading metadata and data using Integration Knowledge Modules (IKM)

  See Section 19.5, "Designing an Interface"

- Extracting data and members using Load Knowledge Modules (LKM)

  See Section 19.5, "Designing an Interface"

19.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 19–1 for handling Hyperion Financial Management data. These KMs use Hyperion Financial Management specific features. It is also possible to use the generic SQL KMs with the Financial Management database. See Chapter 4, "Generic SQL" for more information.

Table 19–1 Hyperion Financial Management Knowledge Modules

Knowledge Module	Description
RKM Hyperion Financial Management	Reverse-engineers Financial Management applications and creates data models to use as targets or sources in Oracle Data Integrator interfaces.
IKM SQL to Hyperion Financial Management Data	Integrates data into Financial Management applications.
IKM SQL to Hyperion Financial Management Dimension	Integrates metadata into Financial Management applications.
LKM Hyperion Financial Management Data to SQL	Loads data from a Financial Management application to any SQL compliant database used as a staging area. This knowledge module will not work if you change the column names of the HFMData data store reverse engineered by the RKM Hyperion Financial Management knowledge module.
LKM Hyperion Financial Management Members To SQL	Loads member lists from a Financial Management application to any SQL compliant database used as a staging area.

19.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Oracle Data Integrator Adapter for Financial Management:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

19.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

19.2.2 Technology Specific Requirements

There are no technology-specifc requirements for using the Oracle Data Integrator Adapter for Financial Management.

19.2.3 Connectivity Requirements

There are no connectivity-specific requirements for using the Oracle Data Integrator Adapter for Financial Management.

19.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating an Hyperion Financial Management Data Server
2. Creating an Hyperion Financial Management Physical Schema

19.3.1 Creating an Hyperion Financial Management Data Server

Create a data server for the Hyperion Financial Management technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Hyperion Financial Management data server:

1. In the Definition tab:
   - Name: Enter a name for the data server definition.
   - Cluster (Data Server): Enter the Financial Management cluster name.
2. Under Connection, enter a user name and password for connecting to the Financial Management server.

Note: The Test button does not work for a Hyperion Financial Management data server connection; it works only for relational technologies that have a JDBC driver.

19.3.2 Creating an Hyperion Financial Management Physical Schema

Create a Hyperion Financial Management physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Under Application (Catalog), specify a FinancialManagement application.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

19.4 Creating and Reverse-Engineering a Financial Management Model

This section contains the following topics:

- Create an Financial Management Model
- Reverse-Engineer an Financial Management Model

19.4.1 Create an Financial Management Model

Create an Financial Management Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

19.4.2 Reverse-Engineer an Financial Management Model

Reverse-engineering a Financial Management application creates an Oracle Data Integrator model that includes a data store for each dimension in the application, a data store for data, an optional data store for data with multiple periods, and an EnumMemberList data store.

To perform a Customized Reverse-Engineering on Hyperion Financial Management with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Hyperion Financial Management technology.

1. In the Reverse tab of the Financial Management Model, select the RKM Hyperion Financial Management.
2. Set the KM options as follows:
   - CREATE_HFMDATA_MULTIPLEPERIODS: Set to `Yes` to create an additional data store for data with multiple periods. The number of periods for that model is specified by the MULTIPERIOD_COUNT option.

     Default is `No`.
   - MULTIPERIOD_COUNT: Specifies the number of periods for the HFMData_MultiplePeriod table.

The RKM connects to the application (which is determined by the logical schema and the context) and imports some or all of these data stores, according to the dimensions in the application:

- HFMData: For loading and extracting data
- HFMData_MultiplePeriods: For data with the number of periods specified by the option MULTIPERIODS_COUNT

Note: This data store is imported only if the CREATE_HFMDATA_MULTIPLEPERIODS option is set to Yes in the model definition.

- Account: For loading the Account dimension.
- Entity: For loading the Entity dimension.
- Scenario: For loading the Scenario dimension.
- Currency: For loading the currency dimension.
- Custom1-4: For loading the Custom1-4 dimensions.
- EnumMembersList-For extracting a members list.

See Section 19.6, "Data Store Tables" for more information about these tables.

19.5 Designing an Interface

After reverse-engineering a Financial Management application as a model, you can use the data stores in this model in these ways:

- As targets of interfaces for loading data and metadata into the application The following figure shows the flow of an interface targeting Financial Management.
- As sources of interfaces for extracting data and member lists from the application The following figure shows the flow of an interface with a Financial Management source.

The KM choice for an interface determines the abilities and performance of this interface. The recommendations in this section help in the selection of the KM for different situations concerning Hyperion Financial Management.

This section contains the following topics:

- Loading Metadata
- Loading Data
- Extracting Data

19.5.1 Loading Metadata

Oracle Data Integrator provides the IKM SQL to Hyperion Financial Management Dimension for loading metadata into a Finanacial Management application.

Metadata comprises dimension members. You must load members, or metadata, before you load data values for the members.

You can load members only to existing Financial Management dimensions. You must use a separate interface for each dimension that you load. You can chain interfaces to load metadata into several dimensions at once.

The IKM SQL to Hyperion Financial Management Dimension supports the following options for defining how the adapter loads metadata into a Financial Management application:

- REPLACE_MODE: If set to `Yes`, metadata is replaced in the application (Replace); if set to No, metadata is overwritten in the application (Merge). Valid values: Yes or `No` (default).
- CLEAR_ALL_METADATA_BEFORE_LOADING: If set to `Yes`, all metadata is cleared before loading. Valid values: `Yes` or `No` (default).

Caution: If you set this option to Yes, you lose any active data or journals in the application.

- LOG_ENABLED: If set to `Yes`, logging is done during the load process to the file specified by theLOG_FILE_NAME option. Valid values: `Yes` or `No` (default).
- LOG_FILE_NAME: The name of the file where logs are saved; default: `Java temp folder/dimension.log`

19.5.2 Loading Data

Oracle Data Integrator provides the IKM SQL to Hyperion Financial Management Data for loading data into a Financial Management application.

You can load data into selected dimension members that are already created in Financial Management. You must set up the Financial Management application before you can load data into it.

Before loading data, ensure that the members (metadata) exist in the Financial Management relational database. A data load fails if the members do not exist.

Note: Use the HFMData or HFMData_MultiplePeriods data stores from a Hyperion Financial Management model as the target data store of your integration interface.

The IKM SQL to Hyperion Financial Management Data supports the following options for defining how the adapter loads and consolidates data in a Financial Management application:

- IMPORT_MODE: Determines how data in the application cells is handled during data load. Valid values are:
  - `Merge` (default): For each unique point of view that exists in the load data and in the application, the load data overwrites the data in the application. For each unique point of view that is in the load data but not in the application, the load data is loaded into the application.
  - `Replace`: For each unique point of view in the load data, the system clearscorresponding values from the application, and then the data is loaded.

Note: Unless the connected user has full access rights to all specified cells, no data is changed.

  - `Replace by Security`: For each unique point of view in the load data to which the user has full access rights, the system clears corresponding values from the application, and then the data is loaded. Cells to which the user lacks full access are ignored.
  - `Accumulate`:For each unique point of view that exists in the load data and in the application, the value from the load data is added to the value in the application.
- ACCUMULATE_WITHIN_FILE: If set to `Yes`, multiple values for the same cells in the load data are added before they are loaded into the application. Valid values: `Yes` or `No` (default).
- FILE_CONTAINS_SHARE_DATA: Set to `Yes` if the load file contains ownership data, such as shares owned. Valid values: `Yes` or `No` (default).

Caution: If ownership data is included in the file and this option is set to No, an error occurs when you load the file.

- CONSOLIDATE_AFTER_LOAD: If set to `Yes`, data is consolidated after being loaded. Valid values: Yes or No (default).
- CONSOLIDATE_ONLY: If set to `Yes`, data is consolidated but not loaded. Valid values: `Yes` and `No` .
- CONSOLIDATE_PARAMETERS: Specifies the parameters for consolidation as comma-separated values in this order: Scenario (required), Year, Period, Parent.Entity, and Type; default: an empty string.

  Valid Type parameter settings:
  - "I" = Consolidate

- "D" = Consolidate All with Data
- "A" = Consolidate All
- "C" = Calculate Contribution
- "F"= Force Calculate Contribution

  Example: `Actual,1999,2,EastRegion.EastSales,A`

- LOG_ENABLED: If set to `Yes`, logging is done during the load process to the file specified by the LOG_FILE_NAME option. Valid values: `Yes` or `No` (default)
- LOG_FILE_NAME: The name of the file where logs are saved; default: `Java temp folder/HFMData.logor HFMData_MultiplePeriod.log.`

19.5.3 Extracting Data

You can extract data for selected dimension members that exist in Financial Management. You must set up the Financial Management application before you can extract data from it.

Before extracting data, ensure that the members (metadata) exist in the Financial Management relational database; no records are extracted for members that do not exist (including the driver member and the members specified in the point of view.)

This section includes the following topics:

- Extracting Financial Management Data
- Extracting Members from Member Lists

19.5.3.1 Extracting Financial Management Data

Oracle Data Integrator provides the LKM Hyperion Financial Management Data to SQL for extracting data from an Essbase application.

Use as a source the source data store (HFMData) from a Hyperion Financial Management model.

LKM Hyperion Financial Management Data to SQL supports the following options for defining how Oracle Data Integrator Adapter for Hyperion Financial Management extracts data:

- SCENARIO_FILTER: The Scenario dimension members for which you are exporting data.

  You can specify comma-delimited Scenario members or one scenario. If you do not specify scenarios, the system exports data for all scenarios.

- YEAR_FILTER: The Year dimension members for which you are exporting data

  You can specify comma-delimited years or one year. If you do not specify years, the system exports data for all years.

- PERIOD_FILTER: The set of Period dimension members for which you are exporting data.

  Specify a range of members using the ~ character between start and end period numbers; for example, 1~12. If you do not specify periods, the system exports data for only the first period.

- ENTITY_FILTER: The Entity dimension members for which you are exporting data

You can specify comma-delimited entities or one entity. To specify the parent and child, separate them with a period; for example, `I.Connecticut`. If you do not specify entities, the system exports data for all entities.

- ACCOUNT_FILTER: The Account dimension members for which you are exporting data.

  You can specify comma-delimited accounts or one account. If you do not specify accounts, the system exports data for all accounts.

- VIEW_FILTER: The View dimension member for which you are exporting data Possible values: `Periodic, YTD, or <Scenario_View> (default)`
- LOG_ENABLED: If set to `Yes`, logging is done during the extract process to the file specified inLOG_FILE_NAME
- LOG_FILE_NAME: The name of the file where logs are saved
- DELETE_TEMPORARY_OBJECTS: If set to `Yes` (default), tables, files, and scripts are deleted after integration.

Tip: Temporary objects can be useful for resolving issues.

19.5.3.2 Extracting Members from Member Lists

Oracle Data Integrator provides the LKM Hyperion Financial Management Members to SQL for extracting members from a dimension in an Essbase application.

You can extract members fromselected member lists and dimensions in a Financial Management application. You must set up the Financial Management application and load member lists into it before you can extract members from a member list for a dimension.

Before extracting members from a member list for a dimension, ensure that the member list and dimension exist in the Financial Management relational database. No records are extracted if the top member does not exist in the dimension.

Use as a source the source data store (EnumMembersList) from a Hyperion Financial Management model.

The LKM Hyperion Financial Management Members to SQL supports the following options for defining how Oracle Data Integrator Adapter for Hyperion Financial Management extracts members of member lists:

- DIMENSION_NAME: The name of the dimension for which you are creating a member list; required.
- MEMBER_LIST_NAME: A label for the member list; required.
- TOP_MEMBER: The top member of the member list.
- LOG_ENABLED: If set to `Yes`, logging is done during the extract process to the file specified by the LOG_FILE_NAME option. Valid values: `Yes` and `No` (default) .
- LOG_FILE_NAME: The name of the file where logs are saved.
- DELETE_TEMPORARY_OBJECTS: If set to `Yes` (default), tables, files, and scripts are deleted after integration.

Tip: Temporary objects can be useful for resolving issues.

19.6 Data Store Tables

The IKM SQL to Hyperion Financial Management loads columns in tables to create data stores. The following tables describe the columns in each data store:

- HFMData
- HFMData_MultiplePeriods
- Account
- Entity
- Scenario
- Currency
- Custom1-4
- EnumMembersList

Note: In the following tables, the column types are String unless the column descriptions specify otherwise.

For Table 19–2 note that if custom dimensions have aliases, the aliases (rather than Custom*N*) are displayed as column names.

Table 19–2 HFMData

Column	Description
Scenario	A Scenario dimension member; example: `Actual`
Year	A Year dimension member; example: `2000`
Entity	An Entity dimension member, in `parent.child` format. For example: `United States.NewYork` to specify member NewYork as a child of member United States.
Account	An Account dimension member; example: `Sales`
Value	A Value dimension member; example: `USD`
ICP	An Intercompany Partner dimension member; example: `[ICP Entities]`
Custom1	A Custom1 dimension member; example: `AllCustomers`
Custom2	A Custom2 dimension member
Custom3	A Custom3 dimension member
Custom4	A Custom4 dimension member
Period	A Period dimension member
Data Value	The value associated with the intersection. This value is passed as a Double.
Description	A description of the data value

For Table 19–3 note that if custom dimensions have aliases, the aliases (rather than Custom*N*) are displayed as column names.

Table 19–3 HFMData_MultiplePeriods

Column	Description
Scenario	A Scenario dimension member; example: `Actual`
Year	A Year dimension member; example: `2000`
Entity	An Entity dimension member, in `parent.child` format. For example: `United States.NewYork` to specify member NewYork as a child of member United States.
Account	An Account dimension member; example: `Sales`
Value	A Value dimension member; example: `USD`
ICP	An Intercompany Partner dimension member; example: `[ICP Entities]`
Custom1	A Custom1 dimension member; example: `AllCustomers`
Custom2	A Custom2 dimension member
Custom3	A Custom3 dimension member
Custom4	A Custom4 dimension member
Period1..*n*	For every data value being loaded, a period must be specified. The number of periods to be loaded for each intersection is specified when the Hyperion Financial Management model is reversed. A period column is created for each specified period.
Data Value1..*n*	Data values to be loaded. The number of periods to be loaded for each intersection is specified when the Hyperion Financial Management model is reversed. A data value column is created for each specified period. This value is passed as a Double.
Description1..*n*	A description for each data value

Table 19–4 Account

Column	Description
Member	An account table; required
Description	A description for the account; required
Parent Member	The parent account member
Account Type	Required; Valid account types: ■ ASSET ■ LIABILITY ■ REVENUE ■ EXPENSE ■ FLOW ■ BALANCE ■ BALANCERECURRING ■ CURRENCYRATE ■ GROUPLABEL ■ DYNAMIC
Is Calculated	Whether the account is calculated. Valid values: `Y` if the account is calculated, or `N` (default) if it is not calculated and manual input is enabled

Table 19–4 (Cont.) Account

Column	Description
Is Consolidated	Whether the account is consolidated into a parent account Valid values: Y if the account is consolidated into a parent, or N (default) if it is not.
Is ICP	Whether intercompany transactions are allowed for this account. Valid values: ■ Y if ICP transactions, including self-ICP transactions, are allowed ■ N (default) if ICP transactions are not allowed ■ R if ICP transactions are allowed but the account is restricted from having ICP transactions with itself If you specify Y or R, enter the name of the ICP TopMember. If you do not enter the top member, the default, [ICP TOP], is used.
Plug Account	The name of the account used for identifying discrepancies in intercompany transactions; required if intercompany transactions are allowed for this account.
Custom 1...4 TopMember	The top member in the hierarchy of a Custom dimension that is valid for the account. The specified member, including all of its parents and descendants, is valid for the account. All other members of the Custom dimension are not valid for the account. These columns required if intercompany transactions are allowed for this account.
Number of Decimal Places	The number of digits to display to the right of the decimal point for the account values; required. Specify an integer from 0 (default) to 9.
Use Line Items	Whether the account can have line items.Valid values: Y if the account uses line items, or N (default) if it does not.
Aggr Custom 1...4	Whether aggregation is enabled for intersections of the account and the Customdimensions. This column is used for special totals, not summing. Valid values: Y (default) if the account is allowed to aggregate with Custom dimensions, or N if it is not .
User Defined 1...3	Optional custom text for the account
XBRL Tag	Optional XBRL tag for the account
Security Class	The name of the security class that defines users who can access the account data. Default: DEFAULT security class.
ICP Top Member	The top member of the ICP group assigned to the account
Enable Data Audit	Whether data auditing is enabled for the account. Valid values: Y (default) to enable auditing, or N to disable auditing
Description 2...10	Optional additional descriptions for the account

Table 19–5 Entity

Column	Description
Member	An entity label; required
Description	A description for the entity; required

Table 19–5 (Cont.) Entity

Column	Description
Parent Member	The parent entity member
Default Currency	The default currency for the entity; required.
Allow Adj	Valid values: Y if journal postings are permitted, or N (default) if journal entries are not permitted.
Is ICP	Valid values: Y if the entity is an intercompany entity, or N (default) if it is not. **Note**: An intercompany entity is displayed in the POV in the ICP dimensions under [ICP Entities].
Allow Adj From Child	Valid values: Y if journal postings from children of this parent entity are permitted, or N (default) if they are not.
Security Class	The name of the security class that defines users who can access the entity's data. Default: `DEFAULT security class`.
User Defined 1...3	Optional custom text for the entity
Holding Company	The holding company for the entity. Valid values: Any valid entity or blank (default).
Description 2...10	Optional additional descriptions for the entity

Table 19–6 Scenario

Column	Description
Member	A scenario label; required
Description	A description for the scenario; required
Parent Member	The parent Scenario member
Default Frequency	Period types for which data input is valid for the scenario; required.
Default View	Whether the view is YTD or Periodic; required.
Zero View Non Adj	Whether the view is YTD or Periodic when missing, nonadjusted data values exist; required.
Zero View Adj	Whether the view is YTD or Periodic when missing, adjusted data values exist; required.
Consol YTD	The view for consolidations; required Valid values: Y for YTD, or N for Periodic
Support PM	Whether Process Management command is enabled in Data Explorer; required. Valid values: Y to enable Process Management, or N to disable Process Management
Security Class	The name of the security class that defines users who can access the scenario data. Default: `DEFAULT security class`.
Maximum Review Level	The maximum process management review level for the scenario. Enter an integer from 1 to 10.
Use Line Items	Valid values: Y if the scenario can accept line items, or N (default) if it cannot.
Enable Data Audit	Valid values: Y to enable auditing, or N (default) to disable auditing.

Table 19–6 (Cont.) Scenario

Column	Description
Def Freq For IC Trans	The default frequency for intercompany transactions. Enter a string that identifies a valid frequency for the application. The default value is an empty string, representing no default frequency.
User Defined 1...3	Optional custom text for the scenario
Description 2...10	Optional additional descriptions for the scenario

Table 19–7 Currency

Column	Description
Member	A currency label; required
Description	A description for the currency; required
Scale	The unit in which amounts are displayed and stored for the currency, which identifies where the decimal point is placed; required Must be one of the following valid integer values: ■ `Blank` = None ■ `0` = Units ■ `1` = Tens ■ `2` = Hundreds ■ `3` = Thousands ■ `4` = Ten Thousands ■ `5` = Hundred Thousands ■ `6` = Millions ■ `7` = Ten Millions ■ `8` = Hundred Millions ■ `9` = Billions
Translation Operator	Whether conversions for the currency are calculated by multiplying or dividing the translation rate. Valid values: `D` to divide (default) or `M` to multiply
Description 2...10	Optional additional descriptions for the currency

Table 19–8 Custom1-4

Column	Description
Member	The label of a custom dimension member; required
Description	A description for the custom dimension member; required
Parent Member	The parent custom member; required
Is Calculated	Whether the base-level custom account is calculated. If a base-level custom account is calculated, you cannot manually enter values. Valid values: Y if the account is calculated, N if it is not calculated.

Table 19–8 (Cont.) Custom1-4

Column	Description
Switch Sign	Whether the sign is changed (Debit/Credit) for FLOW accounts using the following rules: ■ ASSET to LIABILITY ■ LIABILITY to ASSET ■ EXPENSE to REVENUE ■ REVENUE to EXPENSE ■ BALANCE to FLOW ■ FLOW to BALANCE Valid values: `Y` if the account type is switched, or `N` if it is not switched
Switch Type	The account type change for FLOW accounts, following these rules: ■ ASSET to EXPENSE ■ EXPENSE to ASSET ■ LIABILITY to REVENUE ■ REVENUE to LIABILITY ■ BALANCE to FLOW ■ FLOW to BALANCE Valid values: `Y` if the account type is switched, or `N` if it is not switched
Security Class	The name of the security class that defines users who can access the custom dimension member data. Default: `DEFAULT security class`.
User Defined 1...3	Optional custom text for the custom dimension member
Aggr Weight	The aggregation weight for the custom dimensions; passed as Double Default: 1
Description 2...10	Optional additional descriptions for the custom dimension member

Table 19–9 EnumMembersList

Column	Description
Member	The members of the member list

20

Oracle Hyperion Planning

This chapter describes how to work with Oracle Hyperion Planning in Oracle Data Integrator.

This chapter includes the following sections:

20.1 Introduction

Oracle Data Integrator Adapter for Hyperion Planning enables you to connect and integrate Oracle's Hyperion Planning with any database through Oracle Data Integrator. The adapter provides a set of Oracle Data Integrator Knowledge Modules (KMs) for loading metadata and data into Planning, Oracle's Hyperion Workforce Planning, and Oracle's Hyperion Capital Expense Planning applications.

20.1.1 Integration Process

Loading a Planning application with metadata and data using Oracle Data Integrator Adapter for Hyperion Planning involves these tasks:

- Setting up an environment: Defining data servers and schemas

  See Section 20.3, "Setting up the Topology".

- Reverse-engineering a Planning application using the adapter's Reverse-engineering Knowledge Module (RKM)

  See Section 20.4, "Creating and Reverse-Engineering a Planning Model".

- Loading metadata and data into the Planning application using the adapter's Integration Knowledge Module (IKM)

  See Section 20.5, "Designing an Interface".

20.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 20–1 for handling Hyperion Planning data. These KMs use Hyperion Planning specific

features. It is also possible to use the generic SQL KMs with the Hyperion Planning database. See Chapter 4, "Generic SQL" for more information.

Table 20–1 Hyperion Planning Knowledge Modules

Knowledge Module	Description
RKM Hyperion Planning	Reverse-engineers Planning applications and creates data models to use as targets in Oracle Data Integrator interfaces. Each dimension (standard dimension and attribute dimension) is reversed as a datastore with the same name as the dimension with appropriate columns. Creates a datastore named "UDA" for loading UDA's.
IKM SQL to Hyperion Planning	Loads metadata and data into Planning applications.

20.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Oracle Data Integrator Adapter for Planning:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

20.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

20.2.2 Technology Specific Requirements

There are no technology-specifc requirements for using the Oracle Data Integrator Adapter for Planning.

20.2.3 Connectivity Requirements

There are no connectivity-specific requirements for using the Oracle Data Integrator Adapter for Planning.

20.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating an Hyperion Planning Data Server
2. Creating an Hyperion Planning Physical Schema

20.3.1 Creating an Hyperion Planning Data Server

Create a data server for the Hyperion Planning technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a Hyperion Planning data server:

1. In the Definition tab:
   - Name: Enter a name for the data server definition.
   - Server (Data Server): Enter the Planning application host name and RMI port number in this format: `<host>:<port>`.
2. Under Connection, enter a user name and password for connecting to the Planning server.

> **Note:** The Test button does not work for a Hyperion Planning data server connection. This button works only for relational technologies that have a JDBC Driver.

20.3.2 Creating an Hyperion Planning Physical Schema

Create a Hyperion Planning physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Under a data server, you can define a physical schema corresponding to an application and the logical schemas on which models are based.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

20.4 Creating and Reverse-Engineering a Planning Model

This section contains the following topics:

20.4.1 Create a Planning Model

Create a Planning Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*

20.4.2 Reverse-engineer a Planning Model

Reverse-engineering a Planning application creates an Oracle Data Integrator model that includes a datastore for each dimension in the application. Note that the Year/Period/Version/Scenario are not reverse-engineered.

To perform a Customized Reverse-Engineering on Hyperion Planning with a RKM, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Hyperion Planning technology.

1. In the Reverse tab of the Planning Model, select the RKM Hyperion Planning.

The RKM connects to the application (which is determined by the logical schema and the context) and imports the following items:

- A datastore for each dimension in the application, with the same name as the dimension
- A datastore called UDA, for UDA loading

20.5 Designing an Interface

After reverse-engineering a Planning application as a model, you can use the datastores in this model as targets of interfaces for loading data and metadata into the application.

The KM choice for an interface determines the abilities and performance of this interface. The recommendations in this section help in the selection of the KM for different situations concerning Hyperion Planning.

This section contains the following topics:

- Loading Metadata
- Loading Data
- Load Options

20.5.1 Loading Metadata

Oracle Data Integrator provides the IKM SQL to Hyperion Planning for loading metadata into a Planning application.

Metadata consists of dimension members. You must load members, or metadata, before you load data values for the members. For example, before loading salary data for five new employees, you load the employees (as members) to the Planning relational database before you load the data to the Oracle's Hyperion Essbase database.

You can load members only to dimensions that exist in Planning. You must use a separate interface for each dimension that you load. You can chain interfaces to load metadata into several dimensions at once.

Note: You must refresh the Essbase database after loading the dimension members in the application. The Essbase database is refreshed if you set the REFRESH_DATABASE option in IKM SQL to Hyperion Planning to Yes. See "Load Options" on page 18.

To load metadata into a Planning application:

1. Create an interface. Make sure that you select the IKM SQL to Hyperion Planning on the Flow tab.
2. Specify the load options as described in Section 20.5.3, "Load Options".
3. Run the interface to load the metadata into the application
4. Validate the dimension:
   a. Log on to Planning Web.
   b. Select **Administration** > **Dimensions**.

20.5.2 Loading Data

Oracle Data Integrator provides the IKM SQL to Hyperion Planning for loading data into a Planning application.

You can load data into selected dimension members that are already created in Planning. You must set up the Planning, Workforce Planning, or Capital Expense Planning application before you can load data into it.

Before loading data, ensure that the members (metadata) exist in the Planning relational database and the Essbase database. A data load fails if the members do not exist. (This includes the driver member and the members specified in the point of view.) If necessary, load metadata and refresh the Essbase database to synchronize the members.

Before loading data into a Planning, Workforce Planning, or Capital Expense Planning application, you must set up the relevant data load and driver dimensions in Planning. After you set up the data load and driver dimensions in Planning, you must determine the point of view for the members whose data you are loading.

To load data into a Planning application:

1. In Planning, specify parameters for data to load:
   a. Select **Administration** > **Data Load Administration**.
   b. For Available Data Load Dimensions, select a dimension, and click **Go**.
   c. For Available Driver Dimensions, select the dimension to which you are loading data in an Essbase database; for example, select the Account dimension.
   d. Select the members of the driver dimension to load with data.

      After the Hyperion Planning data load is set up, use Hyperion Planning RKM to perform the reverse-engineering process. Reverse-engineering retrieves and updates the datastore for the data load dimension with additional columns (fields) required for the data load.
   e. Click **Save**.
2. In Oracle Data Integrator Studio, run an interface for loading data.

   Note: You can use the same interface for loading metadata and data. Section 20.5.3, "Load Options" lists the options of the IKM SQL to Hyperion Planning
3. Check the Operator log to see if the interface ran successfully.
4. To validate the data load, use either method:
   - Create a Planning data form to retrieve data.
   - Check Oracle's Essbase Administration Services to ensure that blocks were created in the appropriate cube.

20.5.3 Load Options

IKM SQL to Hyperion Planning supports these options for defining how Oracle Data Integrator Adapter for Hyperion Planning loads data:

- LOAD_ORDER_BY_INPUT

Possible values: Yes or No; default: No If set to Yes, members are loaded in the same order as in the input records.

- SORT_PARENT_CHILD

  Possible values: Yes or No; default: No If set to Yes, incoming records are sorted so that all parents are inserted before children.

- LOG_ENABLED

  Possible values: Yes or No; default: No If set to Yes, logging is done during the load process to the file specified by the LOG_FILE_NAME option.

- LOG_FILE_NAME

  The name of the file where logs are saved; default value:Java temp folder/ dimension.log

- MAXIMUM_ERRORS_ALLOWED

  Maximum number of errors before the load process is stopped; default value: 0

  If set to 0 or a negative number, the load process is not stopped regardless of the number of errors.

- LOG_ERRORS

  Possible values: Yes or No; default: No

  If set to Yes, error records are loggedto the file specified by the ERROR_LOG_FILE property.

- ERROR_LOG_FILE

  The name of the file where error records are logged; default value: Java temp folder/ dimension.err

- ERR_COL_DELIMITER

  The column delimiter used for the error record file; default value: comma (,)

- ERR_ROW_DELIMITER

  The row delimiter used for the error record file; default value: \r\n

  Note: Row and column delimiters values can also be specified in hexadecimal. A value that starts with 0x is treated as hexadecimal; for example, 0x0041 is treated as the letter A.

- ERR_TEXT_DELIMITER

  The text delimiter to be used for the column values in the error record file

- ERR_LOG_HEADER_ROW:

  Possible values: Yes or No; default: Yes

  If set to Yes, the row header (with all column names) is logged in the error records file.

- REFRESH_DATABASE:

  If set to Yes, completion of the load operation invokes a cube refresh.

  Possible values: Yes or No; default: No

20.6 Datastore Tables and Data Load Columns

IKM SQL to Hyperion Planning loads columns in tables to create datastores. The following topics describe the columns in each datastore:

- Accounts
- Employee
- Entities
- User-Defined Dimensions
- Attribute Dimensions
- UDA

Data Load Columns are columns used for loading data into dimensions.

20.6.1 Accounts

Table 20–2 describes the columns of the Accounts table. See Section 20.6.7, "Data Load Columns" for descriptions of additional columns that are displayed for loading Account dimension data if the application has been set up for data load in Planning.

Table 20–2 Accounts

Column	Description
Account	Takes the name of the account member you are loading. If this member exists, its properties are modified; otherwise, the record is added. This field is required. The value for this field must meet these requirements: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string.

Table 20–2 (Cont.) Accounts

Column	Description
Parent	Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension. When you load data for a member and specify a different parent member that from the parent member in the application, the member is updated with the parent value that you specify. Example: If Member 1 has a parent value of Member A in your Planning application and you load Member 1 with a parent value of Member B, your application is updated, and Member B becomes the parent of Member 1. Member 1 and its descendants are moved from Member A to Member B. If the column is left blank, it is ignored during the load. The record is not loaded if one of the following situations occurs: ■ The specified parent is a descendant of the member that you are loading. ■ The specified parent does not exist in the Planning application.
Default Alias	Takes an alternate name for the member being loaded. If you are modifying properties and do not specify a value, the alias is not changed in the Planning application. If you specify <NONE> or <none> as the value, the alias in the Planning application is deleted. The value for this column must meet the following requirements for a successful load: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string; default value: a null string.
Additional Alias	Can take an alternate name for the member being loaded. There will be as many Alias columns as there are Alias tables defined in Planning. The value for multiple alias columns must conform to the same requirements as those listed for the default alias column.
Data Storage	Takes the storage attribute for the member being loaded. Valid values: ■ Store ■ Dynamic Calc ■ Dynamic Calc and Store ■ Shared ■ Never Share (default) ■ Label Only This value is passed as a string.

Table 20–2 (Cont.) Accounts

Column	Description
Two Pass Calculation	Boolean value to indicate whether the member being loaded has the Two-Pass Calculation associated attribute. Valid values: 0 for False (default), or any other number for True. Values are valid only when the Data Storage value is Dynamic Calc or Dynamic Calc and Store; otherwise, the record is rejected.
Account Type	Takes the account type of the member that is being loaded. Valid values: Revenue, Expense, Asset, Liability, Equity, and Saved Assumption. The default is taken from the parent of the member that is being loaded, or it is Expense if the member is being added to the root dimension.
Time Balance	Takes a type for members with an account type of Saved Assumption only or when the record is rejected. Valid values: Flow, First, Balance, Average, and two averaging options, Actual_365 and Actual_Actual. (Actual_365 assumes the actual number of days in each month and 28 days in February; Actual_Actual accounts for 29 days in February during leap years.) The default is taken from the parent of the member being loaded or is Flow if the member is being added to the root dimension. This value is passed as a string. Default values of Time Balance for Account types: ■ Revenue-Flow ■ Expense-Flow ■ Asset-Balance ■ Liability-Balance ■ Equity-Balance Note: When Time Balance is Flow, records with any valid Skip Values are loaded, but Skip Value is disabled for all account types.
Skip Value	Skip ValueTakes the skip option that is set for the Time Balance property. When the Time Balance property is set to First, Balance, or Average, these Skip options are available: ■ None-Indicates that zeros and #missing value are considered when the parent value is calculated ■ Missing-Excludes #missing values when calculating parent values ■ Zeros-Excludes zero values when calculating parent values ■ Missing and Zeros-Excludes #missing and zero values when calculating parent values Note: When Time Balance is Flow, records with any valid Skip Values are loaded, but Skip Value is disabled for all Account types.

Table 20–2 (Cont.) Accounts

Column	Description
Data Type	Takes the data storage value. Valid values: ■ Currency-Stores and displays the member's data value in the default currency. ■ Non-currency-Stores and displays the member's data value as a numeric value. ■ Percentage-Stores data values as a numeric value and displays the member's data value as a percentage. ■ Smart list / enumeration-Stores data values as a numeric value and displays the member's data value as a string. ■ Date-Stores and displays the member's data value in the format *mm/dd/yyyy* or *dd/ mm/yyyy* ■ Text-Stores and displays the member's data value as text. ■ Unspecified-Stores and displays the member's data value as "unspecified." The default value is taken from the parent of the member being loaded or is Currency if the member is being added to the root dimension.
Exchange Rate Type	Takes the exchange rate. This column is dependent on the value specified for the Data Type column. Valid values: ■ Average, Ending, and Historical when Data Type is equal to Currency ■ None when Data Type is equal to Non-currency or Percentage This value is passed as a string. The default value is taken from the parent of the member that is being loaded or, if the member is being added to the root dimension, is based on the account type and takes the following values: ■ Revenue-Average ■ Expense-Average ■ Asset-Ending ■ Liability-Ending ■ Equity-Ending ■ Saved Assumption-None
Use 445	Indicates the distribution selected in the Planning application. If the application has no distribution, this column is not displayed. Valid values are 0 and 1 (or any number other than 0); default value: 1.

Table 20–2 (Cont.) Accounts

Column	Description
Variance Reporting	Takes a value for account members with an account type of Saved Assumption or if the record is rejected. Valid values: ■ Expense-designates the saved assumption as an expense. The actual amount is subtracted from the budgeted amount to determine the variance. ■ Non-Expense-designates the saved assumption as revenue. The budgeted amount is subtracted from the actual amount to determine the variance. This value is passed as a string. The default value is taken from the parent of the member being loaded or, if the member is being added to the root dimension, is based on the value of the count type. For Account types, the value is set to the following: ■ Revenue-Non-Expense ■ Expense-Expense ■ Asset-Non-Expense ■ Liability-Non-Expense ■ Equity-Non-Expense
Source Plan Type	Takes a plan type name for the plan type assigned to the member being loaded. Valid values are any plan types specified in Planning application. This value is passed as a string. The default is taken from the parent of the member being loaded. If the source plan of the parent is not valid for the member, the specified plan type is not selected for the member in the application, and the first plan type that the member is used in is used. If the member is being loaded to the root dimension, the first plan type the member is used in is used. When you update or save the parent of a member, the system verifies if the Source Plan Type associated with the member being loaded is valid for the new parent. If the member's source plan type is not a valid plan type of its parent member, you receive the error message, "The source plan type is not in the subset of valid plan types." If the source plan type of a member is valid for the parent member but not for the member itself, the member is saved but its source plan type is set to the first valid plan type (in the order Plan 1, Plan 2, Plan 3, Wrkforce, Capex). Note: If a Source Plan Type is specified in the adapter but is not valid for the parent, the record is rejected.
Plan Type *(Plan1)*	Boolean value that indicates if the member being loaded is used in Plan1. Valid values are 0 for False and any other number for True. The default value is True. The name of the column varies depending on the name of the plan type in the Planning application.

Table 20–2 (Cont.) Accounts

Column	Description
Aggregation *(Plan1)*	Takes the aggregation option for the member being loaded as related to Plan1. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Plan Type *(Plan 2)*	Boolean value that indicates if the member being loaded is used in Plan2. Valid values are 0 for False and any other number for True. The default value is True. The name of the column varies depending on the name of the plan type in the Planning application.
Aggregation *(Plan2)*	Takes the aggregation option for the member being loaded as related to Plan2. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Plan Type *(Plan3)*	Boolean value that indicates if the member being loaded is used in Plan3. Valid values: 0 for False or any other number for True; default value: True. The name of the column varies depending on the name of the plan type in the Planning application.
Aggregation *(Plan3)*	Takes the aggregation option for the member being loaded as related to Plan3. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never

Table 20–2 (Cont.) Accounts

Column	Description
Plan Type *(Wrkforce)*	For Workforce Planning: The Plan Type (Wrkforce) column is a Boolean value that indicates if the member being loaded is used in Workforce Planning. Valid values are 0 for False and any other number for True. The default is True. The actual name of the column varies, depending on by the name of the plan type in the Planning application.
Aggregation *(Wrkforce)*	For Workforce Planning: The Aggregation (Wrkforce) column takes the aggregation option for the member being loaded as related to Workforce Planning. This column is available only if the Planning application is valid for this plan type. The name of the column varies, depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Plan Type (Capex)	For Capital Expense Planning: The Plan Type (Capex) column is a Boolean value that indicates if the member being loaded is used in Capital Expense Planning. Valid values are 0 for False and any other number for True. The default is True. The actual name of the column varies, depending on by the name of the plan type in the Planning application.
Aggregation (Capex)	For Capital Expense Planning: Takes the aggregation option for the member being loaded as related to Capital Expense Planning. This column is available only if the Planning application is valid for this plan type. The name of the column varies, depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Custom Attribute	Takes the custom attribute member values. The name of the column is determined by the name of the custom attribute in the Planning application. The number of custom attribute columns varies depending on the number of attributes defined for the Account dimension. If you modify properties and do not specify a value, the custom attribute is not changed in the Planning application. If you specify <NONE> or <none> as the value, then the custom attribute in the Planning application is deleted. This value is passed as a string.
Member Formula	Takes the member formula values defined for the dimension member. By default, there is no member formula associated with a dimension or dimension member. You cannot load member formulas for dimension members that are Shared or Label Only.

Table 20–2 (Cont.) Accounts

Column	Description
UDA	Specifies a list of user-defined attributes to be updated. Note: You must define the UDA for the dimension members within Planning or by way of the UDA target.
Smart Lists	Takes the name of a user-defined Smart List defined in the Planning application. This value is passed as a string. The default for Smart Lists is <None>. Smart Lists are used in a metadata or dimension load (not a data load) allowing you to define the association of the Smart List name (not the values) with a given dimension member. You can have multiple Smart Listsassociatedwith a dimension but only one Smart Listassociated witha dimension member. These predefined Smart Lists are available in a Workforce Planning application: ■ None ■ Status ■ FT_PT ■ HealthPlan ■ TaxRegion ■ Month ■ Performance ■ Position ■ EmployeeType
Description	Takes a description for the member that is being loaded. By default, the Description column is empty. Note: If you do not enter a value for this column or do not connect the column, a new member is loaded without a description, and the description of an existing member is unchanged. If you enter <NONE> as the value for this column, any existing description for the member is deleted and is not loaded with the member.
Operation	Takes any of these values: ■ Update (default)-Adds, updates, or moves the member being loaded. ■ Delete Level 0-Deletes the member being loaded if it has no children. ■ Delete Idescendants-Deletes the member being loaded and all of its descendants. ■ Delete Descendants-Deletes the descendants of the member being loaded, but does not delete the member itself. Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.

20.6.2 Employee

Table 20–3 describes the columns of the Employee table. See Section 20.6.7, "Data Load Columns" for descriptions of additional columns that are displayed for loading Employee dimension data if the application has been set up for data load in Planning.

Table 20–3 Employee

Column	Description
Employee	Takes the name of the account member you are loading. If this member exists, its properties are modified; otherwise, the record is added. This field is required. The value for this field must meet these requirements: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string.
Parent	Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension. When you load data for a member and specify a different parent member that from the parent member in the application, the member is updated with the parent value that you specify. Example: If Member 1 has a parent value of Member A in your Planning application and you load Member 1 with a parent value of Member B, your application is updated, and Member B becomes the parent of Member 1. Member 1 and its descendants are moved from Member A to Member B. If the column is left blank, it is ignored during the load. The record is not loaded if one of the following situations occurs: ■ The specified parent is a descendant of the member that you are loading. ■ The specified parent does not exist in the Planning application.

Table 20–3 (Cont.) Employee

Column	Description
Default Alias	Takes an alternate name for the member being loaded. If you are modifying properties and do not specify a value, the alias is not changed in the Planning application. If you specify <NONE> or <none> as the value, the alias in the Planning application is deleted. The value for this column must meet the following requirements for a successful load: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string; default value: a null string.
Additional Alias	Can take an alternate name for the member being loaded. There will be as many Alias columns as there are Alias tables defined in Planning. The value for multiple alias columns must conform to the same requirements as those listed for the default alias column.
Data Storage	Takes the storage attribute for the member being loaded. Valid values: ■ Store ■ Dynamic Calc ■ Dynamic Calc and Store ■ Shared ■ Never Share (default) ■ Label Only This value is passed as a string.
Valid for Consolidation	The column is ignored.
Two Pass Calculation	Boolean value to indicate whether the member being loaded has the Two-Pass Calculation associated attribute. Valid values: 0 for False (default), or any other number for True. Values are valid only when the Data Storage value is Dynamic Calc or Dynamic Calc and Store; otherwise, the record is rejected.

Table 20–3 (Cont.) Employee

Column	Description
Data Type	Takes the data storage value. Valid values: ■ Currency-Stores and displays the member's data value in the default currency. ■ Non-currency-Stores and displays the member's data value as a numeric value. ■ Percentage-Stores data values as a numeric value and displays the member's data value as a percentage. ■ Smart list / enumeration-Stores data values as a numeric value and displays the member's data value as a string. ■ Date-Stores and displays the member's data value in the format *mm/dd/yyyy* or *dd/ mm/yyyy* ■ Text-Stores and displays the member's data value as text. ■ Unspecified-Stores and displays the member's data value as "unspecified." The default value is taken from the parent of the member being loaded or is Currency if the member is being added to the root dimension.
Custom Attribute	Takes the custom attribute member values. The name of the column is determined by the name of the custom attribute in the Planning application. The number of custom attribute columns varies depending on the number of attributes defined for the Employee dimension. If you modify properties and do not specify a value, the custom attribute is not changed in the Planning application. If you specify <NONE> or <none> as the value, then the custom attribute in the Planning application is deleted. This value is passed as a string.
Aggregation *(Plan1)*	Takes the aggregation option for the member being loaded as related to Plan1. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never

Table 20–3 (Cont.) Employee

Column	Description
Aggregation *(Plan2)*	Takes the aggregation option for the member being loaded as related to Plan2. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Aggregation *(Plan3)*	Takes the aggregation option for the member being loaded as related to Plan3. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Aggregation *(Wrkforce)*	For Workforce Planning: The Aggregation (Wrkforce) column takes the aggregation option for the member being loaded as related to Workforce Planning. This column is available only if the Planning application is valid for this plan type. The name of the column varies, depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never

Table 20–3 (Cont.) Employee

Column	Description
Aggregation (Capex)	For Capital Expense Planning: Takes the aggregation option for the member being loaded as related to Capital Expense Planning. This column is available only if the Planning application is valid for this plan type. The name of the column varies, depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Member Formula	Takes the member formula values defined for the dimension member. By default, there is no member formula associated with a dimension or dimension member. You cannot load member formulas for dimension members that are Shared or Label Only.
UDA	Specifies a list of user-defined attributes to be updated. Note: You must define the UDA for the dimension members within Planning or by way of the UDA target.
Smart Lists	Takes the name of a user-defined Smart List defined in the Planning application. This value is passed as a string. The default for Smart Lists is <None>. Smart Lists are used in a metadata or dimension load (not a data load) allowing you to define the association of the Smart List name (not the values) with a given dimension member. You can have multiple Smart Lists associatedwith a dimension but only one Smart List associated with a dimension member. These predefined Smart Lists are available in a Workforce Planning application: ■ None ■ Status ■ FT_PT ■ HealthPlan ■ TaxRegion ■ Month ■ Performance ■ Position ■ EmployeeType
Description	Takes a description for the member that is being loaded; empty by default. Note: If you do not enter a value for this column or do not connect the column, a new member is loaded without a description, and the description of an existing member is unchanged. If you enter <NONE> as the value for this column, any existing description for the member is deleted and is not loaded with the member.

Table 20–3 (Cont.) Employee

Column	Description
Operation	Takes any of these values: ■ Update (default)-Adds, updates, or moves the member being loaded. ■ Delete Level 0-Deletes the member being loaded if it has no children. ■ Delete Idescendants-Deletes the member being loaded and all of its descendants. ■ Delete Descendants-Deletes the descendants of the member being loaded, but does not delete the member itself. Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.

20.6.3 Entities

Table 20–4 describes the columns of the Entities table. See Section 20.6.7, "Data Load Columns" for descriptions of additional columns that are displayed for loading Entities data if the application has been set up for data load in Planning.

Table 20–4 Entities

Column	Description
Entity	Takes the name of the member you are loading. If this member exists, its properties are modified. If the member does not exist, then the record is added. This column is required. The value for this column must meet the following requirements for a successful load: The value for this field must meet these requirements: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string.

Table 20–4 (Cont.) Entities

Column	Description
Parent	Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension. When you update a member of an application using the Load method and specify a parent member that is different than the parent member in the application, the member is updated with the new parent value specified in your flow diagram. For example, if Member 1 has a parent value of Member A in your Planning application and you load Member 1 with a parent value of Member B, the system updates your application and makes Member B the parent of Member 1. Member 1 and its descendants are moved from Member A to Member B. If the column is left blank, it is ignored during the load. The record is not loaded if one of the following situations occurs: ■ The specified parent is a descendant of the member that you are loading. ■ The specified parent does not exist in the Planning application.
Default Alias	Takes an alternate name for the member being loaded. If you are modifying properties and do not specify a value, the alias is not changed in the Planning application. If you specify <NONE> or <none> as the value, the alias in the Planning application is deleted. The value for this column must meet the following requirements for a successful load: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string; default value: a null string.
Additional Alias	Additional Alias columns can take alternate names for the member being loaded. There are as many Alias columns as there are Alias tables defined in Planning. The value for multiple alias columns must conform to the same requirements as those listed for the default alias column.

Table 20–4 (Cont.) Entities

Column	Description
Data Storage	Takes the storage attribute for the member being loaded. Valid values: ■ Store ■ Dynamic Calc ■ Dynamic Calc and Store ■ Shared ■ Never Share (default) ■ Label Only This value is passed as a string.
Two Pass Calculation	Boolean value to indicate if the member being loaded has the Two-Pass Calculation attribute associated in the Planningapplication. Valid values: 0 for False (default), or any other number for True. Values are valid only when the Data Storage value is Dynamic Calc or Dynamic Calc and Store; otherwise, the record is rejected.
Data Type	Takes the data storage value. Valid values: ■ Currency-Stores and displays the member's data value in the default currency. ■ Non-currency-Stores and displays the member's data value as a numeric value. ■ Percentage-Stores data values as a numeric value and displays the member's data value as a percentage. ■ Smart list / enumeration-Stores data values as a numeric value and displays the member's data value as a string. ■ Date-Stores and displays the member's data value in the format *mm/dd/yyyy* or *dd/ mm/yyyy* ■ Text-Stores and displays the member's data value as text. ■ Unspecified-Stores and displays the member's data value as "unspecified." The default value is taken from the parent of the member being loaded or is Currency if the member is being added to the root dimension.
Base Currency	Takes the base currency for the entity being loaded. It takes the code of the currency as defined in your Planning application. The default value is USD. This column is displayed only when the application is defined to be multi-currency.
Plan Type *(Plan1)*	Boolean value that indicates if the member being loaded is used in Plan1. Valid values: 0 for False or any other number for True (default). The name of the column varies depending on the name of the plan type in the Planning application.

Table 20–4 (Cont.) Entities

Column	Description
Aggregation *(Plan1)*	Takes the aggregation option for the member being loaded as related to Plan1. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Plan Type *(Plan2)*	Boolean value that indicates if the member being loaded is used in Plan2. Valid values are 0 for False and any other number for True. The default value is True. The name of the column varies depending on the name of the plan type in the Planning application.
Aggregation *(Plan2)*	Takes the aggregation option for the member being loaded as related to Plan2. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Plan Type *(Plan 3)*	Boolean value that indicates if the member being loaded is used in Plan3. Valid values: 0 for False or any other number for True; default value: True. The name of the column varies depending on the name of the plan type in the Planning application.
Aggregation *(Plan3)*	Takes the aggregation option for the member being loaded as related to Plan3. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never

Table 20–4 (Cont.) Entities

Column	Description
Aggregation *(Wrkforce)*	For Workforce Planning: The Aggregation (Wrkforce) column takes the aggregation option for the member being loaded as related to Workforce Planning. This column is available only if the Planning application is valid for this plan type. The name of the column varies, depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Aggregation (Capex)	For Capital Expense Planning: Takes the aggregation option for the member being loaded as related to Capital Expense Planning. This column is available only if the Planning application is valid for this plan type. The name of the column varies, depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Custom Attribute	Takes the custom attribute member values. The name of the column is determined by the name of the custom attribute in the Planning application. The number of custom attribute columns varies depending on the number of attributes defined for the Entity dimension. If you modify properties and do not specify a value, the custom attribute is not changed in the Planning application. If you specify <NONE> or <none> as the value, then the custom attribute in the Planning application is deleted. This value is passed as a string.
Member Formula	Takes the member formula values defined for the dimension member. By default, there is no member formula associated with a dimension or dimension member. You cannot load member formulas for dimension members that are Shared or Label Only.
UDA	Specifies a list of user-defined attributes to be updated. Note: You must define the UDA for the dimension members within Planning or by way of the UDA target.

Table 20–4 (Cont.) Entities

Column	Description
Smart Lists	Takes the name of a user-defined Smart List defined in the Planning application. This value is passed as a string. The default for Smart Lists is <None>. Smart Lists are used in a metadata or dimension load (not a data load) allowing you to define the association of the Smart List name (not the values) with a given dimension member. You can have multiple Smart Lists associatedwith a dimension but only one Smart List associated with a dimension member. These predefined Smart Lists are available in a Workforce Planning application: ■ None ■ Status ■ FT_PT ■ HealthPlan ■ TaxRegion ■ Month ■ Performance ■ Position ■ EmployeeType
Description	Takes a description for the member that is being loaded; empty by default. Note: If you do not enter a value for this column or do not connect the column, a new member is loaded without a description, and the description of an existing member is unchanged. If you enter <NONE> as the value for this column, any existing description for the member is deleted and is not loaded with the member.
Operation	Takes any of these values: ■ Update (default)-Adds, updates, or moves the member being loaded. ■ Delete Level 0-Deletes the member being loaded if it has no children. ■ Delete Idescendants-Deletes the member being loaded and all of its descendants. ■ Delete Descendants-Deletes the descendants of the member being loaded, but does not delete the member itself. Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.

20.6.4 User-Defined Dimensions

Table 20–5 describes the columns of the User-Defined Dimensions table.

Table 20–5 User-Defined Dimensions

Column	Description
Entity	Takes the name of the member you are loading. If this member exists, its properties are modified. If the member does not exist, then the record is added. This column is required. The value for this column must meet the following requirements for a successful load: The value for this field must meet these requirements: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string.
Parent	Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension. When you update a member of an application using the Load method and specify a parent member that is different than the parent member in the application, the member is updated with the new parent value specified in your flow diagram. For example, if Member 1 has a parent value of Member A in your Planning application and you load Member 1 with a parent value of Member B, the system updates your application and makes Member B the parent of Member 1. Member 1 and its descendants are moved from Member A to Member B. If the column is left blank, it is ignored during the load. The record is not loaded if one of the following situations occurs: ■ The specified parent is a descendant of the member that you are loading. ■ The specified parent does not exist in the Planning application.

Table 20–5 (Cont.) User-Defined Dimensions

Column	Description
Default Alias	Takes an alternate name for the member being loaded. If you are modifying properties and do not specify a value, the alias is not changed in the Planning application. If you specify <NONE> or <none> as the value, the alias in the Planning application is deleted. The value for this column must meet the following requirements for a successful load: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string; default value: a null string.
Additional Alias	Additional Alias columns can take alternate names for the member being loaded. There are as many Alias columns as there are Alias tables defined in Planning. The value for multiple alias columns must conform to the same requirements as those listed for the default alias column.
Data Storage	Takes the storage attribute for the member being loaded. Valid values: ■ Store ■ Dynamic Calc ■ Dynamic Calc and Store ■ Shared ■ Never Share (default) ■ Label Only This value is passed as a string.
Two Pass Calculation	Boolean value to indicate if the member being loaded has the Two-Pass Calculation attribute associated in the Planningapplication. Valid values: 0 for False (default), or any other number for True. Values are valid only when the Data Storage value is Dynamic Calc or Dynamic Calc and Store; otherwise, the record is rejected.

Table 20–5 (Cont.) User-Defined Dimensions

Column	Description
Data Type	Takes the data storage value. Valid values: ■ Currency-Stores and displays the member's data value in the default currency. ■ Non-currency-Stores and displays the member's data value as a numeric value. ■ Percentage-Stores data values as a numeric value and displays the member's data value as a percentage. ■ Smart list / enumeration-Stores data values as a numeric value and displays the member's data value as a string. ■ Date-Stores and displays the member's data value in the format *mm/dd/yyyy* or *dd/ mm/yyyy* ■ Text-Stores and displays the member's data value as text. ■ Unspecified-Stores and displays the member's data value as "unspecified." The default value is taken from the parent of the member being loaded or is Currency if the member is being added to the root dimension.
Aggregation *(Plan1)*	Takes the aggregation option for the member being loaded as related to Plan1. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Aggregation *(Plan2)*	Takes the aggregation option for the member being loaded as related to Plan2. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never

Table 20–5 (Cont.) User-Defined Dimensions

Column	Description
Aggregation *(Plan3)*	Takes the aggregation option for the member being loaded as related to Plan3. This column is available only ifthe Planning application is valid for this plan type. The name of the column varies depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Aggregation *(Wrkforce)*	For Workforce Planning: The Aggregation (Wrkforce) column takes the aggregation option for the member being loaded as related to Workforce Planning. This column is available only if the Planning application is valid for this plan type. The name of the column varies, depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never
Aggregation (Capex)	For Capital Expense Planning: Takes the aggregation option for the member being loaded as related to Capital Expense Planning. This column is available only if the Planning application is valid for this plan type. The name of the column varies, depending on the name of the plan type in the Planning application. This value is passed as a string. Valid values: ■ + (default) ■ - ■ * ■ / ■ % ■ ~ ■ Never

Table 20–5 (Cont.) User-Defined Dimensions

Column	Description
Custom Attribute	Takes the custom attribute member values. The name of the column is determined by the name of the custom attribute in the Planning application. The number of custom attribute columns varies depending on the number of attributes defined for the Entity dimension. If you modify properties and do not specify a value, the custom attribute is not changed in the Planning application. If you specify <NONE> or <none> as the value, then the custom attribute in the Planning application is deleted. This value is passed as a string.
Member Formula	Takes the member formula values defined for the dimension member. By default, there is no member formula associated with a dimension or dimension member. You cannot load member formulas for dimension members that are Shared or Label Only.
UDA	Specifies a list of user-defined attributes to be updated. Note: You must define the UDA for the dimension members within Planning or by way of the UDA target.
Smart Lists	Takes the name of a user-defined Smart List defined in the Planning application. This value is passed as a string. The default for Smart Lists is <None>. Smart Lists are used in a metadata or dimension load (not a data load) allowing you to define the association of the Smart List name (not the values) with a given dimension member. You can have multiple Smart Lists associatedwith a dimension but only one Smart List associated with a dimension member. These predefined Smart Lists are available in a Workforce Planning application: ■ None ■ Status ■ FT_PT ■ HealthPlan ■ TaxRegion ■ Month ■ Performance ■ Position ■ EmployeeType
Description	Takes a description for the member that is being loaded; empty by default. Note: If you do not enter a value for this column or do not connect the column, a new member is loaded without a description, and the description of an existing member is unchanged. If you enter <NONE> as the value for this column, any existing description for the member is deleted and is not loaded with the member.

Table 20–5 (Cont.) User-Defined Dimensions

Column	Description
Operation	Takes any of these values: ■ Update (default)-Adds, updates, or moves the member being loaded. ■ Delete Level 0-Deletes the member being loaded if it has no children. ■ Delete Idescendants-Deletes the member being loaded and all of its descendants. ■ Delete Descendants-Deletes the descendants of the member being loaded, but does not delete the member itself. Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.

20.6.5 Attribute Dimensions

Table 20–6 describes the columns of the Attribute Dimensions table.

Note: The Parent, Default Alias, and Additional Alias columns are available only in Planning 9.3.1 and later.

Table 20–6 Attribute Dimensions

Column	Description
Entity	Takes the name of the member you are loading. If this member exists, its properties are modified. If the member does not exist, then the record is added. This column is required. The value for this column must meet the following requirements for a successful load: The value for this field must meet these requirements: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string.

Table 20–6 (Cont.) Attribute Dimensions

Column	Description
Parent	Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension. When you update a member of an application using the Load method and specify a parent member that is different than the parent member in the application, the member is updated with the new parent value specified in your flow diagram. For example, if Member 1 has a parent value of Member A in your Planning application and you load Member 1 with a parent value of Member B, the system updates your application and makes Member B the parent of Member 1. Member 1 and its descendants are moved from Member A to Member B. If the column is left blank, it is ignored during the load. The record is not loaded if one of the following situations occurs: ■ The specified parent is a descendant of the member that you are loading. ■ The specified parent does not exist in the Planning application.
Default Alias	Takes an alternate name for the member being loaded. If you are modifying properties and do not specify a value, the alias is not changed in the Planning application. If you specify <NONE> or <none> as the value, the alias in the Planning application is deleted. The value for this column must meet the following requirements for a successful load: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string; default value: a null string.
Additional Alias	Additional Alias columns can take alternate names for the member being loaded. There are as many Alias columns as there are Alias tables defined in Planning. The value for multiple alias columns must conform to the same requirements as those listed for the default alias column.

Table 20–6 (Cont.) Attribute Dimensions

Column	Description
Operation	Takes any of these values: ■ Update (default)-Adds, updates, or moves the member being loaded. ■ Delete Level 0-Deletes the member being loaded if it has no children. ■ Delete Idescendants-Deletes the member being loaded and all of its descendants. ■ Delete Descendants-Deletes the descendants of the member being loaded, but does not delete the member itself. Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.

20.6.6 UDA

Table 20–7 describes the columns of the UDA table.

Table 20–7 UDA

Column	Description
Dimension	Takes the dimension name for the UDA. You can associate UDAs only with dimensions that exist in the Planning application. If the UDA exists, its properties are modified; otherwise, the record is added. This column is required.
UDA	Takes the values of the UDA that you are loading.
Dimension	Takes the values of the UDA you are loading. The value for this column must meet the following requirements for a successful load: The value for this column must meet the following requirements for a successful load: ■ Unique ■ Alphanumeric ■ Not more than 80 characters ■ Member name cannot contain tabs, double quotation marks ("), or backslash (\) characters. ■ Member name cannot start with any of these characters: ' \ < | , = @ _ + - { } ( ) . ■ Value must not be an Essbase reserved word such as Children, Parent, $$$UNIVERSE $$$, #MISSING, or #MI. For more information about reserved words in Essbase, see the *Hyperion Essbase - System 9 Database Administrator's Guide* or Essbase online help. This value is passed as a string; default value: a null string.

Table 20–7 (Cont.) UDA

Column	Description
Operation	Takes any of these values: ■ Update (default)-Adds, updates, or moves the member being loaded. ■ Delete Level 0-Deletes the member being loaded if it has no children. ■ Delete Idescendants-Deletes the member being loaded and all of its descendants. ■ Delete Descendants-Deletes the descendants of the member being loaded, but does not delete the member itself. Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.

20.6.7 Data Load Columns

These columns for loading data into Account, Employee, Entities, and user-defined dimensions are displayed if the application has been set up for data load in Planning.

Table 20–8 Data Load Columns

Columns	Description
Data Load Cube Name	Takes the name of the plan type to which data is being loaded. The value is passed as a string. Valid values are any plan types specified in the Planning application. For example: ■ Plan1 ■ Plan2 ■ Plan3 ■ Wkforce ■ Capex
Driver Member	Takes the name of the driver member that is selected when the Planning, Oracle's Hyperion® Workforce Planning, or Oracle's Hyperion® Capital Expense Planning application is set up for loading data. You can have one driver dimension per load. The Driver Dimension and Driver Dimension Members are defined in the Data Load Administration page in Planning. The driver members are the members into which the data is loaded. The number of driver member columns depends on the number of driver members you select in Oracle's Hyperion® Planning - System 9. The value is passed as a string representing a numeric value or, if a Smart List is bound to the member represented on this column, a Smart List value. Note: The Smart List field on this load method does not affect this column.
Point-of-View	Takes the names of all the other dimensions that are required to determine the intersection to load the data. The value is passed as a string. The data load automatically performs cross-product record creations based on dimension parameters defined in the POV. For example, an employee's Smart List attribute values that are constant over time such as full time status for all twelve months need only be supplied once in the data feed and the load file will create and load that data record for each relevant cell intersection.

Column	Description
Data Load Cube Name	Takes the name of the plan type to which data is being loaded. The value is passed as a string. Valid values are any plan types specified in the Planning application. For example: ■ Plan1 ■ Plan2 ■ Plan3 ■ Wkforce ■ Capex
Driver Member	Takes the name of the driver member that is selected when the Planning, Oracle's Hyperion® Workforce Planning, or Oracle's Hyperion® Capital Expense Planning application is set up for loading data. You can have one driver dimension per load. The Driver Dimension and Driver Dimension Members are defined in the Data Load Administration page in Planning. The driver members are the members into which the data is loaded. The number of driver member columns depends on the number of driver members you select in Oracle's Hyperion® Planning - System 9. The value is passed as a string representing a numeric value or, if a Smart List is bound to the member represented on this column, a Smart List value. Note: The Smart List field on this load method does not affect this column.
Point-of-View	Takes the names of all the other dimensions that are required to determine the intersection to load the data. The value is passed as a string. The data load automatically performs cross-product record creations based on dimension parameters defined in the POV. For example, an employee's Smart List attribute values that are constant over time such as full time status for all twelve months need only be supplied once in the data feed and the load file will create and load that data record for each relevant cell intersection.

21

Oracle OLAP

This chapter describes how to work with Oracle OLAP in Oracle Data Integrator.

This chapter includes the following sections:

21.1 Introduction

Oracle Data Integrator (ODI) seamlessly integrates data in an Oracle OLAP. All Oracle Data Integrator features are designed to work best with the Oracle OLAP cubes, including reverse-engineering and integration interfaces.

Oracle Data Integrator uses Java Database Connectivity (JDBC) to connect to the Oracle database instance containing the Oracle OLAP cubes.

21.1.1 Concepts

The Oracle Data Integrator Knowledge Modules for Oracle OLAP provide integration and connectivity between Oracle Data Integrator and Oracle OLAP cubes. Oracle Data Integrator is able to handle two different types of cubes with the Oracle OLAP KMs, depending on the storage mode of these cubes:

- ROLAP (Relational OnLine Analytical Processing) cubes are based on a relational storage model. ROLAP cubes can handle a large amount of data and benefit all features of the relational database.
- MOLAP (Multidimensional OnLine Analytical Processing) data is stored in form of multidimensional cubes. The MOLAP model provides high query performance and fast data retrieval for a limited amount of data.

The Oracle Data Integrator KMs for Oracle OLAP use mature integration methods for Oracle OLAP in order to:

- Reverse-Engineer Oracle OLAP data structures (all tables used by a ROLAP or a MOLAP cube).
- Integrate data in an Oracle Analytical Workspace target in incremental update mode.

Note: The Oracle Data Integrator Oracle OLAP KMs are similar to the standard Oracle Database KMs. This chapter describes the Oracle OLAP specificities. See Chapter 2, "Oracle Database" for a description of the Oracle Database KMs.

21.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 21–1 for handling Oracle OLAP data. The KMs use Oracle OLAP specific features. It is also possible to use the generic SQL KMs and Oracle Database KMs with the Oracle OLAP. See Chapter 4, "Generic SQL" and Chapter 2, "Oracle Database" for more information.

Table 21–1 Oracle OLAP Knowledge Modules

Knowledge Module	Description
RKM Oracle OLAP (Jython)	Reverse-engineering knowledge module to retrieve the tables, views, columns, Primary Keys, Unique Keys and Foreign keys from Oracle Database, which are used by a ROLAP or a MOLAP Cube. This KM provides logging (Use Log & Log File Name) options.
IKM Oracle AW Incremental Update	This KM is similar to the IKM Oracle Incremental Update. It has additional options for handling MOLAP cubes.

21.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Oracle OLAP Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

21.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

21.2.2 Technology Specific Requirements

There are no connectivity requirements for using Oracle OLAP data in Oracle Data Integrator. The requirements for the Oracle Database apply also to Oracle OLAP. See Chapter 2, "Oracle Database" for more information.

The RKM Oracle OLAP (Jython) uses in addition Oracle OLAP libraries. Copy the awxml.jar and olap_api.jar from the `ORACLE_HOME/olap/api/lib` folder into the additional drivers folder for ODI.

21.2.3 Connectivity Requirements

There are no connectivity requirements for using Oracle OLAP data in Oracle Data Integrator. The requirements for the Oracle Database apply also to Oracle OLAP. See Chapter 2, "Oracle Database" for more information.

21.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating an Oracle Data Server
2. Creating an Oracle Physical Schema

21.3.1 Creating an Oracle Data Server

This step consists in declaring in Oracle Data Integrator the data server, as well as the physical and logical schemas that store the Oracle OLAP cubes.

21.3.1.1 Creation of the Data Server

Create a data server for the Oracle technology as described in Section 2.3.1, "Creating an Oracle Data Server".

21.3.2 Creating an Oracle Physical Schema

Create an Oracle physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

21.4 Setting Up an Integration Project

Setting up a project using the Oracle OLAP features follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with Oracle OLAP:

- IKM Oracle AW Incremental Update
- RKM Oracle OLAP (Jython)

Import also the Oracle Database knowledge modules recommended in Chapter 2, "Oracle Database".

21.5 Creating and Reverse-Engineering an Oracle Model

This section contains the following topics:

- Create an Oracle Model
- Reverse-engineer an Oracle OLAP Cube

21.5.1 Create an Oracle Model

Create an Oracle Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

21.5.2 Reverse-engineer an Oracle OLAP Cube

Oracle OLAP supports Customized reverse-engineering. The RKM Oracle OLAP (Jython) retrieves the metadata from the Oracle tables used by an Oracle OLAP cube.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on Oracle OLAP, use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields specific to the Oracle technology:

1. In the Reverse tab of the Oracle Model, select the `KM: RKM Oracle OLAP (Jython).<project name>`.
2. Set the RKM options as follows:
   - MOLAP: Set to `YES` to reverse an Analytic Workspace. If this option is set to `YES`, the following options are mandatory:
     - AW_NAME: Indicate the name of the Analytical Workspace.
     - AW_URL: Specify the URL of the Analytical Workspace.
     - AW_OWNER: Indicate the name of the Analytical Workspace Owner.
     - AW_PASSWORD: Indicate the password of the Analytical Workspace Owner.
   - ROLAP: Set to `YES` to reverse tables from a ROLAP schema.
   - USE_LOG: Set to `YES` to write the log details of the reverse-engineering process into a log file.
   - LOG_FILE_NAME: Specify the name of the log file.

   The reverse-engineering process returns the tables used by a cube as datastores. You can then use these datastores as a source or a target of your interfaces.

21.6 Working with Oracle OLAP KMs in Integration Interfaces

You can use the Oracle Data Integrator Oracle OLAP KMs as well as the standard Oracle Database KMs. The Oracle OLAP KM specific steps are detailed in the following sections.

21.6.1 Using Oracle OLAP as a Source in an Integration Interface

After performing a reverse-engineering using the RKM Oracle OLAP (Jython), you can use Oracle OLAP data tables as a source of an integration interface to extract data from the Oracle OLAP database and integrate them into another system (Data warehouse, other database...). Using Oracle OLAP as a source in these conditions is identical to using an Oracle datastore as a source in an integration interface. The Generic SQL and Oracle Database KMs can be used for this purpose.

See the following chapters for more information:

- Chapter 2, "Oracle Database"

- Chapter 4, "Generic SQL"

21.6.2 Using Oracle ROLAP as a Target in an Integration Interface

After performing a reverse-engineering using the RKM Oracle OLAP (Jython), you can use Oracle ROLAP data tables as a target of an integration interface to load data from any system to the Oracle ROLAP database. Using Oracle ROLAP as a target in these conditions is identical to using an Oracle datastore as a target in an integration interface. The Generic SQL and Oracle Database KMs can be used for this purpose.

See the following chapters for more information:

- Chapter 2, "Oracle Database"
- Chapter 4, "Generic SQL"

21.6.3 Using Oracle MOLAP as a Target in an Integration Interface

Using Oracle MOLAP as a Target in an integration interface is similar to using Oracle ROLAP as a target with the difference that, in addition to the standard features of the integration process, you can refresh the MOLAP cube at the execution of the integration interface by using the IKM Oracle AW Incremental Update.

This IKM is similar to the IKM Oracle Incremental Update. See Chapter 2, "Oracle Database"for more information. It has four additional options for handling MOLAP cubes:

- AW_NAME: The name of the Analytical Workspace.
- AW_OWNER: The name of the Analytical Workspace owner.
- CUBE_NAME: The name of the cube.
- REFRESH_CUBE: Set this option to `YES` to refresh the cube for an Analytical Workspace.

In order to avoid refreshing the cube at every integration interface step, use the IKM Oracle AW Incremental Update with the refresh cube options only in the last integration interface of the package.

In the last integration interface set the options to refresh the cube as follows:

- Set the REFRESH_CUBE option to `YES`.
- Specify the values for the AW_OWNER, AW_NAME, and CUBE_NAME options.

Part III

Other Technologies

This part describes how to work with other technologies in Oracle Data Integrator.

Part III contains the following chapters:

22

JMS

This chapter describes how to work with Java Message Services (JMS) in Oracle Data Integrator.

This chapter includes the following sections:

22.1 Introduction

Oracle Data Integrator provides a simple and transparent method to integrate JMS destinations. This chapter focuses on processing JMS messages with a text payload in batch mode. For XML payload processing, refer to Chapter 23, "JMS XML".

22.1.1 Concepts

The JMS Knowledge Modules apply to most popular JMS compliant middleware, including Oracle Service Bus, Sonic MQ, IBM Websphere MQ, and so forth. Most of these Knowledge Modules include transaction handling to ensure message delivery.

22.1.1.1 JMS Message Structure

This section describes the structure of a message in a JMS destination.

A JMS Message consists of three sections:

- Header
- Properties
- Payload

Header

The header contains in the header fields standard metadata concerning the message, including the destination (JMSDestination), Message ID (JMSMessageID), Message Type (JMSType), and so forth.

Properties

The properties section contains additional metadata concerning the message. These metadata are properties, that can be separated in three groups:

- JMS-Defined properties which are optional JMS Headers. Their name begins with JMSX(JMSXUserID, JMSXAppID, etc.).
- Provider-specific properties. They are specific to the router vendor. Their names start with JMS_<vendor name>.
- Application-specific properties. These properties depend on the application sending the messages. These are user-defined information that is not included in the message payload.

The Header and Properties sections provide a set of header fields and properties that:

- Have a specific Java data type (Boolean, string, short, and so forth),
- Can be accessed for reading and/or writing,
- Can be used for filtering on the router through the JMS Selector.

Payload

The payload section contains the message content. This content can be anything (text, XML, binary, and so forth).

22.1.1.2 Using a JMS Destination

Oracle Data Integrator is able to process JMS Text and Byte messages that are delivered by a JMS destination. Each message is considered as a container for rows of data and is handled through the JMS Queue or JMS Topic technology.

With JMS Queue/JMS Topic technologies, each JMS destination is defined similarly to a flat file datastore. Each message in the destination is a record in the datastore.

In the topology, each JMS router is defined as a JMS Topic/Queue data server, with a single physical schema. A JMS router may be defined therefore twice to access its topics using one data server, and its queues using another one.

Each JMS destination (Topic of Queue) is defined as a JMS datastore which resource name matches the name of the JMS destination (name of the queue or topic as defined in the router). A model groups message structures related to different topics or queues.

The JMS datastore structure is defined similarly to a flat file (delimited or fixed width). The properties or header fields of the message can be declared with JMS-specific data types as additional pseudo-columns in this flat file structure. Each message in the destination is processed as a record of a JMS datastore.

Processing Messages

JMS destinations are handled as regular file datastores and messages as rows from these datastores. With these technologies, entire message sets are produced and consumed within each interface.

Message publishing as well consumption requires a *commit* action to finalize removing/posting the message from/to the JMS destination. Committing is particularly important when reading. Without a commit, the message is read but not consumed. It remains in the JMS Topic/Queue and will be re-read at a later time.

Both the message content and pseudo-columns can be used as regular columns in the integration interfaces (for mapping, filter, etc.). Certain pseudo-columns (such as the

one representing the MESSAGE_ID property) are read-only, and some properties of header fields are used (or set) through the Knowledge Module options.

Using Data Integrator you can transfer information either through the message payload - the columns - , or through the properties - pseudo-columns - (application properties, for example).

Using the properties to carry information is restricted by third-party applications producing or consuming the messages.

Filtering Messages

It is possible to filter messages from a JMS destination in two ways:

- By defining a *filter* using the datastore's columns and pseudo-columns. In this case Data Integrator performs the filtering operation after consuming the messages. This implies that messages rejected by this filter may also be consumed.
- By defining a *Message Selector* (MESSAGE_SELECTOR KM option). This type of filter can only use the properties or header fields of the message. The filter is processed by the router, and only the messages respecting the filter are consumed, reducing the number of messages transferred.

22.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 22–1 for handling JMS messages.

Table 22–1 JMS Knowledge Modules

Knowledge Module	Description
IKM SQL to JMS Append	Integrates data into a JMS compliant message queue or topic in text or binary format from any SQL compliant staging area. Consider using this IKM if you plan to transform and export data to a target JMS queue or topic. If most of your source datastores are located on the same data server, we recommend using this data server as staging area to avoid extra loading phases (LKMs). To use this IKM, the staging area must be different from the target.
LKM JMS to SQL	Loads data from a text or binary JMS compliant message queue or topic to any SQL compliant database used as a staging area. This LKM uses the Agent to read selected messages from the source queue/topic and write the result in the staging temporary table created dynamically. To ensure message delivery, the message consumer (or subscriber) does not commit the read until the data is actually integrated into the target by the IKM. Consider using this LKM if one of your source datastores is a text or binary JMS message.

22.2 Installation and Configuration

Make sure you have read the information in this section before you start using the JMS Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

22.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

http://www.oracle.com/technology/products/oracle-data-integrator/index.html.

22.2.2 Technology Specific Requirements

The JMS destinations are usually accessed via a JNDI service. The configuration and specific requirements for JNDI and JMS depends on the JMS Provider you are connecting to. Refer to the JMS Provider specific documentation for more details.

22.2.3 Connectivity Requirements

Oracle Data Integrator does not include specific drivers for JMS providers. Refer to the JMS Provider documentation for the connectivity requirement of this provider.

22.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating a JMS Data Server
2. Creating a JMS Physical Schema

22.3.1 Creating a JMS Data Server

A JMS data server corresponds to one JMS provider/router that is accessible through your local network.

It exists two types of JMS data servers: JMS Queue and JMS Topic.

- A *JMS Queue data server* is used to access several queues in the JMS router.
- A *JMS Topic data server* is used to access several topics in the JMS router

22.3.1.1 Creation of the Data Server

Create a data server either for the JMS Queue technology or for the JMS Topic technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a JMS Queue or JMS Topic data server.

1. In the Definition tab:
   - Name: Name of the data server as it will appear in Oracle Data Integrator.
   - User/Password: Not used here. Leave these fields empty.
2. In the JNDI tab:
   - JNDI Authentication: Set this field to `None`.
   - JNDI User: Enter the username to connect to the JNDI directory (optional step).

- Password: This user's password (optional step).
- JNDI Protocol: From the list, select the JNDI protocol (optional step).
- JNDI Driver: Name of the initial context factory java class to connect to the JNDI provider, for example: `com.sun.jndi.ldap.LdapCtxFactory` for LDAP
- JNDI URL: `<JMS_RESOURCE>`, for example `ldap://<host>:<port>/<dn>` for LDAP
- JNDI Resource: Logical name of the JNDI resource corresponding to your JMS Queue or Topic connection factory.

  For example, specify `QueueConnectionFactory` if you want to access a message queue and `TopicConnectionFactory` if you want to access a topic. Note that these parameters are specific to the JNDI directory and the provider.

22.3.2 Creating a JMS Physical Schema

Create a JMS physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Note: Only one physical schema is required per JMS data server.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

22.4 Setting Up an Integration Project

Setting up a project using JMS follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with JMS:

- IKM SQL to JMS Append
- LKM JMS to SQL

22.5 Creating and Defining a JMS Model

This section contains the following topics:

Note: It is not possible to reverse-engineer a JMS model. To create a datastore you have to create a JMS model and define the JMS datastores.

22.5.1 Create a JMS Model

Create a JMS Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

A JMS Model is a set of datastores corresponding to the Topics or Queues of a router. Each datastore corresponds to a specific Queue or Topic. The datastore structure defines the message structure for this queue or topic. A model is always based on a Logical Schema. In a given Context, the Logical Schema corresponds to one JMS Physical Schema. The Data Schema corresponding to this Physical Schema contains the Topics or Queues.

22.5.2 Defining the JMS Datastores

In Oracle Data Integrator, each datastore is a JMS Topic or Queue. Each message in this topic or queue is a row of the datastore.

A JMS message may carry any type of information and there is no metadata retrieval method available. Therefore reverse-engineering is not possible.

To define the datastore structure, do one of the following:

- Create the datastore as a file datastore and manually declare the message structures.
- Use the File reverse-engineering through an Excel spreadsheet in order to automate the reverse engineering of messages. See Chapter 3, "Files" for more information about this reverse-engineering method.
- Duplicate a datastore from another model into the JMS model.

Important: The datastores' resource names must be identical to the name of JMS destinations (this is the logical JNDI name) that will carry the message corresponding to their data. Note that these names are frequently case-sensitive.

Declaring JMS Properties as Pseudo-Columns

The property pseudo-columns represent properties or header fields of a message. These pseudo-columns are defined in the Oracle Data Integrator model as columns in the JMS datastore, with JMS-specific datatypes. The JMS-specific datatypes are called JMS_xxx (for example: JMS String, JMS Long, JMS Int, and so forth).

To define these property pseudo-columns, simply declare additional columns named identically to the properties and specified with the appropriate JMS-specific datatypes.

If you define pseudo-columns that are named like standard, provider-specific or application-specific properties, they will be consumed or published with the message as such. If a pseudo-column is not listed in the standard or provider-specific set of JMS properties, It is considered as additional application-specific property.

For example, to use or set in interfaces the JMSPriority default property on messages consumed from or pushed to a JMS queue called CUSTOMER, you would add a column called *JMSPriority* (with this exact case) to the CUSTOMER datastore. This column would have the *JMS Int* datatype available for the JMS Queue technology.

Warning:

- Property pseudo-columns must be defined and positioned in the JMS datastore after the columns making up the message payload. Use the Order field in the column definition to position these columns. The order of the pseudo-columns themselves is not important as long as they appear at the end of the datastore definition.
- Pseudo-columns names are case-sensitive.

For more information about JMS Properties, see:

- Section 22.7, "JMS Standard Properties"
- Section 22.7.1, "Using JMS Properties"

22.6 Designing an Interface

You can use JMS as a source or a target of an integration interface. It cannot be used as the staging area.

The KM choice for an interface or a check determines the abilities and performance of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning JMS messages.

22.6.1 Loading Data from a JMS Source

JMS can be used as a source or a target in an interface. Data from a JMS message Queue or Topic can be loaded to any SQL compliant database used as a staging area. The LKM choice in the Interface Flow tab to load data between JMS and another type of data server is essential for the performance of an interface.

Oracle Data Integrator provides the LKM JMS to SQL for loading data from a JMS source to a Staging Area. This LKM loads data from a text or binary JMS compliant message queue or topic to any SQL compliant database used as a staging area.

Table 22–2 lists the JMS specific options.

22.6.2 Integrating Data in a JMS Target

Oracle Data Integrator provides the IKM SQL to JMS Append that implements optimized data integration strategies for JMS. This IKM integrates data into a JMS compliant message queue or topic in text or binary format from any SQL compliant staging area. Table 22–2 lists the JMS specific KM options of this IKM.

The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

JMS Knowledge Modules Options

Table 22–2 lists the JMS specific KM options of the JMS IKM and LKM.

The JMS specific options of this LKM are similar to the options of the IKM SQL to JMS Append. There are only two differences:

- The DELETE_TEMPORARY_OBJECTS option is only provided for the LKM.
- The PUBLISH option is only provided for the IKM.

Table 22–2 JMS Specific KM Options

Option	Used to	Description
PUBLISH	Write	Check this option if you want to publish new messages in the destination. This option is set to Yes by default.
JMS_COMMIT	Read/Write	Commit the publication or consumption of a message. Uncheck this option if you don't want to commit your publication/consumption on your router. This option is set to Yes by default.
JMSDELIVERYMODE	Write	JMS delivery mode (1: Non Persistent, 2: Persistent). A persistent message remains on the server and is recovered on server crash.
JMSEXPIRATION	Write	Expiration delay in milliseconds for the message on the server [0..4 000 000 000]. 0 signifies that the message never expires. Warning! After this delay, a message is considered as expired, and is no longer available in the topic or queue. When developing interfaces it is advised to set this parameter to zero.
JMSPRIORITY	Write	Relative Priority of the message: 0 (lowest) to 9 (highest).
SENDMESSAGETYPE	Write	Type of message to send (1 -> BytesMessage, 2 ->TextMessage).
JMSTYPE	Write	Optional name of the message.
CLIENTID	Read	Subscriber identification string. This option is described only for JMS compatibility. Not used for publication.
DURABLE	Read	D: Session is durable. Indicates that the subscriber definition remains on the router after disconnection.
MESSAGEMAXNUMBER	Read	Maximum number of messages retrieved [0 .. 4 000 000 000]. 0: All messages are retrieved.
MESSAGETIMEOUT	Read	Time to wait for the first message in milliseconds [0 .. 4 000 000 000]. if MESSAGETIMEOUT is equal to 0, then there is no timeout. MESSAGETIMEOUT and MESSAGEMAXNUMBER cannot be both equal to zero. if MESSAGETIMEOUT= 0 and MESSAGEMAXNUMBER =0, then MESSAGETIMEOUT takes the value 1. Warning! An interface may retrieve no message if this timeout value is too small.
NEXTMESSAGETIMEOUT	Read	Time to wait for each subsequent message in milliseconds [0 .. 4 000 000 000]. The default value is 1000. Warning! An interface may retrieve only part of the messages available in the topic or the queue if this value is too small.
MESSAGESELECTOR	Read	Message selector in ISO SQL syntax. See Section 22.7.1, "Using JMS Properties" for more information on message selectors.

22.7 JMS Standard Properties

This section describes the JMS properties contained in the message header and how to use them.

In Oracle Data Integrator, pseudo-columns corresponding to the JMS Standard properties should be declared in accordance with the descriptions provided in Table 22–3.

The JMS type and access mode columns refer to the use of these properties in Oracle Data Integrator or in Java programs. In Oracle Data Integrator, some of these properties are used through the IKM options, and the pseudo-column values should not be set by the interfaces.

For more details on using these properties in a Java program, see `http://java.sun.com/products/jms/`.

Table 22–3 Standard JMS Properties of Message Headers

Property	JMS Type	Access (Read/Write)	Description
JMSDestination	JMS String	R	Name of the destination (topic or queue) of the message.
JMSDeliveryMode	JMS Integer	R/W (set by IKM option)	Distribution mode: 1 = Not Persistent or 2 = Persistent. A persistent message is never lost, even if a router crashes. When sending messages, this property is set by the JMSDELIVERYMODE KM option.
JMSMessageID	JMS String	R	Unique Identifier for a message. This identifier is used internally by the router.
JMSTimestamp	JMS Long	R	Date and time of the message sending operation. This time is stored in a UTC standard format (1).
JMSExpiration	JMS Long	R/W (set by IKM option)	Message expiration date and time. This time is stored in a UTC standard format (1). To set this property the JMSEXPIRATION KM option must be used.
JMSRedelivered	JMS Boolean	R	Indicates if the message was resent. This occurs when a message consumer fails to acknowledge the message reception.
JMSPriority	JMS Int	R/W	Name of the destination (topic or queue) the message replies should be sent to.
JMSCorrelationID	JMS String	R/W	Correlation ID for the message. This may be the JMSMessageID of the message this message generating this reply. It may also be an application-specific identifier.

Table 22–3 (Cont.) Standard JMS Properties of Message Headers

Property	JMS Type	Access (Read/Write)	Description
JMSType	JMS String	R/W (set by IKM option)	Message type label. This type is a string value describing the message in a functional manner (for example `SalesEvent`, `SupportProblem`). To set this property the JMSTYPE KM option must be used.

Table 22–4 lists the optional JMS-defined properties in the JMS standard.

Table 22–4 Standard JMS Properties of Message Headers

Property	JMS Type	Access (Read/Write)	Description
JMSXUserID	JMS String	R	Client User ID.
JMSXAppID	JMS String	R	Client Application ID.
JMSSXProducerTXID	JMS String	R	Transaction ID for the production session. This ID is the same for all the messages sent to a destination by a producer between two JMS commit operations.
JMSSXConsumerTXID	JMS String	R	Transaction ID for current consumption session. This ID is the same of a batch of message read from a destination by a consumer between two JMS commit read operations.
JMSXRcvTimestamp	JMS Long	R	Message reception date and time. This time is stored in a UTC standard format (1).
JMSXDeliveryCount	JMS Int	R	Number of times a message is received. Always set to 1.
JMSXState	JMS Int	R	Message state. Always set to 2 (Ready).
JMSXGroupID	JMS String	R/W	ID of the group to which the message belongs.
JMSXGroupSeq	JMS Int	R/W	Sequence number of the message in the group of messages.

(1): The UTC (Universal Time Coordinated) standard is the number of milliseconds that have elapsed since January 1st, 1970

22.7.1 Using JMS Properties

In addition to their contents, messages have a set of properties attached to them. These may be provider-specific, application-specific (user defined) or JMS Standard Properties.

JMS properties are used in Oracle Data Integrator as complementary information to the message, and are used, for example, to filter the messages.

22.7.1.1 Declaring JMS Properties

When Defining the JMS Datastores, you must append pseudo-columns corresponding to the JMS properties that you want to use in your interfaces. See Declaring JMS Properties as Pseudo-Columns for more information.

22.7.1.2 Filtering on the Router

With this type of filtering, the filter is specified when sending the JMS read query. Only messages matching the message selector filter are retrieved. The message selector is specified in Oracle Data Integrator using the MESSAGE_SELECTOR KM option

> **Note:** Router filtering is not a JMS mandatory feature. It may be unavailable. Please confirm that it is available by reviewing the JMS provider documentation.

The MESSAGE_SELECTOR is programmed in an SQL WHERE syntax. Comparison, boolean and mathematical operators are supported:

`+, -, *, /, =, >, <, <>, >=, <=, OR, AND, BETWEEN, IN, NOT, LIKE, IS NULL.`

> **Notes:**
>
> - The `IS NULL` clause handles properties with an empty value but does not handle nonexistent application-specific properties.
>
>   For example, if the selector `COLOR IS NULL` is defined, a message with the application-specific property COLOR specified with an empty value is consumed correctly. Another message in the same topic/queue without this property specified would raise an exception.

Examples

Filter all messages with priority greater than 5

```
JMSPriority > 5
```

Filter all messages with priority not less than 6 and with the type Sales_Event.

```
NOT JMSPriority < 6 AND JMSType = 'Sales_Event'
```

22.7.1.3 Filtering on the Client

Filtering is performed after receiving the messages, and is setup by creating a standard Oracle Data Integrator interface filter which must be executed on the staging area. A filter uses pseudo-columns from the source JMS datastore. The pseudo-columns defined in the Oracle Data Integrator datastore represent the JMS properties. See Declaring JMS Properties as Pseudo-Columns for more information. Note that messages filtered this way are considered as consumed from the queue or topic.

22.7.1.4 Using Property Values as Source Data

It is possible to use the values of JMS properties as source data in an interface. This is carried out by specifying the pseudo-columns of the source JMS datastore in the mapping. See Declaring JMS Properties as Pseudo-Columns for more information.

22.7.1.5 Setting Properties when Sending a Message

When sending messages it is possible to specify JMS properties by mapping values of the pseudo-columns in an interface targeting a JMS datastore. Certain properties may be set using KM options. See Section 22.7, "JMS Standard Properties" for more information.

23

JMS XML

This chapter describes how to work with Java Message Services (JMS) with a XML payload in Oracle Data Integrator.

This chapter includes the following sections:

23.1 Introduction

Oracle Data Integrator provides a simple and transparent method to integrate JMS destinations. This chapter focuses on processing JMS messages with a XML payload. For text payload processing in batch mode, refer to Chapter 22, "JMS".

23.1.1 Concepts

The JMS XML Knowledge Modules apply to most popular JMS compliant middleware, including Oracle Service Bus, Sonic MQ, IBM Websphere MQ, and so forth. Most of these Knowledge Modules include transaction handling to ensure message delivery.

23.1.1.1 JMS Message Structure

See Section 22.1.1.1, "JMS Message Structure" for information about the JMS message structure.

23.1.1.2 Using a JMS Destination

Oracle Data Integrator is able to process XML messages that are delivered by a JMS destination. Each message is considered as a container for XML data and is handled through the JMS XML Queue or JMS XML Topic technology.

With JMS XML Queue/JMS XML Topic technologies, each messages payload contains a complete XML data structure. This structure is mapped into a relational schema (XML Schema) that appears as a model, using the Oracle Data Integrator XML Driver.

Note: This method is extremely similar to XML files processing. In JMS XML, the message payload is the XML file. See Chapter 5, "XML Files" and Appendix B, "Oracle Data Integrator Driver for XML Reference" for more information about XML Files processing and the XML Driver.

In the topology, each JMS destination is defined as a JMS XML Topic/Queue data server with a single physical schema. A data server/physical schema pair will be declared for each topic or queue delivering message in the XML format.

The structure of the XML message mapped into a relational structure (called the XML schema) appears as a data model. Each datastore in this model represents a portion (typically, an element type) in the XML file.

Processing Messages

As each XML message corresponds to an Oracle Data Integrator model, the entire model must be used and loaded as one single unit when a JMS XML message is consumed or produced. The processing unit for an XML message is the package.

It is not possible to declare the properties or header fields of the message in the model or use them as columns in an interface. They still can be used in message selectors, or be set through KM options.

Consuming an XML message

Processing an incoming XML message is performed in packages as follows:

1. *Synchronize the JMS message to the XML schema*: This operation reads the message and generates the XML schema. This is usually performed by the first interface accessing the message.
2. *Extract the data*: A sequence of interfaces use datastores from the XML schema as sources. This data is usable until the session is terminated, another message is read by a new *Synchronize* action, or the *Commit JMS Read* is performed.
3. *Commit JMS Read*: This operation validates the message consumption and deletes the XML schema. It should be performed by the last interface which extracts data from the XML message.

Producing an XML message

To produce an XML message, a package must be designed to perform the following tasks:

1. *Initialize the XML schema*: This operation creates an empty XML schema corresponding to the XML message to generate. This operation is usually performed in the first interface loading the structure.
2. *Load the data*: A sequence of interfaces loads data into the XML schema.
3. *Synchronize the XML schema to JMS*: This operation converts the XML schema to an XML message, and sends it to the JMS destination. This operation is usually performed by the last interface loading the schema.

Filtering Messages

It is possible to filter messages from a JMS XML destination by defining a *Message Selector* (MESSAGE_SELECTOR KM option) to filter messages on the server. This type of filter can use only the properties or header fields of the message. The filter is

processed by the server, reducing the amount of information read by Data Integrator. It is also possible to filter data in the interface using data extracted from the XML schema. These filters are processed in Data Integrator, after the message is synchronized to the XML schema.

23.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 23–1 for handling XML messages.

Table 23–1 JMS XML Knowledge Modules

Knowledge Module	Description
IKM SQL to JMS XML Append	Integrates data into a JMS compliant message queue or topic in XML format from any ANSI SQL-92 standard compliant staging area.
LKM JMS XML to SQL	Loads data from a JMS compliant message queue or topic in XML to any ANSI SQL-92 standard compliant database used as a staging area.

23.2 Installation and Configuration

Make sure you have read the information in this section before you start using the JMS Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

23.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

23.2.2 Technology Specific Requirements

The JMS destinations are usually accessed via a JNDI service. The configuration and specific requirements for JNDI and JMS depends on the JMS Provider you are connecting to. Refer to the JMS Provider specific documentation for more details.

23.2.3 Connectivity Requirements

This section lists the requirements for connecting to a JMS XML database.

Oracle Data Integrator does not include specific drivers for JMS providers. Refer to the JMS Provider documentation for the connectivity requirement of this provider.

XML Configuration

XML content is accessed through the Oracle Data Integrator JDBC for XML driver. The driver is installed with Oracle Data Integrator.

Ask your system administrator for the location of the DTD file describing the XML content.

23.3 Setting up the Topology

Setting up the Topology consists of:

1. Creating a JMS XML Data Server
2. Creating a JMS XML Physical Schema

23.3.1 Creating a JMS XML Data Server

An JMS XML data server corresponds to one JMS provider/router that is accessible through your local network.

There are two types of JMS XML data servers: JMS Queue XML and JMS Topic XML.

- A *JMS Queue XML data server* is used to connect a single queue in the JMS router to integrate XML messages.
- A *JMS Topic XML data server* is used to connect a single Topic in the JMS router to integrate XML messages.

The Oracle Data Integrator JMS driver loads the messages that contains the XML content into a relational schema in memory. This schema represents the hierarchical structure of the XML message and enables unloading the relational structure back to the JMS messages.

23.3.1.1 Creation of the Data Server

Create a data server either for the JMS Queue XML technology or for the JMS Topic XML technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The creation process for a JMS XML Queue or JMS Topic XML data server is identical to the creation process of an XML data server except that you need to define a JNDI connection with JMS XML specific information in the JNDI URL. See Section 5.3.1, "Creating an XML Data Server" for more information.

This section details only the fields required or specific for defining a JMS Queue XML or JMS Topic XML data server.

1. In the Definition tab:
   - Name: Name of the data server as it will appear in Oracle Data Integrator.
   - User/Password: Not used here. Leave these fields empty.
2. In the JNDI tab:
   - JNDI Authentication: From the list, select the authentication mode.
   - JNDI User: Enter the username to connect to the JNDI directory (not mandatory).
   - Password: This user's password (not mandatory).
   - JNDI Protocol: From the list, select the JNDI protocol (not mandatory).
   - JNDI Driver: Name of the initial context factory java class to connect to the JNDI provider, for example:

     ```
     com.sun.jndi.ldap.LdapCtxFactory
     ```

- JNDI URL: `<JMS_RESOURCE>?d=<DTD_FILE>&s=<SCHEMA>&JMS_DESTINATION=<JMS_DESTINATION_NAME>`.

  The JNDI URL properties are described inTable 23–2.

- JNDI Resource: Logical name of the JNDI resource corresponding to your JMS Queue (or Topic) connection factory.

Note: Specify `QueueConnectionFactory` if you want to access a message queue and TopicConnectionFactory if you want to access a topic. Note that these parameters are specific to the JNDI directory.

Table 23–2 JNDI URL Properties

Parameter	Value	Notes
d	<DTD File location>	DTD File location (relative or absolute) in UNC format. Use slash "/" in the path name and not backslash "\" in the file path. This parameter is mandatory.
re	<Root element>	Name of the element to take as the root table of the schema. This value is case sensitive. This parameter can be used for reverse-engineering a specific message definition from a WSDL file, or when several possible root elements exist in a XSD file.
ro	true | false	If true, the XML file is opened in read only mode.
s	<schema name>	Name of the relational schema where the XML file will be loaded.This value must match the one set for the physical schema attached to this data server. This parameter is mandatory.
cs	true | false	Load the XML file in case sensitive or insensitive mode. For case insensitive mode, all element names in the DTD file should be distinct (Ex: Abc and abc in the same file are banned). The case sensitive parameter is a permanent parameter for the schema. It CANNOT be changed after schema creation. Please note that when opening the XML file in insensitive mode, case will be preserved for the XML file.
JMSXML_ ROWSEPARATOR	5B23245D	Hexadecimal code of the string used as a line separator (line break) for different XML contents. Default value is 5B23245D which corresponds to the string [#$].
JMS_ DESTINATION	JNDI Queue name or Topic name	JNDI Name of the JMS Queue or Topic. This parameter is mandatory.

Example

If using an LDAP directory as the JNDI provider, you should use the following parameters:

- JNDI Driver: `com.sun.jndi.ldap.LdapCtxFactory`
- JNDI URL: `ldap://<ldap_host>:<port>/<dn>?d=<DTD_FILE>&s=<SCHEMA>&JMS_DESTINATION=<JMS_DESTINATION_NAME>`
- JNDI Resource: `<Name of the connection factory>`

23.3.2 Creating a JMS XML Physical Schema

Create a JMS XML physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Note: For the name of the Schema and Work Schema use the schema name defined in the `s=<schema name>` property of the JNDI URL of the JMS Queue XML or JMS Topic XML data server.

Note: Only one physical schema is required per JMS XML data server.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

23.4 Setting Up an Integration Project

Setting up a project using JMS XML follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the following knowledge modules into your project for getting started with JMS XML:

- IKM SQL to JMS XML Append
- LKM JMS XML to SQL

23.5 Creating and Reverse-Engineering a JMS XML Model

This section contains the following topics:

23.5.1 Create a JMS XML Model

Create a JMS Queue XML or JMS Topic XML Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

A JMS Queue XML or JMS Topic XML Model corresponds to a set of datastores, with each datastore representing an entry level in the XML file. The models contain datastores describing the structure of the JMS messages. A model contains the message structure of one topic or one queue. This model's structure is reverse-engineered from the DTD or the XML file specified in the data server definition, using standard reverse-engineering.

23.5.2 Reverse-Engineering a JMS XML Model

JMS XML supports Standard reverse-engineering - which uses only the abilities of the XML driver.

To perform a Standard Reverse-Engineering on JMS Queue XML or JMS Topic XML use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Oracle Data Integrator will automatically add the following columns to the tables generated from the XML data:

- Primary keys (PK columns) for parent-child relationships
- Foreign keys (FK columns) for parent-child relationships
- Order identifier (ORDER columns) to enable the retrieval of the order in which the data appear in the XML file.

These extra columns enable the hierarchical XML structure's mapping in a relational structure stored in the schema. See d Appendix B, "Oracle Data Integrator Driver for XML Reference" for more information.

23.6 Designing an Interface

The KM choice for an interface or a check determines the abilities and performance of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning XML messages.

23.6.1 Loading Data from a JMS XML Source

JMS XML can be used as a source or a target in an interface. Data from an XML message Queue or Topic can be loaded to any ANSI SQL-92 standard compliant database used as a staging area. The LKM choice in the Interface Flow tab to load data between JMS XML and another type of data server is essential for successful data extraction.

Oracle Data Integrator provides the LKM JMS XML to SQL for loading data from a JMS compliant message queue or topic in XML to any ANSI SQL-92 standard compliant database used as a staging area. This LKM uses the Agent to read selected messages from the source queue/topic and write the result in the staging temporary table created dynamically.

To ensure message delivery, the message consumer (or subscriber) does not commit the read until the data is actually integrated into the target by the IKM.

Consider using this LKM if one of your source datastores is an XML JMS message.

In order to load XML messages from a JMS provider, the following steps must be followed:

- The first interface reading the XML message from the JMS XML source must use the LKM JMS XML to SQL with the SYNCHRO_JMS_TO_XML LKM option set to `Yes`. This option creates and loads the XML schema from the message retrieved from the queue or topic.
- The last interface should commit the message consumption by setting the JMS_COMMIT to `Yes`.

Table 23–3 lists the JMS specific options of this knowledge module.

23.6.2 Integrating Data in a JMS XML Target

Oracle Data Integrator provides the IKM SQL to JMS XML Append that implements optimized data integration strategies for JMS XML. This IKM integrates data into a

JMS compliant message queue or topic in XML format from any ANSI SQL-92 standard compliant staging area.

To use this IKM, the staging area must be different from the target.

In order to integrate XML data into a JMS XML target, the following steps must be followed:

- The first interface loading the XML schema must provide a value for the ROOT_TABLE (it is the model's table that corresponds to the root element of the XML file), and also set the INITIALIZE_XML_SCHEMA option to `Yes`.

  Note: The root table of the XML schema usually corresponds to the datastore at the top of the hierarchy tree view of the JMS XML model. Therefore the ROOT_TABLE parameter should take the value of the resource name for this datastore.

- The interfaces should load the datastores in the hierarchy order, starting by the top of the hierarchy and going down. The interfaces loading subsequent levels of the XML schema hierarchy should load the foreign key column linking the current hierarchy level to a higher one.

  For example, when loading the second level of the hierarchy (the one under the root table), the foreign key column should be set to `'0'` (Zero), as it is the value that is set by the IKM in the root table primary key when the root table is initialized.

- The last interface should send the XML schema to the JMS provider by setting the SYNCHRO_JMS_TO_XML parameter to `Yes`.

Example

An XML file format generates a schema with the following hierarchy of datastores:

```
+ GEOGRAPHY_DIM (GEO_DIMPK, ...)
  |
  +--- COUNTRY (GEO_DIMFK, COUNTRYPK, COUNTRY_NAME, ...)
       |
       +--- REGION (COUNTRYFK, REGIONPK, REGION_NAME, ...)
```

In this hierarchy, GEOGRAPHY_DIM is the root table, and its GEOGRAPHY_DIMPK column is set to `'0'` at initialization time. The tables should be loaded in the GEOGRAPHY_DIM, COUNTRY, REGION sequence.

- When loading the second level of the XML hierarchy (COUNTRY) make sure that the FK field linking this level to the root table level is set to `'0'`. In the model above, when loading COUNTRY, we must load the COUNTRY.GEOGRAPHY_DIMFK set to `'0'`.
- You must also link the records of REGION to the COUNTRY level by loading the REGION.COUNTRYFK column with values that correspond to a parent record in COUNTRY (having `REGION.COUNTRYFK = COUNTRY.COUNTRYPK`).

For more information on loading data to XML schemas, see Appendix B, "Oracle Data Integrator Driver for XML Reference".

Table 23–3 lists the JMS specific KM options of this IKM. Options that are specific to XML messages are in bold.

JMS XML Knowledge Modules Options

Table 23–3 lists the KM options for the LKM and IKM for JMS XML. Options that are specific to XML messages are in bold.

Although most options are the same for the LKM and IKM, there are only few differences:

- The INITIALIZE_XML_SCHEMA and ROOT_TABLE options are only provided for the IKM.
- The DELETE_TEMPORARY_OBJECTS and JMS_COMMIT options are only provided for the LKM.
- Set JMS_COMMIT to `Yes` to commit the message consumption on the Router (JMS XML).

Table 23–3 JMS Specific KM Options

Option	Used to	Description
CLIENTID	Read	Subscriber identification string. Not used for publication.
DURABLE	Read	D: Session is durable. Indicates that the subscriber definition remains on the router after disconnection.
INITIALIZE_XML_SCHEMA	Write	Initializes an empty XML schema. This option must be set to `YES` for the first interface loading the schema.
JMSDELIVERYMODE	Write	JMS delivery mode (1: Non Persistent, 2: Persistent). A persistent message remains on the server and is recovered on server crash.
JMSEXPIRATION	Write	Expiration delay in milliseconds for the message on the server [0..4 000 000 000]. 0 signifies that the message never expires. **Warning!** After this delay, a message is considered as expired, and is no longer available in the topic or queue. When developing interfaces it is advised to set this parameter to zero.
JMSPRIORITY	Write	Relative Priority of the message: 0 (lowest) to 9 (highest).
JMSTYPE	Write	Optional name of the message.
MESSAGEMAXNUMBER	Read	Maximum number of messages retrieved [0 .. 4 000 000 000]. 0: All messages are retrieved.
MESSAGESELECTOR	Read	Message selector in ISO SQL syntax for filtering on the router. See Section 22.7.1, "Using JMS Properties" for more information on message selectors.
MESSAGETIMEOUT	Read	Time to wait for the first message in milliseconds [0 .. 4 000 000 000]. If MESSAGETIMEOUT is equal to 0, then there is no timeout. MESSAGETIMEOUT and MESSAGEMAXNUMBER cannot be both equal to zero. If `MESSAGETIMEOUT= 0` and `MESSAGEMAXNUMBER =0`, then MESSAGETIMEOUT takes the value 1. **Warning!** An interface may retrieve no message if this timeout value is too small.

Table 23–3 (Cont.) JMS Specific KM Options

Option	Used to	Description
NEXTMESSAGETIMEOUT	Read	Time to wait for each subsequent message in milliseconds [0 .. 4 000 000 000]. The default value is 1000. **Warning!** An interface may retrieve only part of the messages available in the topic or the queue if this value is too small.
ROOT_TABLE	Write	Resource name of the datastore that is the root of the XML model hierarchy.
SENDMESSAGETYPE	Write	Type of message to send (1 -> BytesMessage, 2 ->TextMessage).
SYNCHRO_XML_TO_JMS	Write	Generates the XML message from the XML schema, and sends this message. This option must be set to `YES` for the last interface that writes to the schema XML.

24

LDAP Directories

This chapter describes how to work with LDAP directories in Oracle Data Integrator.

This chapter includes the following sections:

24.1 Introduction

Oracle Data Integrator supports LDAP directories integration using the Oracle Data Integrator Driver for LDAP.

24.1.1 Concepts

The LDAP concepts map the Oracle Data Integrator concepts as follows: An LDAP directory tree, more specifically the entry point to this LDAP tree, corresponds to a data server in Oracle Data Integrator. Within this data server, a single schema maps the content of the LDAP directory tree.

The Oracle Data Integrator Driver for LDAP (LDAP driver) loads the hierarchical structure of the LDAP tree into a relational schema. This relational schema is a set of tables that can be queried or modified using standard SQL statements.

The relational schema is reverse-engineered as a data model in ODI, with tables, columns, and constraints. This model is used like a normal relational data model in ODI. Any changes performed in the relational schema data (insert/update) is immediately impacted by the driver in the LDAP data.

See Appendix A, "Oracle Data Integrator Driver for LDAP Reference" for more information on this driver.

24.1.2 Knowledge Modules

Oracle Data Integrator does not provide specific Knowledge Modules (KM) for the LDAP technology. You can use LDAP as a SQL data server. LDAP data servers support both the technology-specific KMs sourcing or targeting SQL data servers, as

well as the generic KMs. See Chapter 4, "Generic SQL" or the technology chapters for more information on these KMs.

24.2 Installation and Configuration

Make sure you have read the information in this section before you start working with the LDAP technology.

- System Requirements
- Technologic Specific Requirements
- Connectivity Requirements

24.2.1 System Requirements

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

24.2.2 Technologic Specific Requirements

There are no technology-specific requirements for using LDAP directories in Oracle Data Integrator.

24.2.3 Connectivity Requirements

This section lists the requirements for connecting to LDAP database.

Oracle Data Integrator Driver for LDAP

LDAP directories are accessed through the Oracle Data Integrator Driver for LDAP. This JDBC driver is installed with Oracle Data Integrator.

To connect to an LDAP directory you must ask the system administrator for the following connection information:

- The URL to connect to the directory
- The User and Password to connect to the directory
- The Base Distinguished Name (Base DN). This is the location in the LDAP tree that ODI will access.

You may also require a connection to the *Reference LDAP Tree* structure and to an *External Storage* database for the driver. See Appendix B, "Oracle Data Integrator Driver for XML Reference" for more information on these concepts and configuration parameters.

24.3 Setting up the Topology

Setting up the topology consists in:

1. Creating an LDAP Data Server

2. Creating a Physical Schema for LDAP

24.3.1 Creating an LDAP Data Server

An LDAP data server corresponds to an LDAP tree that is accessible to Oracle Data Integrator.

24.3.1.1 Creation of the Data Server

Create a data server for the LDAP technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining a LDAP data server:

1. In the Definition tab:
   - **Name**: Name of the data server that will appear in Oracle Data Integrator.
   - **User/Password**: Name and password of the LDAP directory user.
2. In the JDBC tab, enter the values according to the driver used:
   - **JDBC Driver**: `com.sunopsis.ldap.jdbc.driver.SnpsLdapDriver`
   - **JDBC URL**: The driver supports two URL formats:
     - `jdbc:snps:ldap?<property>=<value>[&<property>=<value>...]`
     - `jdbc:snps:ldap2?<property>=<value>[&<property>=<value>...]`

     These two URLs accept the key properties listed in Table 24–1. See Appendix A.3.1, "Driver Configuration" for a detailed description of these properties and for a comprehensive list of all JDBC driver properties.

Note: The first URL requires the LDAP directory password to be encoded. The second URL allows you to give the LDAP directory password without encoding it. It is recommended to use the first URL to secure the LDAP directory password.

Table 24–1 JDBC Driver Properties

Property	Value	Notes
ldap_auth	<authentication mode>	LDAP Directory authentication method. See the `auth` property in Table A–2
ldap_url	<LDAP URL>	LDAP Directory URL. See the `url` property in Table A–2
ldap_user	<LDAP user name>	LDAP Directory user name. See the `user` property in Table A–2
ldap_password	<LDAP user password>	LDAP Directory user password. This password must be encoded if using the jdbc:snps:ldap URL syntax. See the `password` property in Table A–2
lldap_basedn	<base DN>	LDAP Directory basedn. See the `basedn` property in Table A–2

URL Examples

To connect an Oracle Internet Directory on server `OHOST_OID` and port `3060`, using the user `orcladmin`, and accessing this directory tree from the basedn `dc=us,dc=oracle,dc=com` you can use the following URL:

```
jdbc:snps:ldap?ldap_url=ldap://OHOST_OID:3060/
&ldap_basedn=dc=us,dc=oracle,dc=com
&ldap_password=ENCODED_PASSWORD
&ldap_user=cn=orcladmin
```

24.3.2 Creating a Physical Schema for LDAP

Create an LDAP physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

24.4 Setting Up an Integration Project

Setting up a Project using the LDAP database follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The recommended knowledge modules to import into your project for getting started are the following:

- LKM SQL to SQL
- LKM File to SQL
- IKM SQL Control Append

24.5 Creating and Reverse-Engineering an LDAP Directory

This section contains the following topics:

- Create an LDAP Model
- Reverse-Engineering an LDAP Model

24.5.1 Create an LDAP Model

A data model groups a set of datastores. Each datastore represents in the context of a directory a class or group of classes. Typically, classes are mapped to tables and attributes to column. See Appendix A.2.1, "LDAP to Relational Mapping" for more information.

Create an LDAP Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

24.5.2 Reverse-Engineering an LDAP Model

LDAP supports standard reverse-engineering, which uses only the abilities of the LDAP driver.

When the reverse-engineering process of the LDAP driver translates the LDAP tree into a relational database structure, it constructs tables from sets of objects in the tree.

The names of these tables must reflect this original structure in order to maintain the mapping between the two. As a result, the table names are composed of the original LDAP object names that may be extremely long and not appropriate as datastore names in integration interfaces.

The solution consists in creating an alias file that contains a list of short and clear table name aliases. See Appendix A.3.3, "Table Aliases Configuration" for more information.

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on LDAP use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

The standard reverse-engineering process will automatically map the LDAP tree contents to a relational database structure. Note that these tables automatically include primary key and foreign key columns to map the directory hierarchy.

The reverse-engineering process also creates a ROOT table that represents the root of the LDAP tree structure from the LDAP entry point downwards.

See Appendix A.2, "LDAP Processing Overview" for more information.

24.6 Designing an Interface

You can use LDAP entries as a source or a target of an integration interface.

The KM choice for an interface or a check determines the abilities and performances of this interface or check. The recommendations in this section help in the selection of the KM for different situations concerning an LDAP data server.

24.6.1 Loading Data from and to LDAP

An LDAP directory can be used as an interface's source or target. The LKM choice in the Interface Flow tab that is used to load data between LDAP entries and other types of data servers is essential for the performance of the interface.

24.6.1.1 Loading Data from an LDAP Directory

Use the Generic SQL KMs or the KMs specific to the other technology involved to load data from an LDAP database to a target or staging area database.

Table 24–2 lists some examples of KMs that you can use to load from an LDAP source to a staging area.

Table 24–2 KMs to Load from LDAP to a Staging Area

Staging Area	KM	Notes
Microsoft SQL Server	LKM SQL to MSSQL (BULK)	Uses SQL Server's bulk loader.
Oracle	LKM SQL to Oracle	Faster than the Generic LKM (Uses Statistics)
Sybase	LKM SQL to Sybase ASE (BCP)	Uses Sybase's bulk loader.
All	LKM SQL to SQL	Generic KM

24.6.1.2 Loading Data to an LDAP Directory

It is not advised to use an LDAP directory as a staging area.

24.6.2 Integrating Data in an LDAP Directory

LDAP can be used as a target of an interface. The IKM choice in the Interface Flow tab determines the performances and possibilities for integrating.

Use the Generic SQL KMs or the KMs specific to the other technology involved to integrate data in an LDAP directory.

Table 24–3 lists some examples of KMs that you can use to integrate data from a staging area to an LDAP target.

Table 24–3 KMs to Integrate Data in an LDAP Directory

Mode	KM	Notes
Append	IKM SQL to SQL Append	Generic KM

24.7 Troubleshooting

This section provides information on how to troubleshoot problems that you might encounter when using LDAP in Oracle Data Integrator. It contains the following topics:

- SQL operations (insert, update, delete) performed on the relational model are not propagated to the LDAP directory.

  You are probably using an external RDBMS to store your relational model.

- `java.util.MissingResourceException: Can't find bundle for base name ldap_....`

  The property bundle file is missing, present in the incorrect directory or the filename is incorrect.

- `java.sql.SQLException: A NamingException occurred saying: [LDAP: error code 32 ....`

  The connection property bundle is possibly incorrect. Check the property values in the bundle files.

- `java.sql.SQLException: A NamingException occurred saying: [LDAP: error code 49 - Invalid Credentials]`

  The authentication property is possibly incorrect. Check the password.

- `java.sql.SQLException: Exception class javax.naming.NameNotFoundException occurred saying: [LDAP: error code 32 - No Such Object].`

  The LDAP tree entry point is possibly incorrect. Check the target DistinguishedName in the LDAP URL.

- `java.sql.SQLException: No suitable driver`

  This error message indicates that the driver is unable to process the URL is registered. The JDBC URL is probably incorrect. Check that the URL syntax is valid. See Section A.3, "Installation and Configuration".

25

Oracle Changed Data Capture Adapters

This chapter describes how to work with Oracle Changed Data Capture Adapters as well as with Attunity Stream in order to integrate changes captured on legacy sources using Oracle Data Integrator.

This chapter includes the following sections:

25.1 Introduction

Oracle Changed Data Capture Adapters offer log-based change data capture (CDC) for enterprise data sources such as CICS, VSAM, Tuxedo, IMS DB, and IMS TM. Captured changes are stored in a storage called Staging Area (which is different from the Oracle Data Integrator interfaces' staging areas).

Attunity Stream is part of the *Attunity Integration Suite (AIS)* and provides the same features as the Oracle Changed Data Capture Adapters. In this section, we will refer to both products as *Attunity Stream*.

The Attunity Stream Staging Area contains the Change Tables used by Attunity Stream to store changes captured from the sources. It maintains the last position read by Oracle Data Integrator (This is the Attunity Stream *Context*, which is different from the Oracle Data Integrator Context concept) and starts at this point the next time a request from Oracle Data Integrator is received. The Change Tables are accessed through Attunity Stream Datasources.

Oracle Data Integrator uses Attunity Stream datasources as a sources of integration interfaces. They cannot be used as target or staging area. Journalizing or data quality check is not possible on this technology.

25.1.1 Concepts

The Attunity Stream concepts map the Oracle Data Integrator concepts as follows: One Workspace within an Attunity Agent (or Daemon) listening on a port corresponds to one data server in Oracle Data Integrator. Within this Daemon, each Datasource (or Datasource/Owner pair) corresponds to one ODI Physical Schema. In each

datasource, the Change Tables appear as ODI Datastores in an ODI model based on the Attunity technology.

25.1.2 Knowledge Modules

Oracle Data Integrator provides the LKM Attunity to SQL for handling Attunity Stream data. The KMs use Attunity Stream specific features.

The Oracle Data Integrator CDC Knowledge Module provides integration from Attunity Stream Staging Areas via a JDBC interface. It is able to:

- Read Attunity Stream data from Attunity Stream Data Sources.
- Load this Attunity Stream data into an ANSI SQL-92 compliant database used as a staging area in Oracle Data Integrator staging area.
- Handle the Attunity Stream Context to ensure consistent consumption of the changes read.

Using the data provided in the Attunity staging area, the Oracle CDC KM cleans the working environment (dropping temporary tables), determines and saves Attunity Stream Context information, loads the journalized data into the collect table and purges the loaded data from the journal.

Note: Although Attunity Stream is used to capture changes in source systems, it is used as a regular JDBC source (only an LKM is used). The Oracle Data Integrator journalizing framework (JKM) is not used for this technology.

25.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Oracle Knowledge Modules:

- System Requirements
- Technology Specific Requirements
- Connectivity Requirements

25.2.1 System Requirements

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

25.2.2 Technology Specific Requirements

Please review the Attunity Streams or Oracle Changed Data Capture Adapters documentation for the requirements and instruction for installing and setting up Streams for your source servers.

25.2.3 Connectivity Requirements

In order to use the Attunity Stream technology, you must first install the Attunity drivers in the drivers directory of your Oracle Data Integrator installation and restart ODI. See "Add Additional Drivers and Open Tools" in the *Oracle Fusion Middleware Installation Guide for Oracle Data Integrator*.

The driver files include the following: `nvjdbc2.jar`, `nvapispy2.jar`, `nvlog2.jar`.

25.3 Setting up the Topology

Setting up the Topology consists in:

1. Creating an Attunity Stream Data Server
2. Creating an Attunity Stream Physical Schema

25.3.1 Creating an Attunity Stream Data Server

An Attunity Stream data server corresponds to the server and workspace storing the Attunity Stream datasources.

25.3.1.1 Creation of the Data Server

Create a data server for the Attunity Stream technology using the standard procedure, as described in "Creating a Data Server" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*. This section details only the fields required or specific for defining an Oracle data server:

1. In the Definition tab:
   - **User**: User profile to connect the workspace. If you are using anonymous access or specifying the user and password on the URL, leave this field and the JDBC Password field empty.
   - **Password**: Master password for the user profile.
2. In the JDBC tab:
   - **JDBC Driver**: `com.attunity.jdbc.NvDriver`
   - **JDBC URL**: `jdbc:attconnect://<host_name>:<port>/<workspace>[;AddDefaultSchema=1][;<parameter>=<value>]`

     You can use in the URL the properties listed in:

Table 25–1 JDBC Attunity Driver Properties

Option	Description
<host_name>	Name of the machine running the Attunity daemon
<port>	Port that the daemon listens to
<workspace>	Daemon's workspace. Default is `Navigator`.
AddDefaultSchema=1	This parameter specifies that a schema shows the default owner name `public` if the data source does not natively support owners. It may be needed in some cases as Oracle Data Integrator makes use of the owner value.

Table 25–1 (Cont.) JDBC Attunity Driver Properties

Option	Description
<parameter>=<value>	Any parameter available for the JDBC driver. Note that it is not needed to specify the datasource using the DefTdpName driver parameter, as Oracle Data Integrator accesses the change tables using the full qualified syntax: `DATASOURCE:OWNER.TABLE_NAME`

For more information on the JDBC URL connection details, see the *Oracle Application Server CDC Adapters Installation Guide*.

25.3.2 Creating an Attunity Stream Physical Schema

Create an Attunity Stream physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

This physical schema represents the Attunity Stream datasource from which you want to read the changed data. While defining the physical schema, the list of datasources and owners available for your workspace is displayed, provided that the data server is correctly configured. Public is displayed if no datasources and owners exist.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

25.4 Setting Up an Integration Project

Setting up a project using the Attunity Stream follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

It is recommended to import the LKM Attunity to SQL into your project for getting started with Attunity Stream.

25.5 Creating and Reverse-Engineering an Attunity Stream Model

This section contains the following topics:

- Create an Attunity Stream Model
- Reverse-engineer an Attunity Stream Model

25.5.1 Create an Attunity Stream Model

Create an Attunity Stream Model using the standard procedure, as described in "Creating a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

25.5.2 Reverse-engineer an Attunity Stream Model

Attunity Stream supports standard reverse-engineering. Standard reverse-engineering returns the change tables stored in the datasource as datastores. The change tables contain some CDC header columns in addition to the data columns used for integration. These columns include timestamps, table_name, operation, transactionID,

context, and so forth. See the Attunity Stream documentation for more information on the header columns content.

To perform a Standard Reverse-Engineering on Oracle use the usual procedure, as described in "Reverse-engineering a Model" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

25.6 Designing an Interface Using the LKM Attunity to SQL

Oracle Data Integrator uses Attunity Stream datasources as a sources of integration interfaces. They cannot be used as target or staging area. Journalizing or data quality check is not possible on this technology.

To create an integration interface, which loads Attunity Stream data into your Oracle Data Integrator integration project, run the following steps:

1. Create an integration interface with Attunity Stream source datastores.
2. Create joins, filters and mappings as usual. Note that joins between change tables are not allowed on the source. They should be performed on the interface's staging area.
3. In the Flow tab of the interface, select the source set containing the source change table(s) and select the **LKM Attunity to SQL**.
4. Set the KM options as follows:
   - DELETE_TEMPORARY_OBJECTS - Set this option to `No`, if you wish to retain temporary objects (files and scripts) after integration.
   - PK_LIST – Specify the list of source columns that holds the primary key of the journalized table. Use SQL syntax and separate with a comma (,) each column name without prefixing it by the table alias, for example `ORDER_ID, CUSTOMER_ID`

Note: When running an interface using this LKM, the changes are consumed from the change table. This KM does not support reading twice the same change.

26

Oracle GoldenGate

This chapter describes how to work with Oracle GoldenGate in order to capture changes on source transactional systems and replicate them in a staging server for consumption by Oracle Data Integrator interfaces.

This chapter includes the following sections:

26.1 Introduction

Oracle GoldenGate (OGG) product offers solutions that provide key business applications with continuous availability and real-time information. It provides guaranteed capture, routing, transformation and delivery across heterogeneous databases and environments in real-time.

Using the Oracle GoldenGate knowledge modules requires that you know and understand Oracle GoldenGate concepts and architecture. See the Oracle GoldenGate Documentation on OTN for more information.

26.1.1 Overview of the GoldeGate CDC Process

Oracle Data Integrator uses Oracle GoldenGate to replicate online data from a source database to a staging database. This staging database contains a copy of the source tables and the ODI Changed Data Capture (CDC) infrastructure, both loaded using Oracle GoldenGate.

The staging database is stored in an Oracle schema. The source database can be Oracle, Microsoft SQL Server, DB2 UDB, or Sybase ASE. In this chapter, <database> refers to any of these source database technologies.

Setting up CDC with GoldenGate is done using the following process:

1. A replica of the source tables is created in the staging database, using, for example, the Oracle Data Integrator Common Format Designer feature.
2. Oracle Data Integrator Changed Data Capture (CDC) is activated on these replicated tables using the JKM <database> to Oracle Consistent (OGG). Starting the journals creates Oracle GoldenGate configuration files and sets up a CDC infrastructure in the staging database. Note that no active process is started for capturing source data at that stage.

3. Using the generated configuration files, an Oracle GoldenGate Extract process is configured and started to capture changes from the source database, and corresponding Replicat processes are configured and started to replicate these changes into the staging database. Changes are replicated into both the replicated source table and the CDC infrastructure. GoldenGate can optionally be configured to perform the initial load of the source data into the staging tables.
4. ODI interfaces can source from the replicated tables and use captured changes seamlessly within any ODI scenario.

26.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules listed in Table 26–1 for replicating online data from a source to a staging database. Unlike other CDC JKMs, the Oracle GoldenGate JKMs journalize data in a staging Oracle database and not in the source server.

The JKM <database> to Oracle Consistent (OGG) performs the following tasks:

- Creates and manages the ODI CDC framework infrastructure on the replicated tables
- Generates the parameter files to setup the Oracle GoldenGate capture (Extract) and Delivery (Replicat) processes
- Provides extra steps to check the configuration of the source database and proposes tips to correct the configuration
- Generates a readme file explaining how to complete the setup

Table 26–1 Oracle GoldenGate Knowledge Modules

Knowledge Module	Description
JKM Oracle to Oracle Consistent (OGG)	Creates the infrastructure for consistent set journalizing on an Oracle staging server and generates the Oracle GoldenGate configuration for replicating data from an Oracle source to this staging server.
JKM DB2 UDB to Oracle Consistent (OGG)	Creates the infrastructure for consistent set journalizing on an Oracle staging server and generates the Oracle GoldenGate configuration for replicating data from an IBM DB2 UDB source to this staging server.
JKM Sybase ASE to Oracle Consistent (OGG)	Creates the infrastructure for consistent set journalizing on an Oracle staging server and generates the Oracle GoldenGate configuration for replicating data from a SybaseASE source to this staging server.
JKM MSSQL to Oracle Consistent (OGG)	Creates the infrastructure for consistent set journalizing on an Oracle staging server and generates the Oracle GoldenGate configuration for replicating data from a Microsoft SQL Server source to this staging server.

26.2 Installation and Configuration

Make sure you have read the information in this section before you start using the Oracle GoldenGate Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements

26.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

http://www.oracle.com/technology/products/oracle-data-integrator/index.html.

See also the Oracle GoldenGate documentation on OTN for source and staging database version platform support.

26.2.2 Technology Specific Requirements

In order to run the *Extract* and *Replicat* processes, Oracle GoldenGate must be installed on both the source and staging servers. Installing Oracle GoldenGate installs all of the components required to run and manage GoldenGate processes.

Oracle GoldenGate Manager Process must be running on each system before Extract or Replicat can be started, and must remain running during their execution for resource management.

Oracle GoldenGate has specific requirement and installation instructions that must be performed before starting the Extract and Replicat processes configured with the Oracle GoldenGate JKMs. See the Oracle GoldenGate Documentation on OTN for more information.

26.2.3 Connectivity Requirements

If the source database is Oracle, there are no connectivity requirements for using Oracle GoldenGate data in Oracle Data Integrator.

If the source database is IBM DB2 UDB, Microsoft SQL Server, or Sybase ASE, Oracle GoldenGate uses the ODBC driver to connect to the source database. You need to install the ODBC driver and to declare the data source in your system. You also need to set the data source name (DSN) in the KM option SRC_DSN.

Note: For Sybase source database only: When defining the data source name, you have to add the database server name to the datasource name as follows:

```
DSN_name@SYBASE_DBSERVER
```

26.3 Working with the Oracle GoldenGate JKMs

To use the JKM <database> to Oracle Consistent (OGG) in your Oracle Data Integrator integration projects, you need to perform the following steps:

1. Define the Topology
2. Create the Replicated Tables
3. Set Up an Integration Project
4. Configure CDC for the Replicated Tables
5. Configure and Start Oracle GoldenGate Processes

6. Design Interfaces Using Replicated Data

26.3.1 Define the Topology

This step consists in declaring in Oracle Data Integrator the staging data server, the source data server, as well as the physical and logical schemas attached to these servers.

To define the topology in this configuration, perform the following tasks:

1. Define the Staging Server
2. Create the Staging Physical Schema
3. Define the Source Data Server
4. Create the Source Physical Schema

26.3.1.1 Define the Staging Server

Create a data server for the Oracle technology as described in Section 2.3.1, "Creating an Oracle Data Server".

26.3.1.2 Create the Staging Physical Schema

Create an Oracle physical schema using the standard procedure, as described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Note: The physical schema defined in the staging server will contain in the data schema the changed records captured and replicated by the Oracle GoldenGate processes. The work schema will be used to store the ODI CDC infrastructure.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

26.3.1.3 Define the Source Data Server

You have to define a source data server from which Oracle GoldenGate will capture changes.

Create a data server for your source technology using the standard procedure. For more information, see the chapter corresponding to your source technology in this guide:

- Section 2.3.1, "Creating an Oracle Data Server"
- Section 6.3.1, "Creating a Microsoft SQL Server Data Server"
- Section 13.3.1, "Creating a DB2/400 Data Server"
- Chapter 15, "Sybase AS Enterprise"

This data server represents the source database instance.

26.3.1.4 Create the Source Physical Schema

Create a physical schema under the data server that you have created in Section 26.3.1.3, "Define the Source Data Server". Use the standard procedure, as

described in "Creating a Physical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in "Creating a Logical Schema" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and associate it in a given context.

26.3.2 Create the Replicated Tables

Oracle GoldenGate will replicate in the staging server the records changed in the source. In order to perform this replication, the source table structures must be replicated in the staging server.

To replicate these source tables:

1. Create a new Data Model using the Oracle technology. This model must use the logical schema created using the instructions in Create the Staging Physical Schema. See "Creating a Model" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information on model creation.

   Note that you do not need to reverse-engineer this data model.

2. Create a new diagram for this model and add to this diagram the source tables that you want to replicate.

3. Generate the DDL Scripts and run these scripts for creating the tables in the staging data server.

4. An initial load of the source data can be made to replicate this data into the staging tables. You can perform this initial load with ODI using the Generate Interface IN feature of Common Format Designer. Alternately, you can use Oracle GoldenGate to perform this initial load, by setting the USE_OGG_FOR_INIT JKM option to `Yes` when you Configure CDC for the Replicated Tables.

> **Note:** See "Working with Common Format Designer" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information on diagrams, generating DDL, and generating Interface IN features.

26.3.3 Set Up an Integration Project

Setting up a project using Oracle GoldenGate features follows the standard procedure. See "Creating an Integration Project" of the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

Depending on the technology of your source data server, import one of the following KMs into your project:

- JKM Oracle to Oracle Consistent (OGG)
- JKM DB2 UDB to Oracle Consistent (OGG)
- JKM Sybase ASE to Oracle Consistent (OGG)
- JKM MSSQL to Oracle Consistent (OGG)

26.3.4 Configure CDC for the Replicated Tables

Changed Data Capture must be configured for the replicated tables. This configuration is similar to setting up consistent set journalizing and is performed using the following steps.

1. Edit the data model that contains the replicated tables. In the Journalizing tab of the data model, set the Journalizing Mode to Consistent Set and select for the Journalizing KM JKM <database> to Oracle Consistent (OGG).

   Set the KM options as follows:

   - LOCAL_TEMP_DIR: Full path to a temporary folder into which the Oracle GoldenGate configuration files will be generated.
   - SRC_OGG_OBJECT_GROUP: Name of the Oracle GoldenGate source object group.
   - SRC_LSCHEMA: Name of the logical schema of the source model.
   - SRC_DB_USER: Source schema or database user name.
   - SRC_DB_PASSWORD: Source schema or database user password.
   - SRC_OGG_PATH: Oracle GoldenGate installation path on the source server.
   - SRC_DSN: Name of the data source. This KM option is required when the ODBC driver is used. Note that this option does not exist in the JKM Oracle to Oracle Consistent (OGG).

     Note: For Sybase users only: When defining the data source name, you have to add the database server name to the datasource name as follows:

     ```
     DSN_name@SYBASE_DBSERVER
     ```

   - STG_OGG_OBJECT_GROUP: Name of the Oracle GoldenGate staging object group.
   - STG_HOST_NAME: Name of the staging machine.
   - STG_MANAGER_PORT: TCP port on which the Oracle GoldenGate Manager process is listening on the staging machine.
   - STG_OGG_PATH: Oracle GoldenGate installation path on the staging server.
   - USE_OGG_FOR_INIT: Generate the Oracle GoldenGate processes to perform the initial load of the replicated tables. If you have performed this initial load using Oracle Data Integrator while Creating the Replicated Tables, you can leave this option to `NO`.

2. Select the tables that you want to replicate or the model if want to replicate all tables, right-click then select **Changed Data Capture** > **Add to CDC**.
3. Select the model, right-click then select **Changed Data Capture** > **Subscriber** > **Subscribe**. Add subscribers for this model.
4. Select the model, right-click then select **Changed Data Capture** > **Start Journal**. The JKM creates the CDC infrastructure and generates the configuration for Oracle GoldenGate.

You can review the result of the journal startup action:

- The Oracle GoldenGate configuration files, as well as a `Readme.txt` file are generated in the directory that is specified in the LOCAL_TEMP_DIR KM option. You can use these files to Configure and Start Oracle GoldenGate Processes.
- The CDC infrastructure is set up correctly. The journalized datastores appear in the Models accordion with a Journalizing Active flag. You can right-click the model and select **Changed Data Capture** > **Journal Data…** to access the journalized data for these datastores.

See "Working with Changed Data Capture" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more conceptual information and detailed instructions on CDC.

> **Note:** Although this CDC configuration supports consistent set journalizing, it is not required to order datastores in the Journalized Table tab of the model after adding them to CDC.

26.3.5 Configure and Start Oracle GoldenGate Processes

The JKM generates in the LOCAL_TEMP_DIR a folder named after the source and target object groups. This folder contains the following:

- The `Readme.txt` file that contains detailed instructions for configuring and starting the Oracle GoldenGate processes.
- The `src` folder that contains configuration files to upload on the source server, in the Oracle GoldenGate installation directory.
- The `stg` folder that contains configuration files to upload on the staging server, in the Oracle GoldenGate installation directory.

The detailed instructions, customized for your configuration, are provided in the readme file.

These instructions include:

1. Uploading or copying files from the `src` folder to the source server.
2. Uploading or copying files from the `stg` folder to the staging server.
3. Running on the source server the `OBEY` file generated by the JKM for starting the Extract process, using the `ggsci` command line.
4. Generating on the source server definition file using the defgen command line.
5. Copying this definition file to the staging server.
6. If the initial load option is used:
   - Running on the staging server the `OBEY` file generated by the JKM for the initial load, using the `ggsci` command line.
   - Running on the source server the `OBEY` file generated by the JKM for the initial load, using the `ggsci` command line.
7. Finally Running on the staging server the `OBEY` file generated by the JKM for the starting the Replicat processes, using the `ggsci` command line.

See the Oracle GoldenGate documentation on OTN for more information on `OBEY` files, the `ggsci` and `defgen` utilities.

26.3.6 Design Interfaces Using Replicated Data

You can use the data in the replicated data as a source in your integration interfaces. This process is similar to using a source datastore journalized in consistent set mode. See "Using Changed Data: Consistent Set Journalizing" in the the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information.

26.4 Advanced Configuration

This section includes the following advanced configuration topics:

- Initial Load Method
- Tuning Replication Performances
- One Source Multiple Staging Configuration

26.4.1 Initial Load Method

The staging tables contain a replica of the structure and data from the source tables. The Oracle GoldenGate processes capture changes on the source tables and apply them to the target. Yet the staging tables must be initially loaded with the original content of the source tables. You can use the following methods to perform the initial load:

- Using Oracle GoldenGate: A specific GoldenGate process loads the whole content of the source tables into the staging tables.
- Using Oracle Data Integrator: The Generate Interfaces IN option of Oracle Data Integrator's Common Format Designer. This method uses ODI interfaces to transfer the data.
- Using database backup/restore tools to copy data and structures.

26.4.2 Tuning Replication Performances

The following KM options can be used to improve replication performances:

- COMPATIBLE: This Oracle-specific option affects the use of the PURGE key word and the way statistics (using DBMS_STATS or ANALYZE) are collected. Set this value to the database version of your staging server.
- NB_APPLY_PROCESS: Number of Oracle GoldenGate Apply processes created on the staging server.
- TRAIL_FILE_SIZE: Size of the Oracle GoldenGate trail file in Megabytes.

For the NB_APPLY_PROCESS and TRAIL_FILE_SIZE parameters, see the Oracle GoldenGate Documentation on OTN for more information on performance tuning.

26.4.3 One Source Multiple Staging Configuration

It is possible to set up a configuration where changes are captured on a single source and replicated to several staging servers. The example below illustrates how to set this up in a typical configuration.

Replication should source from source server SRC and replicate in both STG1 and STG2 staging servers.

1. Configure CDC for STG1 with the following configuration:
   - SRC_OGG_OBJECT_GROUP = SRC

- SRC_SETUP_OGG_PROCESSES = YES
- STG_OGG_OBJECT_GROUP = STG1
- STG_SETUP_OGG_PROCESSES = YES
- ENABLE_ODI_CDC= YES

2. Start the journal and follow the instructions in the readme to set up the Oracle GoldenGate processes in SRC and STG1.
3. Configure CDC for STG2 with the following configuration:
   - SRC_OGG_OBJECT_GROUP = SRC (Use the same name as for STG1)
   - SRC_SETUP_OGG_PROCESSES = NO (The processes have been set up with STG1)
   - STG_OGG_OBJECT_GROUP = STG2
   - STG_SETUP_OGG_PROCESSES = YES
   - ENABLE_ODI_CDC= YES

Start the journal and follow the instructions in the readme to set up the Oracle GoldenGate processes in SRC and STG2. Note that playing the configuration on SRC again will not recreate a capture process, trail files, or definition files. It will simply create a new Oracle GoldenGate Datapump process to push data to STG2.

27

Oracle Enterprise Service Bus

This chapter describes how to work with Oracle Enterprise Service Bus in Oracle Data Integrator.

This chapter includes the following sections:

27.1 Introduction

Oracle Data Integrator features are designed to work best with Enterprise Service Bus (ESB), including integration interfaces that load a target table from several source tables and handle cross-references.

27.1.1 Concepts

Cross-referencing is the Oracle Fusion Middleware Function, available through its Enterprise Service Bus (ESB) component, and leveraged typically by any loosely coupled integration, which is truly built on the Service Oriented Architecture. It is used to manage the runtime correlation between the various participating applications of the integration.

The cross-referencing feature of Oracle SOA Suite enables you to associate identifiers for equivalent entities created in different applications. For example, you can use cross-references to associate a customer entity created in one application (with native id Cust_100) with an entity for the same customer in another application (with native id CT_001).

Cross-reference (XRef) facilitates mapping of native keys for entities across applications. For example, correlate the same order across different ERP systems.

The implementation of cross-referencing uses an Oracle database schema to store a cross-reference table (called XREF_DATA) that stores information to reference records across systems and data stores.

The optional ability to update or delete source table data after the data is loaded into the target table is also a need in integration. This requires that the bulk integration provides support for either updating some attributes like a status field or purging the source records once they have been successfully processed to the target system.

27.1.2 Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in Table 27–1 for handling ESB cross-references. The KMs use ESB specific features.

Table 27–1 ESB Knowledge Modules

Knowledge Module	Description
LKM SQL to SQL (ESB XREF)	This KM supports cross-references while loading data from a standard ISO source. It supports both Oracle and DB2. The LKM SQL to SQL (ESB XREF) has to be used in conjunction with the IKM SQL Control Append (ESB XREF) in the same interface.
LKM MSSQL to SQL (ESB XREF)	This KM is a version of the LKM SQL to SQL (ESB XREF) optimized for Microsoft SQL Server.
IKM SQL Control Append (ESB XREF)	This KM provides support for cross-references while integrating data to an Oracle, DB2 or Microsoft SQL Server target. It integrates data to the target table in truncate/insert (append) mode, and supports data checks.

27.1.3 Overview of the XREF KM Process

The overall process can be divided into the following three main phases:

Loading Phase (LKM)

During the loading phase, a *Source Primary Key* is created using columns from the source table. This *Source Primary Key* is computed using a user-defined SQL expression that should return a VARCHAR value. This expression is specified in the SRC_PK_EXPRESSION KM option.

For example, for a source Order Line Table (aliased OLINE in the interface) you can use the following expression:

```
TO_CHAR(OLINE.ORDER_ID) || '-' || TO_CHAR(OLINE.LINE_ID)
```

This value will be finally used to populate the cross-reference table.

Integration and Cross-Referencing Phase (IKM)

During the integration phase, a *Common ID* is created for the target table. The value for the *Common ID* is computed from the expression in the XREF_SYS_GUID KM option. This expression can be for example:

- A database sequence (`<SEQUENCE_NAME>. NEXTVAL`)
- A function returning a global unique Id (`SYS_GUID()` for Oracle, `NewID()` for SQL Server)

This *Common ID* is pushed to the target columns of the target table that are marked with the UD1 flag.

Both the *Common ID* and the *Source Primary Key* are pushed to the cross-reference table (XREF_DATA). In addition, the IKM pushes to the cross-reference table a unique *Row Number* value that creates the cross-reference between the *Source Primary Key* and *Common ID*. This *Row Number* value is computed from the XREF_ROWNUMBER_EXPRESSION KM option, which takes typically expressions similar to the *Common ID* to generate a unique identifier.

The same *Common ID* is reused (and not re-computed) if the same source row is used to load several target tables across several interfaces with the Cross-References KMs. This allows the creation of cross-references between a unique source row and different targets rows.

Updating/Deleting Processed Records (LKM)

This optional phase (parameterized by the SRC_UPDATE_DELETE_ACTION KM option) deletes or updates source records based on the successfully processed source records:

- If SRC_UPDATE_DELETE_ACTION takes the `DELETE` value, the source records processed by the interface are deleted.
- If SRC_UPDATE_DELETE_ACTION takes the `UPDATE` value, the source column of the source records processed by the interface is updated with the SQL expression given in the SRC_UPD_EXPRESSION KM option. The name of this source column must be specified in the SRC_UPD_COL KM option.

27.2 Installation and Configuration

Make sure you have read the information in this section before you start using the ESB Knowledge Modules:

- System Requirements and Certifications
- Technology Specific Requirements
- Connectivity Requirements

27.2.1 System Requirements and Certifications

Before performing any installation you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements for the products you are installing.

The list of supported platforms and versions is available on Oracle Technical Network (OTN):

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

27.2.2 Technology Specific Requirements

There are no connectivity requirements for using ESB in Oracle Data Integrator. The requirements for the Oracle Database apply also to ESB. See Chapter 2, "Oracle Database" for more information.

27.2.3 Connectivity Requirements

There are no connectivity requirements for using ESB in Oracle Data Integrator. The requirements for the Oracle Database apply also to ESB. See Chapter 2, "Oracle Database" for more information.

27.3 Working with XREF using the ESB Cross-References KMs

This section consists of the following topics:

- Defining the Topology

27.3.1 Defining the Topology

The steps to create the topology in Oracle Data Integrator, which are specific to projects using ESB Cross-References KMs, are the following:

1. Create the data servers, physical and logical schemas corresponding to the sources and targets.
2. Create a data server for the Oracle technology as described in Section 2.3.1, "Creating an Oracle Data Server".
3. Under this Oracle data server, create a physical and a logical schema called *ESB_XREF* for the schema containing the cross-reference table named *XREF_DATA*. If this table is stored in a data server already declared, you only need to create the schemas.

   See "Creating a Physical Schema" and "Creating a Logical Schema" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information.

27.3.2 Setting up the Project

Import the following KMs into your project, if they are not already in your project:

- IKM SQL Control Append (ESB XREF)
- LKM SQL to SQL (ESB XREF) or LKM MSSQL to SQL (ESB XREF) if using Microsoft SQL Server.

27.3.3 Designing an Interface with the ESB Cross-References KMs

To create an integration interface, which both loads a target table from several source tables and handles cross-references between one of the sources and the target, run the following steps:

1. Create an interface with the source and target datastores which will have the cross-references.
2. Create joins, filters and mappings as usual. Make sure to check the UD1 flag for the column of the target datastore that will be the placeholder for the *Common ID*. Note that you do not need to map this column.
3. In the Flow tab of the interface, select the source set containing the source table to cross-reference, and select the LKM SQL to SQL (ESB XREF) or LKM MSSQL to SQL (ESB XREF) if the source data store is in Microsoft SQL Server.
4. Specify the KM options as follows:
   - SRC_PK_EXPRESSION

     Specify the expression representing the *Source Primary Key* value that you want to store in the XREF table. If the source table has just one column defined as a key, enter the column name (for example SEQ_NO). If the source key has multiple columns, specify the expression to use for deriving the key value. For example, if there are two key columns in the table and you want to store the concatenated value of those columns as your source value in the XREF table enter SEQ_NO | DOC_DATE. This option is mandatory.
   - SRC_UPDATE_DELETE_ACTION

Indicates what action to take on the source records after integrating data into the target. See Table 27–2 for a list of possible values.

Table 27–2 Values of the SRC_UPDATE_DELETE_ACTION

Value	Description
NONE	Specify `NONE` for no action on the source records.
UPDATE	Enter `UPDATE` to update the source records flag according to SRC_UPD_COL and SRC_UPD_EXPRESSION. If you select the `UPDATE` option you also need to specify the following options: SRC_PK_LOGICAL_SCHEMA, SRC_PK_TABLE_NAME, SRC_PK_TABLE_ALIAS, SRC_UPD_COL, and SRC_UPD_EXPRESSION.
DELETE	Enter `DELETE` to delete the source records after the integration. If you select the DELETE option, you also need to specify the following options: SRC_PK_LOGICAL_SCHEMA, SRC_PK_TABLE_NAME, and SRC_PK_TABLE_ALIAS.

5. Select your staging area in the Flow tab of the interface and select the **IKM SQL Control Append (ESB XREF)**.
6. Specify the KM options as follows:
   - XREF_TABLE_NAME - Enter the name of the source table that will be stored in the reference table.
   - XREF_COLUMN_NAME - This is the name of the source primary key that will be stored in the XREF table.
   - XREF_SYS_GUID_EXPRESSION - Expression to be used to computing the *Common ID*. This expression can be for example:
   - a database sequence (`<SEQUENCE_NAME>.NEXTVAL`)
   - a function returning a global unique Id (`SYS_GUID()` for Oracle and `NewID()` for SQL Server)
   - XREF_ROWNUMBER_EXPRESSION - This is the value that is pushed into the *Row Number* column of the XREF_DATA table. Use the default value of GUID unless you have the need to change it to a sequence.
   - FLOW_CONTROL - Set to `YES` in order to be able to use the CKM Oracle.

Note: If the target table doesn't have any placeholder for the *Common ID* and you are for example planning to populate the source identifier in one of the target columns, you must use the standard mapping rules of ODI to indicate which source identifier to populate in which column.

If the target column that you want to load with the *Common ID* is a unique key of the target table, it needs to be mapped. You must put a dummy mapping on that column. At runtime, this dummy mapping will be overwritten with the generated common identifier by the integration knowledge module. Make sure to flag this target column with UD1.

27.4 Knowledge Module Options Reference

This section lists the KM options for the following Knowledge Modules:

- LKM SQL to SQL (ESB XREF)
- LKM MSSQL to SQL (ESB XREF)
- IKM SQL Control Append (ESB XREF)

Table 27–3 LKM SQL to SQL (ESB XREF)

Option	Values	Mandatory	Description
SRC_UPDATE_DELETE_ACTION	NONE | UPDATE | DELETE	Yes	Indicates what action to take on source records after integrating data into the target. See Table 27–2 for a list of valid values for this option.
SRC_PK_EXPRESSION	Concatenating expression	Yes	Expression that concatenates values from the PK to have them fit in a single large varchar column. For example: for the source Orderline Table (aliased OLINE in the interface) you can use expression: `TO_CHAR(OLINE.ORDER_ID) || '-' || TO_CHAR(OLINE.LINE_ID)`
SRC_PK_LOGICAL_SCHEMA	Name of source table's logical schema	No	Indicates the source table's logical schema. The source table is the one from which we want to delete or update records after processing them. This logical schema is used to resolve the actual physical schema at runtime depending on the Context. For example: `ORDER_BOOKING`. This option is required only when SRC_UPDATE_DELETE_ACTION is set to `UPDATE` or `DELETE`.
SRC_PK_TABLE_NAME	Source table name, default is MY_TABLE	No	Indicate the source table name of which we want to delete records after processing them. For example: `ORDERS` This option is required only when SRC_UPDATE_DELETE_ACTION is set to `UPDATE` or `DELETE`.
SRC_PK_TABLE_ALIAS	Source table alias, default is MY_ALIAS	No	Indicate the source table's alias within this interface. The source table is the one from which we want to delete or update records after processing them. For example: `ORD`. This option is required only when SRC_UPDATE_DELETE_ACTION is set to `UPDATE` or `DELETE`.

Table 27–3 (Cont.) LKM SQL to SQL (ESB XREF)

Option	Values	Mandatory	Description
SRC_UPD_COL	Aliased source column name	No	Aliased source column name that holds the update flag indicator. The value of this column will be updated after integration when SRC_UPDATE_DELETE_ACTION is set to `UPDATE` with the expression literal SRC_UPD_EXPRESSION. The alias used for the column should match the one defined for the source table. For example: `ORD.LOADED_FLAG`. This option is required only when SRC_UPDATE_DELETE_ACTION is set to `UPDATE`.
SRC_UPD_EXPRESSION	Literal or expression	No	Literal or expression used to update the SRC_UPD_COL. This value will be used to update this column after integration when SRC_UPDATE_DELETE_ACTION is set to `UPDATE`. For example: `RECORDS PROCESSED`. This option is required only when SRC_UPDATE_DELETE_ACTION is set to `UPDATE`.
DELETE_TEMPORARY_OBJECTS	Yes | No	Yes	Set this option to `NO` if you wish to retain temporary objects (files and scripts) after integration. Useful for debugging.

LKM MSSQL to SQL (ESB XREF)

See Table 27–3 for details on the LKM MSSQL to SQL (ESB XREF) options.

Table 27–4 IKM SQL Control Append (ESB XREF)

Option	Values	Mandatory	Description
INSERT	Yes | No	Yes	Automatically attempts to insert data into the Target Datastore of the Interface.
COMMIT	Yes | No	Yes	Commit all data inserted in the target datastore.
FLOW_CONTROL	Yes | No	Yes	Check this option if you wish to perform flow control.
RECYCLE_ERRORS	Yes | No	Yes	Check this option to recycle data rejected from a previous control.
STATIC_CONTROL	Yes | No	Yes	Check this option to control the target table after having inserted or updated target data.
TRUNCATE	Yes | No	Yes	Check this option if you wish to truncate the target datastore.
DELETE_ALL	Yes | No	Yes	Check this option if you wish to delete all the rows of the target datastore.
CREATE_TARG_TABLE	Yes | No	Yes	Check this option if you wish to create the target table.
DELETE_TEMPORARY_OBJECTS	Yes | No	Yes	Set this option to `NO` if you wish to retain temporary objects (tables, files and scripts) after integration. Useful for debugging.
XREF_TABLE_NAME	XREF table name	Yes	Table Name to use in the XREF table. Example: `ORDERS`

Table 27–4 (Cont.) IKM SQL Control Append (ESB XREF)

Option	Values	Mandatory	Description
XREF_COLUMN_NAME	Column name	Yes	Primary key column name to use as a literal in the XREF table
XREF_SYS_GUID_ EXPRESSION	SYS_GUID()	Yes	Enter the expression used to populate the common ID for the XREF table (column name "VALUE"). Valid examples are: `SYS_GUID()`, `MY_SEQUENCE.NEXTVAL`, and so forth.
XREF_ROWNUMBER_ EXPRESSION	SYS_GUID()	Yes	Enter the expression used to populate the row_number for the XREF table. For example for Oracle: `SYS_GUID(),MY_ SEQUENCE.NEXTVAL` and so forth.

A

Oracle Data Integrator Driver for LDAP Reference

This appendix describes how to work with the Oracle Data Integrator driver for LDAP.

This appendix includes the following sections:

A.1 Introduction to Oracle Data Integrator Driver for LDAP

With *Oracle Data Integrator Driver for LDAP (LDAP driver)* Oracle Data Integrator is able to manipulate complex LDAP trees using standard SQL queries.

The LDAP driver supports:

- Manipulation of LDAP entries, their object classes and attributes
- Standard SQL (Structured Query Language) Syntax
- Correlated subqueries, inner and outer joins
- ORDER BY and GROUP BY
- COUNT, SUM, MIN, MAX, AVG and other functions
- All Standard SQL functions
- Referential Integrity (foreign keys)
- Persisting modifications into directories

A.2 LDAP Processing Overview

The LDAP driver works in the following way:

1. The driver loads (upon connection) the LDAP structure and data into a relational schema, using a LDAP to Relational Mapping.
2. The user works on the relational schema, manipulating data through regular SQL statements. Any changes performed in the relational schema data (insert/update) are immediately impacted by the driver in the LDAP data.

A.2.1 LDAP to Relational Mapping

The *LDAP to Relational Mapping* is a complex but automated process that is used to generate a relational structure. As LDAP servers do not provide metadata information in a standard way, this mapping is performed using data introspection from the LDAP tree. Therefore, automatic mapping is carried out on the contents of the LDAP tree used as a source for this process.

This section contains the following topics:

- General Principle
- Grouping Factor
- Mapping Exceptions
- Reference LDAP Tree

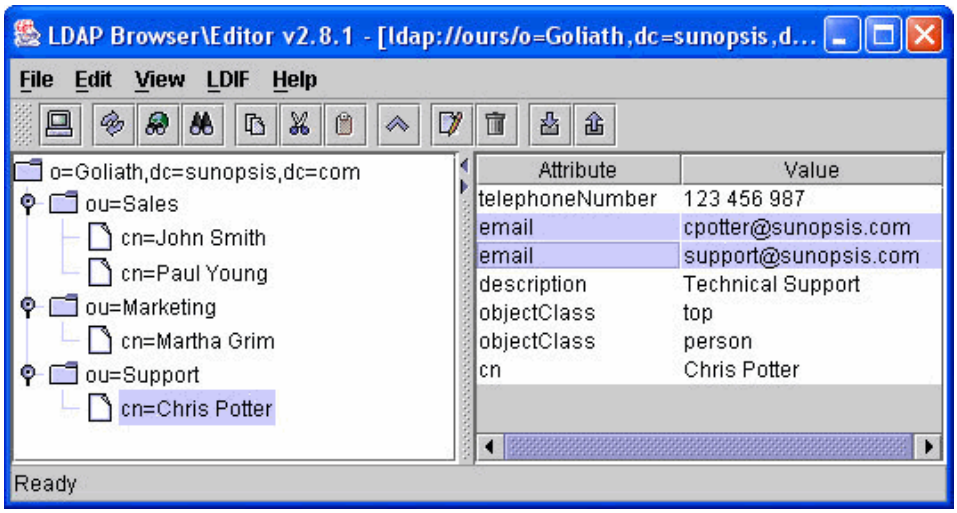
A.2.1.1 General Principle

The LDAP driver maps LDAP elements to a relational schema in the following way:

- Each LDAP class or combination of classes is mapped to a table. Each entry from the LDAP tree is mapped to a record in the table.
- Each attribute of the class instances is mapped to a column.
- Hierarchical relationships between entries are mapped using foreign keys. A table representing a hierarchical level is created with a primary key called `<tablename>PK`. Records reference their parent tables through a `<parent_level_tablename>FK column`. The root of the LDAP tree structure is mapped to a table called `ROOT` containing a `ROOTPK` column in a unique record.
- Attributes with multiple values for an entry (for example, a *Person* entry with several *email* attributes) are mapped as sub-tables called `<parent_tablename><attribute_name>`. Each sub-table contains a `<parent_tablename>FK column` linking it to the parent table.

Figure A–1 shows an LDAP tree with OrganizationalUnit entries linking to Person instances. In this case, certain Person entries have multiple email addresses.

Figure A–1 LDAP Tree Example

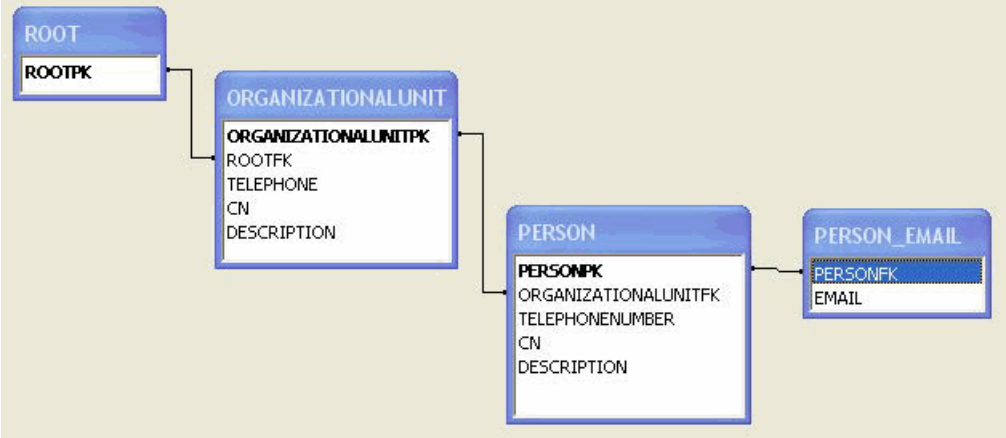


This LDAP tree will be mapped into the following relational structure:

- The ROOT table represents the root of the hierarchy and contains one ROOTPK column.
- The ORGANIZATIONALUNIT table represents different *organizationalUnit* instances of the tree. It contains the ORGANIZATIONALUNITPK primary key column and the attributes of the *organizationalUnit* instances (*cn*, *telephoneNumber*, etc.). It is linked to the ROOT table by the ROOTFK foreign key column.
- The PERSON table represents the instances of the *person* class. It contains the PERSONPK primary key column and the ORGANIZATIONALUNITFK linking it to the ORGANIZATIONALUNIT table and the attributes of PERSON instances, (*telephoneNumber*, *description*, *cn*).
- The *email* attribute appears as a PERSON_EMAIL table containing the EMAIL column and a PERSONFK linking a list of email attributes to a PERSON record.

Figure A–2 shows the resulting relational structure.

Figure A–2 Relational Structure mapped from the LDAP Tree Example shown in Figure A–1



A.2.1.2 Grouping Factor

In LDAP directories, class entries are often specified by inheriting attributes from multiple class definitions. In the relational mapping procedure, the LDAP driver translates this fact by combining each combination of classes in an LDAP entry to generate a new table.

For example, some entries of the *Person* class may also be instances of either of the *Manager* or *BoardMember* classes (or both). In this case, the mapping procedure would generate a PERSON table (for the instances of Person) but also MANAGER_PERSON, BOARDMEMBER_PERSON, BOARDMEMBER_MANAGER_PERSON and so forth, tables depending on the combination of classes existing in the LDAP tree.

In order to avoid unnecessary multiplication of generated tables, it is possible to parameterize this behavior. The *Grouping Factor* parameter allows this by defining the number of divergent classes below which the instances remain grouped together in the same table. This resulting table contains flag columns named *IS_<classname>*, whose values determine the class subset to which the instance belongs. For example, if IS_<classname> is set to 1, then the instance represented by the record belongs to <classname>.

The behavior where one table is created for each combination of classes corresponds to a Grouping Factor equal to zero. With a grouping factor equal to one, instances with only one divergent class remain in the same table.

In our example, with a Grouping Factor higher than or equal to 2, all company person instances (including *Person*, *Manager* and *BoardMember* class instances) are grouped in the `PERSON` table. The `IS_MANAGER` and `IS_BOARDMEMBER` columns enable the determination of PERSON records that are also in the *Manager* and/or *BoardMember* classes.

A.2.1.3 Mapping Exceptions

This section details some specific situations of the mapping process.

- **Table name length limits and collisions**: In certain cases, name-length restrictions may result in possible object name collisions. The LDAP driver avoids such situations by automatically generating 3 digit suffixes to the object name.
- **Key column**: It is possible to have the driver automatically create an additional `SNPSLDAPKEY` column containing the Relative Distinguished Name (RDN) that can be used as identifier for the current record (original LDAP class instance). This is done by setting the `key_column` URL property to true. This `SNPSLDAPKEY` column must be loaded if performing DML commands that update the LDAP tree contents. Note that this column is created only in tables that originate from LDAP instances. Tables that correspond to multiple valued instance attributes will *not* be created with these columns.
- **Case sensitivity**: This is set by the `case_sens` URL property that makes the RDBMS and LDAP servers to enforce case-sensitivity.
- **Special characters**: It is possible in LDAP to have non-alphanumeric characters into attribute or class names. These characters are converted to underscores ("_") during the mapping. Exception: If non alphanumeric, the first character is converted to "`x`".
- **SQL Reversed Keywords**: Generated tables and columns with names that match SQL keywords are automatically renamed (an underscore is added after their name) in the relational structure to avoid naming conflicts between table/column names and SQL keywords. For example, a class named `SELECT` will be mapped to a table named `SELECT_`.

A.2.1.4 Reference LDAP Tree

As LDAP servers do not provide metadata information in a standard way, the *LDAP to Relational Mapping* process is performed by default using data introspection from the LDAP tree.

With the LDAP driver it is also possible to use a *Reference LDAP Tree* for the *LDAP to Relational Mapping* process instead of using the LDAP tree that contains the actual data.

This Reference LDAP Tree is configured using the `lm_props` property of the driver URL. This property specifies a `.properties` file that contains the connection information to a LDAP tree whose hierarchical structure rigorously reflects that of the operational LDAP tree but without the accompanying data volume.

This technique reveals certain advantages:

- The Reference LDAP Tree can be maintained by the directory administrator as a stable definition of the operational LDAP tree.

- The Reference LDAP Tree contains few instances that make up the skeleton of the real LDAP tree, and the LDAP to Relational Mapping process runs faster on this small reference tree. This is particularly important for large operational LDAP directories, and will result in reduced processing time and resources for running the procedure.

The use of this technique, however, imposes a certain number of constraints in the design of the precise structure of the Reference LDAP Tree:

- All optional LDAP instance attributes must be instantiated in the reference entries. Even if these attributes are absent in the operational LDAP directory entries, they must be declared in the Reference LDAP Tree if they are to be used at a later time.
- Any multiple valued attributes that exist in the operational LDAP directory must be instantiated as such in the Reference LDAP Tree. For example, if any *Person* instance in the operational LDAP directory possesses two *telephoneNumber* attributes, then the generic *Person* class *must* instantiate at least two *telephoneNumber* attributes in the Reference LDAP Tree.

> **Note:** These issues have a direct impact on the generated relational structure by forcing the creation of additional tables and columns to map multiple attribute fields and must be taken into consideration when designing the Reference LDAP Tree.

A.2.2 Managing Relational Schemas

This section contains the following topics:

- Relational Schema Storage
- Accessing Data in the Relational Structure

A.2.2.1 Relational Schema Storage

The relational structure resulting from the LDAP to Relational mapping may be managed by *virtual mapping* or stored in an *external database*.

The *virtual mapping* stores the relational structure in the run-time agent's memory and requires no other component. The relational structure is transparently mapped by the driver to the LDAP tree structure. SQL commands and functions that are available for the LDAP driver are listed in the SQL Syntax.

> **Note:** The virtual mapping may require a large amount of memory for large LDAP tree structures.

The *external database* may be any relational database management system. The driver connects through JDBC to this engine and uses it to store the relational schema. This method provides the following benefits:

- Processing and storage capabilities of the selected external database engine.
- Acccess to the specific SQL statements, procedures, and functions of the external database engine.
- Flexible persistence of the relational structure. This schema content may persist after the connection to the LDAP driver is closed.

A.2.2.2 Accessing Data in the Relational Structure

DML operations on tables in the relational are executed with standard SQL statements.

Modifications made to the relational data are propagated to the directory depending on the selected storage :

- In the case where the *virtual mapping* is used, all insert, update, and delete requests are automatically propagated to the original LDAP server in an autocommit mode. No explicit COMMIT or ROLLBACK statements will have any impact on the Oracle Data Integrator driver for LDAP.
- In the case where the *external database* is used to store the relational structure, all types of DML statements may be used with the driver. However, it is important to know that no modifications will be propagated to the original LDAP server.

A.3 Installation and Configuration

The Oracle Data Integrator driver for LDAP is automatically installed during the Oracle Data Integrator installation. The following topics cover advanced configuration topics and reference information.

This section contains the following topics:

- Driver Configuration
- Using Property Bundles
- Table Aliases Configuration

Note: You must add the libraries and drivers required to connect the LDAP directory using JNDI to the Oracle Data Integrator classpath.

Note: If using an external database engine you must also make sure that the JDBC driver used to connect to the external database and the `.properties` file are in the classpath.

A.3.1 Driver Configuration

This section details the driver configuration.

- The driver name is: `com.sunopsis.ldap.jdbc.driver.SnpsLdapDriver`
- The driver supports two URL formats:
  - `jdbc:snps:ldap?<property=value>[&...]`
  - `jdbc:snps:ldap2?<property=value>[&...]`

  The first URL requires the LDAP directory password to be encoded. The second URL allows you to give the LDAP directory password without encoding it.

  Note: It is recommended to use the first URL to secure the LDAP directory password.

The properties for the URL are detailed in Table A–1.

Table A–1 Driver Properties

Property	Mandatory	Type	Default	Description
db_props or dp	No	string (file location)	Empty string	Name of a `.properties` file containing the external database connection configuration. See Section A.3.2.2, "External Database Connection Configuration" for the details of this file content. **Note:** This property should contain the name of the *.properties* file without the file extension. **Note:** This `.properties` file must be in the run-time agent classpath. **Note:** You can specify the external database connection configuration using all the `db_` properties listed below in this table.
ldap_props or lp	No	string (file location)	N/A	Name of a `.properties` file containing the directory connection configuration. See Section A.3.2.1, "LDAP Directory Connection Configuration" for the details of this file content. **Note:** This property should contain the name of the *.properties* file without the file extension. **Note:** This `.properties` file must be in the run-time agent classpath. **Note:** You can specify the LDAP directory connection configuration using all the `ldap_` properties listed below in this table.
lm_props or lm	No	string (file location)	N/A	Name of a `.properties` file containing the directory connection configuration for the *Reference LDAP Tree*. See Section A.3.2.1, "LDAP Directory Connection Configuration" for the details of this file content, and Section A.2.1.4, "Reference LDAP Tree" for an explanation of the reference tree. **Note:** This property should contain the name of the *.properties* file without the file extension. **Note:** This `.properties` file must be in the run-time agent classpath. **Note:** You can specify the reference LDAP directory connection configuration using all the `lm_` properties listed below in this table.
case_sens or cs	No	boolean (true | false)	false	Enable / disable case sensitive mode for both LDAP- and RDBMS-managed objects.
alias_ bundle or ab	No	string (file location)	Empty string	Name of a `.properties` file containing the list of aliases for the LDAP to Relational Mapping. See Section A.3.3, "Table Aliases Configuration" for more information. **Note:** This property should contain the name of the *.properties* file without the file extension. **Note:** This `.properties` file must be in the run-time agent classpath.
alias_ bundle_ encoding or abe	No	string (encoding code)	Default encoding	Alias bundle file encoding. This encoding is used while reading and overwriting the alias_bundle file. If it is not defined then the default encoding would be used. You will find a list of supported encoding at the following URL: `http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html`.

Table A–1 (Cont.) Driver Properties

Property	Mandatory	Type	Default	Description
grouping_ factor or gf	No	alphanumeric	2	Determines how many object classes will be grouped together to set up a single relational table mapping. See Section A.2.1.2, "Grouping Factor" for more information.
key_column or kc	No	boolean (true | false)	false	If set to true, a technical column called `SNPSLDAPKEY` is created to store the Relative Distinguished Name (RDN) for each LDAP entry. See Section A.2.1.3, "Mapping Exceptions" for more information.
numeric_ ids or ni	No	boolean (true | false)	false	If set to true, all internal Primary and Foreign Keys are of NUMERIC type. Otherwise, they are of the VARCHAR type.
id_length or il	No	Alphanumeri c	10 / 30	The length of the internal Primary and Foreign Key columns. The default is 10 for NUMERIC column types and 30 for VARCHAR column types.
table_prefix or tp	No	string	N/A	Prefix added to relational tables of the current connection.
log_file or lf	No	string (file location)	N/A	Trace log file name. If the log file name is not set the trace data is displayed on the standard output. The presence of this property triggers trace logging for a particular relational schema.

Table A–1 (Cont.) Driver Properties

Property	Mandatory	Type	Default	Description
log_level or ll	No	alphanumeric	1	Log level. This property is ignored if log_file is not specified. The log level can is a bit mask that can be specified either in hexadecimal or decimal value. Log Level Values: ■ 0x1 (1): General information (important) ■ 0x2 (2): General information (detailed) ■ 0x4 (4): SQL statements ■ 0x8 (8): LDAP-Relational mapping information ■ 0x10 (16): LDAP-Relational mapping validation & renaming information (Table and columns name modifications, etc) ■ 0x20 (32): Display the LDAP model parsed and the corresponding relational model. ■ 0x40 (64): Display the table creation statements. ■ 0x80 (128): Display data insert statements. ■ 0x100 (256): Grouping information (important) ■ 0x200 (512): Grouping information (detailed) ■ 0x400 (1024): Display details on the relational model building ■ 0x800 (2048): Display the elements read from the LDAP tree ■ 0x1000 (4096): Display SQL statements causing changes into the LDAP tree Examples: ■ Important and detailed general information: log_level=3 (1+2) ■ Trace native SQL commands and important internal events: log_level=5 (1+4) ■ Trace relational mapping calculation and validation: log_level=24 (16+8) ■ Trace all events: log_level=8191 (1+2+ ... + 2048 + 4096)
ldap_auth	No	string	simple	LDAP Directory authentication method. See the `auth` property in Section A–2, " LDAP Directory Connection Properties"
ldap_url	Yes	string	N/A	LDAP Directory URL. See the `url` property in Section A–2, " LDAP Directory Connection Properties"
ldap_user	No	string	Empty string	LDAP Directory user name. See the `user` property in Section A–2, " LDAP Directory Connection Properties"
ldap_password	No	string	Empty string	LDAP Directory user password. See the `password` property in Section A–2, " LDAP Directory Connection Properties"
lldap_basedn	No	string	N/A	LDAP Directory basedn. See the `basedn` property in Section A–2, " LDAP Directory Connection Properties"
lm_auth	No	string	simple	Reference LDAP authentication method. See the `auth` property in Section A–2, " LDAP Directory Connection Properties"
lm_url	Yes	string	N/A	Reference LDAP URL. See the `url` property in Section A–2, " LDAP Directory Connection Properties"

Table A–1 (Cont.) Driver Properties

Property	Mandatory	Type	Default	Description
lm_user	No	string	Empty string	Reference LDAP Directory user name. See the `user` property in Section A–2, " LDAP Directory Connection Properties"
lm_ password	No	string	Empty string	Reference LDAP Directory user password. See the `password` property in Section A–2, " LDAP Directory Connection Properties"
lm_basedn	No	string	N/A	Reference LDAP Directory basedn. See the `basedn` property in Section A–2, " LDAP Directory Connection Properties"
db_driver	Yes	string	N/A	External Database JDBC Driver. See the `driver` property in Section A–3, " External Database Connection Properties"
db_url	Yes	string	N/A	External Database JDBC URL. See the `url` property in Section A–3, " External Database Connection Properties"
db_user	No	string	Empty string	External Database user. See the `user` property in Section A–3, " External Database Connection Properties"
db_ password	No	string	Empty string	External Database password. See the `password` property in Section A–3, " External Database Connection Properties"
db_schema	No	string	Empty string	External Database schema. See the `schema` property in Section A–3, " External Database Connection Properties"
db_catalog	No	string	Empty string	External Database catalog. See the `catalog` property in Section A–3, " External Database Connection Properties"
db_drop_ on_ disconnect or db_dod	No	boolean (Y | N)	true	Drop tables on disconnect on the external database. See the `drop_on_disconnect` property in Section A–3, " External Database Connection Properties"
db_load_ mode or db_lm	No	string	ci	Loading method for the external database. See the `load_ mode` property in Section A–3, " External Database Connection Properties"

URL Examples

The following section lists URL examples:

- `jdbc:snps:ldap?lp=ldap_mir&ldap_basedn=o=tests&gf=10&lf=`

  Connects to the LDAP directory specified in the ldap_mir `.properties` file, overriding the basedn property of the ldap bundle and using a grouping factor of 10. General information (important) is sent to the standard output.

- `jdbc:snps:ldap?lp=ldap_ ours&lm=generic&ab=c:/tmp/aliases.txt&gf=10&kc=true`

  Connects to the LDAP directory using the ldap_ours `.properties` file; a generic Directory tree for relational model creation is signaled by the lm property; an alias bundle file is used for the creation of the relational structure; a maximum grouping factor of 10 is used; key column creation is enabled for the SNPSLDAPKEY field to allow updates requests in the relational model.

- `jdbc:snps:ldap?lp=ldap_mir&dp=mysql_mir_ldap&ldap_ basedn=dc=tests&lm=ldap_mir&lm_ basedn=dc=model&ab=d:/temp/mapldap.txt&`

  Connects to the LDAP directory using the ldap_mir `.properties` file; overriding ldap basedn property; using the "dc=model" subtree of the same directory to

perform mapping; using an alias bundle; overriding the lm database property (load mode); specifying a grouping factor of 0 to indicate no grouping (grouping disabled); Full trace logging is activated.

- Connects to a LDAP directory on the hydraroid machine. The LDAP server connection information - url, base dn, user and password - is specified in the URL using the ldap_xxx properties.

```
jdbc:snps:ldap?ldap_url=ldap://hydraroid:389/dc=localhost,dc=localdomain&ldap_
password=KPLEKFMJKCLFJMDFDDGPGPDB&ldap_user=cn=orcladmin&ldap_
basedn=ou=applications
```

A.3.2 Using Property Bundles

As described in Table A–1, the LDAP driver URL can reference property bundles (`.properties` files) that contain the following information:

- LDAP Connection information for the Directory
- LDAP Connection information for the Reference Directory
- External Database connection information

The values stored the bundle files (`.properties` files) can be also passed in the URL:

- **LDAP directory connection:** The properties start with `ldap_`. For example, `ldap_basedn`.
- **LDAP Reference Directory connection:** The properties start with `lm_`. For example, `lm_basedn`.
- **External Database connection:** The properties start with `db_`. For example, `db_url`.

When using property bundle files, you must make sure that the property bundle is present in the Oracle Data Integrator classpath. Typically, you should install this bundle in the drivers directories.

You should also make sure that the URL refers to the property bundle without specifying the file extension. For example, if you have in your classpath the `prod_directory.properties` file, you should refer to this file as follows: `lp=prod_directory`.

> **Note:** It is important to understand that the LDAP driver loads external property bundle files once only at runtime startup. If errors occur in these files, it is advisable to exit Oracle Data Integrator and then reload it before re-testing.

A.3.2.1 LDAP Directory Connection Configuration

The Oracle Data Integrator driver for LDAP uses the following properties to connect to a directory server that contains the LDAP data or the *Reference LDAP Tree*. These properties can be provided either in a property bundle file or on the driver URL.

The properties for configuring a directory connection are detailed in Table A–2.

Table A–2 LDAP Directory Connection Properties

Property	Mandatory	Type	Default	Description
auth	No	string	simple	The authentication method

Table A–2 (Cont.) LDAP Directory Connection Properties

Property	Mandatory	Type	Default	Description
url	Yes	string	N/A	URL to connect to the directory. It is an LDAP URL. **Note:** This driver supports the LDAPS (LDAP over SSL) protocol. The LDAPS URL must start with ldaps://. To connect a server using LDAPS, you must manually install the certificate in the java machine. See the *keytool* program provided with the JVM for more information.
user	No	string	Empty string	The LDAP server user-login name. Mandatory only if "auth" is set. **Note:** If user and password properties are provided to create the connection with the JDBC Driver for LDAP, then they are used to connect the LDAP directory.
password	No	string	Empty string	LDAP server user-login password. Mandatory only if "auth" is set. **Note:** The password needs to be encrypted, unless the 'jdbc:snps:ldap2' URL syntax. **Note:** To encrypt the password, use the `encode.bat` command. See the *Oracle Fusion Middleware Installation Guide for Oracle Data Integrator* for more information.
basedn	No	string	N/A	The base dn with which you wish to connect to the LDAP tree. The base dn is the top level of the LDAP directory tree. If it not specified, the base dn specified in the LDAP URL is used.

The following is an example of an LDAP properties file content:

```
url=ldap://ours:389
user=cn=Directory Manager
password=ENCODED_PASSWORD
basedn=dc=oracle,dc=com
```

A.3.2.2 External Database Connection Configuration

The Oracle Data Integrator driver for LDAP may store the relational structure mapping of the LDAP tree in an external database engine.

Note: The list of technologies that support external storage is available on Oracle Technical Network (OTN) :

http://www.oracle.com/technology/products/oracle-data-integrator/index.html.

The properties for configuring an external database connection are detailed in Table A–3.

Table A–3 External Database Connection Properties

Property	Mandatory	Type	Default	Description
driver	Yes	string	N/A	JDBC driver name
url	Yes	string	N/A	JDBC URL
user	No	string	Empty string	Login used to connect the database

Table A–3 (Cont.) External Database Connection Properties

Property	Mandatory	Type	Default	Description
password	No	string	Empty string	Encrypted database user password. **Note:** To encrypt the password, use the `encode.bat` command. See the *Oracle Fusion Middleware Installation Guide for Oracle Data Integrator* for more information.
schema	No	string	Empty string	Database schema storing the LDAP Tree. This property should not be used for Microsoft SQLServer, and the catalog property should be used instead.
catalog	No	string	Empty string	Database catalog storing the LDAP Tree. For Microsoft SQL Server only. This property should not be used simultaneously with the schema property.
drop_on_disconnect or dod	No	boolean (true | false)	true	If true, drop the tables from the database at disconnection time. If set to false the tables are preserved in the database.
load_mode or lm	No	string	ci	The loading method. Values may be: ■ n (none): the model and table mappings are created in memory only. ■ dci (drop_create_insert): drop all tables that may cause name conflicts then create tables and load the LDAP tree into the relational model. ■ ci(create_insert): Create the relational tables and throw an exception for existing tables, then load the LDAP tree into the relational model.

The following is an example of an external database `.properties` file to connect to an external Oracle database:

```
driver=oracle.jdbc.OracleDriver
url=jdbc:oracle:thin:@hydraro:1521:SNPTST1
user=LDAP_T_1
password=ENCODED_PASSWORD
schema=LDAP_T_1
```

A.3.3 Table Aliases Configuration

The LDAP driver allows a certain flexibility in the definition of the model table names in Oracle Data Integrator by the use of table aliases. This is particularly useful when the algorithm used to navigate the LDAP tree generates long composite names from the LDAP object class hierarchy. To avoid issues related to RDBMS-specific object name-length constraints, the LDAP driver can set up and use aliases.

> **Note:** It is also possible to change the default "Maximum Table Name Length" and "Maximum Column Name Length" values on the Others tab of the Technology Editor in the Physical Architecture accordion.

To create a table alias file:

1. In the LDAP Driver Data Server URL, include and set the alias_bundle (ab) property that indicates the name of the alias text file, for example:

   ```
   jdbc:snps:ldap?.....&ab=..../tmp/aliases.txt&....
   ```

The alias file is created by the driver at connection time when the `alias_bundle` property is specified. Typically, a user connects initially through the LDAP driver which creates this file containing a list of potential table names to be created by the reverse-engineering operation.

2. Test the connection to the LDAP data server.
3. Verify the that the text file has been created and has the expected structure. The list consists of `<original table name > = <desired alias name>` values. Example A–1 shows an extract of an alias file after the user has provided shortened names. See step 4 for more information.

Example A–1 Alias File

```
INETORGPERSON_ORGANIZATIONALPERSON_PERSON_BISOBJECT_MAIL = PERSONMAIL
ORGANIZATIONALUNIT_RFC822MAILMEMBER = ORG_228MAIL
INETORGPERSON_ORGANIZATIONALPERSON_PERSON = ORG_PERSON
ORGANIZATIONALUNIT_MEMBER = ORG_UN_MEMBER
ORGANIZATIONALUNIT = ORG_UNIT
ROOT = ROOT
....
```

4. In the alias text file, add short text value aliases to replace the originally derived composite names and save the file.
5. Reconnect to the same LDAP data server. The relational schema is created and this time the aliases will be used for defining relational table names.
6. Now reverse-engineer the LDAP directory as described in Section 24.5.2, "Reverse-Engineering an LDAP Model". Oracle Data Integrator will create datastores with the table names defined as aliases in the alias file.

Note: If any modifications have been applied to the object class structure or attribute sets of the LDAP directory, the driver will rewrite this file while including the new or modified entries to the table name list.

A.4 SQL Syntax

The SQL statements described in Section A.4.1, "SQL Statements" are available when using the Oracle Data Integrator driver for LDAP. They enable the management of relational data structure and data through standard SQL Syntax.

Note:

- If you are using an external database you may use its proprietary query engine syntax in place of the following commands.
- The LDAP driver works uniquely in auto commit mode. No explicit transaction management with COMMIT or ROLLBACK commands is permitted.
- When using an external database to store LDAP tree data, DDL statements may only be carried out on temporary tables.

Table A–4 summarizes the recommendations to apply when performing the listed DML operations on specific key fields.

Table A–4 DML Opertaions on Key Fields

Type of Column	Insert	Update	Delete
Foreign Key	Pay attention to master table referential constraints and ordered table populate operations.	Not permitted	Pay attention to master table referential constraints and ordered delete requests.
Primary Key	Pay attention to slave table referential constraints and ordered table populate operations.	Not permitted	Pay attention to slave table referential constraints and ordered delete requests
IS_xxx	Pay attention to associating the correct flag value to the original object class.	Not permitted	OK
Key_Column	Pay attention to setting the RDN value in the correct LDAP syntax.	Not permitted	OK

A.4.1 SQL Statements

Any number of commands may be combined. The semicolon (;) may be used to separate each command but is not necessary.

A.4.1.1 DISCONNECT

```
DISCONNECT
```

Closes this connection.

Remarks

- It is not required to call this command when using the JDBC interface: it is called automatically when the connection is closed.
- After disconnecting, it is not possible to execute other queries with this connection.

A.4.1.2 INSERT INTO

Insert one or more new rows of data into a table.

```
INSERT INTO <table_name> [ ( <column_name> [,...] ) ]
      { VALUES (<expression> [,...]) | <SELECT Statement> }
```

A.4.1.3 SELECT

Retrieves information from one or more tables in the schema.

```
SELECT [DISTINCT] { <select_expression> | <table_name>.* | * } [, ... ]
    [ INTO <new_table> ]
      FROM <table_list>
    [ WHERE <expression> ]
    [ GROUP BY <expression> [, ...] ]
    [ ORDER BY <order_expression> [, ...] ]
    [ { UNION [ALL] | {MINUS|EXCEPT} | INTERSECT } <select_statement>
 ]
<table_list> ::=
<table_name> [ { INNER | LEFT [OUTER] } JOIN <table_name> ON <expression> ]
   [, ...]

<select_expression> ::=
```

```
{ <expression> | COUNT(*) | {COUNT | MIN | MAX | SUM | AVG}
  (<expression>) <column_alias>}

<order_expression> ::=
{ <column_number> | <column_alias> | <select_expression> } [ ASC | DESC ]
```

A.4.1.4 UPDATE

Modifies data of a table in the database.

```
UPDATE table SET column = <expression> [, ...] [WHERE <expression>]
```

A.4.1.5 Expressions, Condition & values

```
<expression> ::=
[NOT] <condition> [ { OR | AND } <condition>
]
<condition> ::=
{ <value> [ || <value> ]
| <value> { = | < | <= | > | >= | <> | != | IS [NOT] } <value>
| EXISTS(<select_statement>)
| <value> BETWEEN <value> AND <value>
| <value> [NOT] IN ( {<value> [, ...] | selectStatement } )
| <value> [NOT] LIKE <value> [ESCAPE] value }

<value> ::=
[ + | - ] { term [ { + | - | * | / } term ]
| ( condition )
| function ( [parameter] [,...] )
| selectStatement giving one value

<term> ::=
{ 'string' | number | floatingpoint | [table.]column | TRUE | FALSE | NULL }
```

<string> ::=

- Starts and ends with a single '. In a string started with ' use '' to create a '.
- LIKE uses '%' to match any (including 0) number of characters, and '_' to match exactly one character. To search for '%' itself, '\%' must be used, for '_' use '_'; or any other escaping character may be set using the ESCAPE clause.

<name> ::=

- A name starts with a letter and is followed by any number of letters or digits. Lowercase is changed to uppercase except for strings and quoted identifiers. Names are not case-sensitive.
- Quoted identifiers can be used as names (for example for tables or columns). Quoted identifiers start and end with ". In a quoted identifier use "" to create a ". With quoted identifiers it is possible to create mixed case table and column names. Example: CREATE TABLE "Address" ("Nr" INTEGER,"Name" VARCHAR); SELECT * FROM "Address". Quoted identifiers are not strings.

<values> ::=

- A 'date' value starts and ends with ', the format is yyyy-mm-dd (see java.sql.Date).
- A 'time' value starts and ends with ', the format is hh:mm:ss (see java.sql.Time).
- Binary data starts and ends with ', the format is hexadecimal. '0004ff' for example is 3 bytes, first 0, second 4 and last 255 (0xff).

A.4.2 SQL FUNCTIONS

Table A–5 describes the numeric functions.

Table A–5 Numeric Functions

Function	Description
ABS(d)	returns the absolute value of a double value
ACOS(d)	returns the arc cosine of an angle
ASIN(d)	returns the arc sine of an angle
ATAN(d)	returns the arc tangent of an angle
ATAN2(a,b)	returns the tangent of a/b
BITAND(a,b)	returns a & b
BITOR(a,b)	returns a | b
CEILING(d)	returns the smallest integer that is not less than d
COS(d)	returns the cosine of an angle
COT(d)	returns the cotangent of an angle
DEGREES(d)	converts radians to degrees
EXP(d)	returns e (2.718...) raised to the power of d
FLOOR(d)	returns the largest integer that is not greater than d
LOG(d)	returns the natural logarithm (base e)
LOG10(d)	returns the logarithm (base 10)
MOD(a,b)	returns a modulo b
PI()	returns pi (3.1415...)
POWER(a,b)	returns a raised to the power of b
RADIANS(d)	converts degrees to radians
RAND()	returns a random number x bigger or equal to 0.0 and smaller than 1.0
ROUND(a,b)	rounds a to b digits after the decimal point
SIGN(d)	returns -1 if d is smaller than 0, 0 if d==0 and 1 if d is bigger than 0
SIN(d)	returns the sine of an angle
SQRT(d)	returns the square root
TAN(d)	returns the trigonometric tangent of an angle
TRUNCATE(a,b)	truncates a to b digits after the decimal point

Table A–6 describes the string functions.

Table A–6 String Functions

Function	Description
ASCII(s)	returns the ASCII code of the leftmost character of s
BIT_LENGTH(s)	returns the string length in bits
CHAR(c)	returns a character that has the ASCII code c
CHAR_LENGTH(s)	returns the string length in characters

Table A–6 (Cont.) String Functions

Function	Description
CONCAT(str1,str2)	returns str1 + str2
DIFFERENCE(s1,s2)	returns the difference between the sound of s1 and s2
HEXTORAW(s1)	returns the string translated from hexadecimal to raw
INSERT(s,start,len,s2)	returns a string where len number of characters beginning at start has been replaced by s2
LCASE(s)	converts s to lower case
LEFT(s,count)	returns the leftmost count of characters of s
LENGTH(s)	returns the number of characters in s
LOCATE(search,s,[start])	returns the first index (1=left, 0=not found) where search is found in s, starting at start
LTRIM(s)	removes all leading blanks in s
OCTET_LENGTH(s)	returns the string length in bytes
RAWTOHEX(s)	returns translated string
REPEAT(s,count)	returns s repeated count times
REPLACE(s,replace,s2)	replaces all occurrences of replace in s with s2
RIGHT(s,count)	returns the rightmost count of characters of s
RTRIM(s)	removes all trailing blanks
SOUNDEX(s)	returns a four character code representing the sound of s
SPACE(count)	returns a string consisting of count spaces
SUBSTR(s,start[,len])	(alias for substring)
SUBSTRING(s,start[,len])	returns the substring starting at start (1=left) with length len. Another syntax is SUBSTRING(s FROM start [FOR len])
TRIM	TRIM([{LEADING | TRAILING | BOTH}] FROM s): removes trailing and/or leading spaces from s.
UCASE(s)	converts s to upper case
LOWER(s)	converts s to lower case
UPPER(s)	converts s to upper case

Table A–7 describes the date and time functions.

Table A–7 Date and Time Functions

Function	Description
CURDATE()	returns the current date
CURTIME()	returns the current time
CURRENT_DATE	returns the current date
CURRENT_TIME	returns the current time
CURRENT_TIMESTAMP	returns the current timestamp

Table A–7 (Cont.) Date and Time Functions

Function	Description
DATEDIFF(s, d1,d2)	returns the counts of unit of times specified in s elapsed from datetime d1 to datetime d2. s may take the following values: 'ms'='millisecond', 'ss'='second','mi'='minute','hh'='hour', 'dd'='day', 'mm'='month', 'yy' = 'year'.
DAYNAME(date)	returns the name of the day
DAYOFMONTH(date)	returns the day of the month (1-31)
DAYOFWEEK(date)	returns the day of the week (1 means Sunday)
DAYOFYEAR(date)	returns the day of the year (1-366)
EXTRACT	EXTRACT ({YEAR | MONTH | DAY | HOUR | MINUTE | SECOND} FROM <datetime>): extracts the appropriate part from the <datetime> value.
HOUR(time)	return the hour (0-23)
MINUTE(time)	returns the minute (0-59)
MONTH(date)	returns the month (1-12)
MONTHNAME(date)	returns the name of the month
NOW()	returns the current date and time as a timestamp
QUARTER(date)	returns the quarter (1-4)
SECOND(time)	returns the second (0-59)
WEEK(date)	returns the week of this year (1-53)
YEAR(date)	returns the year

Note that A date value starts and ends with ', the format is yyyy-mm-dd (see java.sql.Date). A time value starts and ends with ', the format is hh:mm:ss (see java.sql.Time).

Table A–8 describes the system functions.

Table A–8 System Functions

Function	Description
IFNULL(exp,value)	if exp is null, value is returned else exp
CASEWHEN(exp,v2,v2)	if exp is true, v1 is returned, else v2
CONVERT(term,type)	converts exp to another data type
COALESCENCE(e1,e2,e3,...)	if e1 is not null then it is returned, else e2 is evaluated. If e2 is null, then is it returned, else e3 is evaluated and so on.
NULLIF(v1,v2)	returns v1 if v1 is not equal to v2, else returns null
CASE WHEN	There are two syntax for the CASE WHEN statement: CASE v1 WHEN v2 THEN v3 [ELSE v4] END: if v1 equals v2 then returns v3 [otherwise v4 or null if ELSE is not specified]. CASE WHEN e1 THEN v1[WHEN e2 THEN v2] [ELSE v4] END: when e1 is true return v1 [optionally repeated for more cases] [otherwise v4 or null if there is no ELSE]
CAST(term AS type)	converts exp to another data type

Table A–9 describes the system and connection functions.

Table A–9 System and Connection Functions

Function	Description
DATABASE()	returns the name of the database of this connection
USER()	returns the user name of this connection
IDENTITY()	returns the last identity values that was inserted by this connection

A.5 JDBC API Implemented Features

Table A–10 lists the JDBC API features of the Oracle Data Integrator driver for LDAP.

Table A–10 JDBC API Features

Feature Groups	JDBC Version	Support
Batch Update	2.0 Core	Yes
Blob/Clob	2.0 Core	No
JNDI DataSources	2.0 Optional	No
Failover support	-	No
Transaction SavePoints	3.0	No
Unicode support	-	No
Disributed Transaction	2.0 Optional	No
Connection Pooling	2.0 Optional	No
Cluster support	-	No

The following table identifies the JDBC classes supported by the Oracle Data Integrator driver for LDAP.

Table A–11 JDBC Classes

JDBC Classes	JDBC Version	Support
Array	2.0 Core	No
Blob	2.0 Core	No
Clob	2.0 Core	No
CallableStatement	1.0	Yes
Connection	1.0	Yes
ConnectionPoolDataSource	2.0 Optional	No
DatabaseMetaData	1.0	Yes
DataSource	2.0 Optional	No
Driver	1.0	Yes
PreparedStatement	1.0	Yes
Ref	2.0 Core	No
RowSet	2.0 Optional	No
ResultSet	1.0	Yes
ResultSetMetaData	1.0	Yes
Statement	1.0	Yes

Table A–11 (Cont.) JDBC Classes

JDBC Classes	JDBC Version	Support
Struct	2.0 Core	No
XAConnection	2.0 Optional	No
XADataSource	2.0 Optional	No

B

Oracle Data Integrator Driver for XML Reference

This appendix describes how to work with the Oracle Data Integrator driver for XML.

This appendix includes the following sections:

B.1 Introduction to Oracle Data Integrator Driver for XML

Oracle Data Integrator Driver for XML (*XML driver*) handles an XML document as a JDBC data source. This allows Oracle Data Integrator to use XML documents as data servers.

With Oracle Data Integrator Driver for XML, Oracle Data Integrator can query XML documents using standard SQL syntax and perform changes in the XML files. These operations occur within transactions and can be committed or rolled back.

The Oracle Data Integrator driver for XML supports the following features:

- Standard SQL (Structured Query Language) Syntax
- Correlated subqueries, inner and outer joins
- ORDER BY and GROUP BY
- COUNT, SUM, MIN, MAX, AVG and other functions
- Standard SQL functions
- Transaction Management
- Referential Integrity (foreign keys)
- Saving Changes made on XML data into the XML files

B.2 XML Processing Overview

The XML driver works in the following way:

1. The driver *loads* (upon connection or user request) the XML structure and data into a relational schema, using a XML to SQL Mapping.
2. The user works on the relational schema, manipulating data through regular SQL statements or specific driver commands for driver operations.
3. Upon disconnection or user request, the XML driver *synchronizes* the data and structure stored in the schema back to the XML file.

B.2.1 XML to SQL Mapping

The XML to SQL Mapping is a complex process that is used to map a hierarchical structure (XML) into a relational structure (schema). This mapping is automatic.

Elements and Attributes Mapping

The XML driver maps XML elements and attributes the following way:

- Elements are mapped as tables with the same name.
- Attributes are mapped as columns named like the attributes. Each column is created in the table representing the attribute's element.

Hierarchy & Order Mapping

Extra data may appear in the relational structure as follows:

- In order to map the hierarchy of XML elements, or a one-to-many relation between elements, the XML driver generates in each table corresponding to an element the following extra columns:
  - `<element_name>PK`: This column identifies the element.
  - `<parent_element_name>FK`: This column links the current element to its parent in the hierarchy. It contains a value matching the parent element's `<element_name>PK` value.
- Records in a table, unlike elements in an XML file, are not ordered, unless a specific column is used to define the order. The driver generates also a column named `<element_name>ORDER` to preserve the order of the elements. When adding new rows in the relational schema, make sure that the ORDER column is correctly set to have the elements correctly ordered under the parent element.
- The root of the hierarchy is identified by a root table named after the root element. This table contains a single records with the following columns:
  - `<root_element_name>PK`: All level 1 sub-elements will refer to this PK entry.
  - `SNPSFILENAME`: This column contains the names of the XML file loaded into this schema.
  - `SNPSFILEPATH`: This column contains the XML file path.
  - `SNPSLOADDATE`: This column contains the date and time when the file was loaded into the schema.

  The values in this table are managed by the driver and should not be modified.

Mapping Exceptions

This section details some specific situations for the mapping of extra data.

- Elements containing only `#PCDATA` are not mapped as tables, but as columns of the table representing their parent element. These columns are named `<element_name>DATA`.
- List Attributes are mapped as a new table with a link (PK, FK) to the table representing the element containing the list.
- XML elements and attributes with names that match SQL keywords are automatically renamed (an underscore is added after their name) in the relational structure to avoid naming conflict between table/column names and SQL keywords. For example, an element named `SELECT` will be mapped to a table named `SELECT_`. Such elements are restored in the XML file with their original naming when a synchronize operation takes place.

Note that extra objects created by the driver are used to keep the XML file consistency. These records must be loaded in the relational schema before it is synchronized to an XML file.

B.2.2 XML Namespaces

The XML driver supports XML namespaces (`xmlns:`) specified for XML attributes and elements.

Elements or attributes specified with a namespace (using the syntax `<namespace>:<element or attribute name>`) are mapped as tables or columns prefixed with the namespace using the syntax: `<namespace>_<element or attribute name>`. When synchronizing the XML data back to the file, the namespace information is automatically generated.

B.2.3 Managing Schemas

A *schema* corresponds to the concept used in Oracle database and other RDBM systems and is a container that holds a set of relational tables. A schema is a generic relational structure in which an entire set of XML file instances may be successfully parsed and extracted. The identified elements and attributes are inserted in the appropriate relational tables and fields.

This schema is generated by the XML driver from either an XML instance file, a DTD file, or an XSD file. It is recommended to generate the schema from a DTD or XSD file.

Note that only a single DTD or XSD file may be referenced in definition of an XML data server URL. In this case, this DTD or XSD may be considered as a master DTD or XSD file if the artifact includes references to other DTD / XSD files. Note that in certain cases multiple schemas may be required. In this case use the `add_schema_bundle` property.

B.2.3.1 Schema Storage

The schema may be stored either in a *built-in engine* or in an *external database*.

- The *built-in engine* requires no other component to run. The XML schema is stored in memory within the driver. The SQL commands and functions available on this driver are detailed in the SQL Syntax.
- The *external database* can be a relational database management system. The driver connects through JDBC to this engine, and uses it to store the schema. This enables the:

- Use of the processing and storage power of the RDBMS engine
- Use of the statements and functions of the RDBMS
- Persistence of schema storage

See Section B.3.3, "Using an External Database to Store the Data" for more information.

B.2.3.2 Multiple Schemas

It is possible to handle, within the same JDBC connection, multiple schemas and to load multiple XML files simultaneously. It is possible to CREATE, TRUNCATE, SET, and LOAD FILE INTO schemas. When connecting to the JDBC driver, you connect to the schema that is specified on the URL. It is possible to set the current schema to another one using the SET SCHEMA command. See Section B.4, "Detailed Driver Commands" for more information.

The *default schema* is a specific schema that is used for storing temporary data. This default schema cannot be associated with an XML file.

It is also possible to automatically create additional schemas with different XML structures when creating the connection to the driver. See Section B.3.1, "Driver Configuration" for more information.

B.2.3.3 Accessing Data in the Schemas

Data in the schemas is handled using the SQL language.

It is possible to access tables in a schema that is different from the current schema. To access the tables of a different schema, prefix the table name with the schema name, followed by a period character (`.`). For example:

```
SELECT col1, schema2.table2.col2, table1.col3 FROM table1, schema2.table2.
```

This query returns data from table1 in the current schema, and from table2 from schema2.

Note: Note that the other schema must be located on the same storage space - *built-in engine* or *external database* - as than the current schema.

B.2.3.4 Case Sensitivity

A schema cannot be case-sensitive. All elements in the schema (tables and columns) are in UPPERCASE. If the XML file element names contain lowercase letters, they are converted to upper case. When the elements are synchronized to the XML file, their names are created with their original case.

B.2.3.5 Loading/Synchronizing

A schema is usually automatically created when connecting to an XML file, and loaded with the data contained in the XML file. It is possible to force the schema creation and the data loading in the schema using specific driver commands. See Section B.4, "Detailed Driver Commands" for more information. It is also possible to force a synchronization process of the data by using the `SYNCHRONIZE` command, as described in Section B.4.9, "SYNCHRONIZE".

B.2.4 Locking

When accessing an XML file, the driver locks it in order to prevent other instances of the driver to connect to the file. The lock file has the same name as the XML file but an `.lck` extension.

If the driver is incorrectly disconnected, a lock may remain on the file. To remove it, delete the `.lck` file. It is also possible to unlock an XML file with the UNLOCK FILE command.

B.2.5 XML Schema (XSD) Support

XSD is supported by the XML driver for describing XML file structures. See Section B.7, "XML Schema Supported Features" for more information.

In addition, the XML driver supports document validation against XSD schemas specified within the XML file. This operation may be performed using the VALIDATE driver specific command.

B.3 Installation and Configuration

The Oracle Data Integrator driver for XML is automatically installed with Oracle Data Integrator. The following topics cover advanced configuration topics and reference information.

This section contains the following topics:

- Driver Configuration
- Automatically Create Multiple Schemas
- Using an External Database to Store the Data

> **Note:** If using an External Database storage, you must also make sure that the JDBC driver used to connect the external database, as well as the `.properties` file are in the classpath.

B.3.1 Driver Configuration

This section details the driver configuration.

- The driver name is: `com.sunopsis.jdbc.driver.xml.SnpsXmlDriver`
- The URL Syntax is:
  `jdbc:snps:xml?[property=value&property=value...]`

The properties for the URL are detailed in Table B–1.

Table B–1 Driver Properties

Property	Mandatory	Type	Default	Description
file or f	No	string (file location)	-	XML file name. Use slash "/" in the path name instead of back slash "\". It is possible to use an HTTP, FTP or File URL to locate the file. Files located by URL are read-only. For an XML file, if this property is missing, a relational schema is created by the XML driver from the DTD/XSD file and no XML file is searched for.
dtd or d	No	string (file location)	-	Description file: This file may be a DTD or XSD file. It is possible to use an HTTP, FTP or File URL to locate the file. Files located by URL are read-only. Note that the DTD or XSD file that is specified in the URL takes precedence over the DTD or XSD file that is specified within the XML file. References should be made with an absolute path. For an XML file, if this property is missing, and no DTD or XSD is referenced in the XML file, the driver will automatically consider a DTD file name similar to the XML file name with `.dtd` extension. A DTD file may be created from the XML file structure depending on the generate_dtd URL property. Note that when no DTD or XSD file is present, the relational structure is built using only the XML file content. It is not recommended to reverse-engineer the data model from such a structure as one XML file instance may not contain all the possible elements described in the DTD or XSD, and data model may be incomplete.
root_elt or re	No	String	-	Name of the element to take as the root table of the schema. This value is case sensitive. This property can be used for reverse-engineering for example a specific message definition from a WSDL file, or when several possible root elements exist in a XSD file. Important: This property is used to designate ONLY the Element in the XSD / DTD file which will serve as the Root Element DEFINITION of any XML instance file Root Element.
read_only or ro	No	boolean (true | false)	false	Open the XML file in read only mode.
schema or s	No	string	-	Name of the schema where the XML file will be loaded. If this property is missing, a schema name is automatically generated from the XML file name. If this property is not specified in the XML data Server URL, the XML Driver will automatically create a schema name. This schema will be named after the five first letters of the XML file name. **Note:** It is not possible to make more than one connection to a schema. Subsequent connections fail if trying to connect to a schema already in use. Important: The schema name should be specified in uppercase. Important: It is forbidden to have a schema name identical to an XML ELEMENT name.

Table B–1 (Cont.) Driver Properties

Property	Mandatory	Type	Default	Description
standalone	No	boolean (true | false)	false	If this option is set to true, the schema for this connection is completely isolated from all other schemas. With this option, you can specify the same schema name for several connections, each schema being kept separated. When using this option, tables in this schema cannot be accessed from other schemas, and this connection cannot access tables from other schemas.
ns_prefix_ generation or nspg	No	auto | xml | xsd	auto	This option defines how namespace prefixes are generated and written in the XML file. ■ auto (default): Prefixes are automatically generated from the namespace names themselves when possible or generated as ns1, ns2, etc. ■ xml: Namespace prefixes are taken from the source XML file, if any. ■ xsd: Namespace prefixes are taken from the XSD file, if any. Note that the xsd option value assumes that a similar prefix is not used in several XSD files to reference a different namespace.
no_default_ns or ndns	No	boolean (true | false)	false	If this property is set to true, the driver generates the target file with no default namespace entry.
no_closing_tags or nct	No	boolean (true | false)	false	If this property is set to true, the driver generates the empty tags without their closing tags (for example <element/>). If set to false the driver generates an empty element as <element></element>. This property is true by default if the v1_compatibility property is used.
db_props or dp	No	string	-	This property is used to use an external database instead of the memory engine to store the schema. The db_props property indicates that the schema must be loaded in a database schema whose connection information are stored in a **external database property file** named like the db_props property with the extension `.properties`. This property file must be located in the application's classpath.
load_data_on_ connect or ldoc	No	boolean (true | false)	true	Load automatically the data in the schema when performing the JDBC connection. If set to false, a SYNCHRONIZE statement is required after the connection to load the data. This option is useful to test the connection or browse metadata without loading all the data.
drop_on_disc or dod	No	boolean (true | false)	false	Drop automatically the schema when closing the JDBC connection. If true, the schema is stored in the built-in engine, it is always dropped. If the schema is stored in an external database, the driver attempts to drop the database schema, but might fail if tables still exist in this schema. The drop_tables_ on_drop_schema property can be specified in the external database property file to ensure that all tables are automatically dropped when the schema is dropped.

Table B–1 (Cont.) Driver Properties

Property	Mandatory	Type	Default	Description
ignore_ unknown_ elements or iue	No	boolean (true | false)	false	Ignore all elements in the XML file that do not exist in the associated DTD (Document Type Definition) or XSD (XML Schema Definition) file.
useMaxValue	No	boolean (true | false)	false	When this property is set to true, elements for which maxOccurs is not specified in the XSD are considered as maxOccurs ="unbounded". Otherwise, the driver assumes that maxOccurs=1 when maxOccurs is not specified.
generate_dtd or gd	No	yes | no | auto	auto	Defines if a DTD file must be created from the XML file structure: ■ auto: create the DTD file if the it does not exist. if the DTD exists, does nothing. ■ yes: always create the DTD file. An existing DTD will be overwritten. ■ no: never create the DTD file. The DTD file must exist. Warning: DTD files created using this option contain only the definition of XML elements appearing in the XML file, and may not be complete.
java_encoding or je	No	string (encoding code)	UTF8	Target file encoding (for example: `ISO8859_1`). You will find a list of supported encoding at the following URL: http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html. Note that if the Java encoding is specified, the XML encoding should also be specified.
xml_encoding or xe	No	string (encoding code)	UTF8	Encoding specified in the generated XML File, in the tag (for example `ISO-8859-1`: `<?xml version="1.0" encoding="ISO-8859-1"?>`. You will find a list of supported encoding at the following URL: http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html. Note that if the XML encoding is specified, the Java encoding should also be specified.
v1_compatibility or v1	No	boolean (true | false)	false	With this property set to true, the driver performs the XML to SQL mapping as if in version 1.x. This property is provided for compatibility.
numeric_id or ni	No	boolean (true | false)	true	If set to true, all internal Primary and Foreign Keys are of NUMERIC type. Otherwise, they are of the VARCHAR type.
id_length or il	No	integer	10 / 30	The length of the internal Primary and Foreign Key columns. The default is 10 for NUMERIC column types and 30 for VARCHAR column.
numeric_scale or ns	No	integer	empty	Scale of the numeric columns generated during the XML to SQL mapping.
no_batch_ update or nobu	No	boolean (true | false)	false	Batch update is not used for this connection. The command to set the batch update is not sent. This prevents errors to occur for external databases that do not support this JDBC feature, or allows to debug errors related to batch update usage.

Table B–1 (Cont.) Driver Properties

Property	Mandatory	Type	Default	Description
add_schema_ bundle or asb	No	string	-	Additional schemas bundle file. This property indicates that additional schemas must be created at connection time. The description for these extra schemas are located in an **additional schemas property file** named like the add_schema_bundle property with the extension ".properties". The additional schemas property file contains a list of valid JDBC driver's URL. In this file, the property names are ignored. Only the list of values is taken into account. All these additional schemas are created with the drop_ on_disconnect option set to true by default. Example of additional schemas property files contents: `addschema_ 1=jdbc:snps:xml?f=c:/myfile.xml&ro=true&s =myschema1 addschema_ 2=jdbc:snps:xml?file=c:/myfile2.xml&s=mys chema2 addschema_ 3=jdbc:snps:xml?d=c:/myfile3.dtd&s=mysche ma3`
add_schema_ path or asp	No	string (directory)	-	Directory containing a set of XSD files. For each XSD file in this directory, an additional schema is created in the built-in engine or external database storage, based on this XSD. Note that no object is created in the external database storage for these additional schemas. The schema names are default generated named (5 first characters of the file name, uppercased).
log_file or lf	No	string (file location)	-	Log file name. If the log file is empty, the trace is displayed in the standard output. The presence of this property triggers the trace for the schema. Each schema may have a different trace file.
log_level or ll	No	Integer	-	Log level. The log level is a mask of the following values: ■ 1: Important internal events ■ 2: Detailed internal events ■ 4: Native SQL commands ■ 8: XML-Relational mapping calculation ■ 16: XML-Relational mapping validation (Table names changes,etc) Examples: ■ Trace Important & Detailed internal events: `log_ level=3 (1+2)` ■ Trace Native SQL commands and Important internal events: `log_level=5 (1+4)` ■ Trace XML-Relational mapping calculation and validation: `log_level=24 (16+8)` ■ Trace all events: `log_level=31 (1+2+4+8+16)`

Table B–2 listsURL samples.

Table B–2 URL Samples

URL Sample	Action
`jdbc:snps:xml`	Connects to the default schema.
`jdbc:snps:xml?f=/tmp/myfile.xml&ro=true&/tmp/mydtd.dtd`	Open the `/tmp/myfile.xml` file in read only mode, using the `/tmp/mydtd.dtd` DTD.
`jdbc:snps:xml?file=/tmp/myfile.xml`	Open the `/tmp/myfile.xml` file in read/write mode.
`jdbc:snps:xml?s=myschema`	Connect directly to the schema `myschema`

B.3.2 Automatically Create Multiple Schemas

It is possible to automatically create additional schemas with different XML structures when creating the connection with the driver. This is performed by:

- Declaring in the add_schema_bundle URL property a property file that contains a list of JDBC URLs corresponding to the different additional schemas to create.
- Declaring in the add_schema_path URL property a directory that contains a set of XSD files. For each XSD file an additional schema is created in the built-in engine, based on the XML schema description.
- Specifying additional valid driver URLs as JDBC properties, named addschema_X (X is a number). An additional schema will be created for each URL found in a JDBC property called addschema_X.

Note that all these additional schemas are automatically dropped when their last connection is closed.

These schemas are created in addition to the one that may be created with the properties specified in the JDBC driver URL

B.3.3 Using an External Database to Store the Data

In most cases, the XML driver stores the relational schema mapping of the XML schema in a *built-in engine*. It is also possible to make the driver store the relational schema in an external relational database.

Note: Supported RDBMS for external storage include Oracle, Microsoft SQL Server, MySQL, and Hypersonic SQL. The complete list of technologies that support external storage is available on Oracle Technical Network (OTN) :

`http://www.oracle.com/technology/products/oracle-data-integrator/index.html`.

Using an external database

To use an external database, you must have:

- A file with the `.properties` extension. This External Database Properties File contains the properties of a JDBC connection to the relational database schema.
- This `.properties` file in the classpath of Oracle Data Integrator that uses the XML driver. Typically, you can install it with your custom drivers.

- The db_props property in the JDBC URL set to the name of the properties file (without the extension `.properties`).

> **Note:** When connecting to the external database, the XML driver uses JDBC connectivity. Make sure that the JDBC driver to access this external database is also available in the ODI classpath.

It is possible to set or override the external database properties on the URL. These properties must be prefixed with the string `dp_`. For example:

```
jdbc:snps:xml?file=/temp/payload.xml&dp_driver=<external_db_driver>&dp_
url=<external_db_url>
```

External Database Properties File

The external database properties file is a text file containing a set of lines with on each line a `<property>=<value>` pair:

The valid properties are described in Table B–3.

Table B–3 Properties of the External Database Properties File

Property	Mandatory	Type	Default	Description
driver	Yes	string	-	JDBC driver name. Important: The driver class file must be in the classpath of the java application.
url	Yes	string	-	JDBC URL
user	Yes	string	-	Login used to connect the database
password	Yes	string	-	Encrypted password of the user. **Note:** To encrypt the password, use the `encode.bat` command. See the *Oracle Fusion Middleware Installation Guide for Oracle Data Integrator* for more information.
schema	Yes	string	-	Database schema storing the relational schema and the XML data. This property should not be used for Microsoft SQLServer, and the catalog property should be used instead.
catalog	Yes	string	-	For Microsoft SQL Server only. Database catalog storing the XML data & information.
drop_on_connect or doc	No	boolean (Y|N)	N	Drop the tables from the database schema if they already exist. If set to N the existing tables are preserved.
create_tables or ct	No	(Y | N | AUTO)	AUTO	Y: create systematically the tables in the schema. N: never create the tables in the schema AUTO: Create the tables if they do not exist.
create_indexes or ci	No	boolean (Y|N)	Y	Y: create indexes on tables' PK and FK N: do not create the indexes. This value provides faster INSERT but dramatically slows SELECT in the data. It also saves storage space on your RDB.
truncate_before_load or tbl	No	boolean (Y|N)	Y	Y: truncate all data when connecting N: preserve existing data

Table B–3 (Cont.) Properties of the External Database Properties File

Property	Mandatory	Type	Default	Description
ids_in_db or iidb	No	boolean (Y | N)	Y	Y: preserve identifiers (counters) in the database for a future append connection N: do not preserve identifiers. Future append is not possible.
drop_tables_on_drop_schema or dtods	No	boolean (Y | N)	Y	Y: a DROP SCHEMA does not only causes the reference to the database schema to be erased from the driver, but also causes all tables to be dropped. N: DROP SCHEMA erases the reference to the database schema from the driver, but the tables are kept in the database schema.
use_prepared_statements or ups	No	boolean (Y | N)	Y	Y: use the prepared statements with the database connection to perform driver operation (load/unload files). N: do not use the prepare statement. Processing is usually faster with prepare statement. The database and driver must support prepared statements in order to use this option.
use_batch_update or ubu	No	boolean (Y | N)	Y	Y: use batch update with the database connection. N: do not use batch update. Inserting data is usually faster with batch update. Should be set to Yes only if the following conditions are met: ■ The database and driver support batch update ■ The database supports prepared statements ■ The use_prepared_statements parameter is set toYes **Note:** The batch update options specified here are only used to load the data in the schema. To use batch update when manipulating data in the schema, you must specify batch update options in your Java application.
batch_update_size or bus	No	integer	30	Batch update size. Records will be written in the database schema by batches of this size, if the use_batch_update property is set to Y.
commit_periodically or cp	No	boolean (Y | N)	Y	A COMMIT will be sent regularly when loading data from the XML file into the database schema. This regular COMMIT avoids overloading of the database log when loading large XML data files. Should be set to Yes only if the following conditions are met: ■ The database supports batch update ■ The database supports prepared statements ■ The use_prepared_statements parameter is set toYes ■ The use_batch_updates parameters is set to Yes **Note:** The commit options specified here are only used to load the data in the schema. To commit when performing transactions in the schema, you must specify the commit in your Java application.
num_inserts_before_commit or nibc	No	integer	1000	Interval in records between each COMMIT, if the commit_periodically property is set to Y.

Table B–3 (Cont.) Properties of the External Database Properties File

Property	Mandatory	Type	Default	Description
reserve_chars_ for_column or rcfc	No	integer	3	Long XML names are truncated to fit the maximum allowed size on the RDBMS, according to the maximum allowed size for column names returned by the JDBC driver. However, there are some situations when you will want to reserve characters to make the driver-generated names shorter. The number of reserved character is defined in the reserve_chars_for_column value. For example, on a database with a maximum of 30 characters and with this property set to 3 (which is the default), all column names will not be larger than 27 characters.
reserve_chars_ for_table or rcft	No	integer	3	Same as reserve_chars_for_column (rcfc) property but applies to names of the table created in the RDBMS schema.
varchar_length or vl	No	integer	255	Size of all the columns of the relational structure that will be used to contain string data.
numeric_length or nl	No	integer	30	Size of all the columns of the relational structure that will be used to contain numeric data.
unicode	No	boolean (true | false)		For MS SQL Server: If unicode = true, nvarchar is used. If unicode = false or not set, varchar is used.

The following sample is an example of a property file for using an Oracle Database as the external storage:

```
driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@HOST:PORT:SID
user=USER_NAME
password=ENCODED_PASSWORD
schema=USER_NAME
drop_on_connect=Y
create_tables=AUTO
create_indexes=Y
truncate_before_load=Y
ids_in_db=Y
drop_tables_on_drop_schema=Y
use_prepared_statements=Y
use_batch_update=Y
batch_update_size=30
commit_periodically=Y
num_inserts_before_commit=1000
reserve_chars_for_column=3
reserve_chars_for_table=3
```

The following sample is an example of a property file for using a Microsoft SQL Server database as the external storage:

```
driver=com.microsoft.jdbc.sqlserver.SQLServerDriver
url=jdbc:microsoft:sqlserver://SERVER_NAME:PORT;SelectMethod=cursor
user=USER_NAME
password=ENCODED_PASSWORD
schema=OWNNER_NAME
drop_on_connect=Y
create_tables=AUTO
```

```
create_indexes=Y
truncate_before_load=Y
ids_in_db=Y
drop_tables_on_drop_schema=Y
use_prepared_statements=Y
use_batch_update=Y
batch_update_size=30
commit_periodically=Y
num_inserts_before_commit=1000
reserve_chars_for_column=3
reserve_chars_for_table=3
```

B.4 Detailed Driver Commands

The following statements are specific to the XML driver, and allow to manage XML files and schemas. They can be launched as standard SQL statements on the JDBC connection to the XML driver.

To manipulate the data stored in the schemas, you may use standard SQL syntax. This syntax is either the built-in engine's SQL Syntax, or the SQL Syntax of the External Database engine you use.

Conventions

The following conventions are used within this document:

- [A] means A is optional
- [A | B] means A or B but the parameter is optional.
- { B | C } means B or C must be used.
- [A] [B] means a set of arguments that are not ordered.
- (and) are the characters '(' and ')'.
- keywords are in UPPERCASE

This section details the following driver specific commands:

- CREATE FILE
- CREATE XMLFILE
- CREATE FOREIGNKEYS
- CREATE SCHEMA
- DROP FOREIGNKEYS
- DROP SCHEMA
- LOAD FILE
- SET SCHEMA
- SYNCHRONIZE
- UNLOCK FILE
- TRUNCATE SCHEMA
- VALIDATE
- COMMIT
- CREATE TABLE

- DELETE
- DISCONNECT
- DROP TABLE
- INSERT INTO
- ROLLBACK
- SELECT
- SET AUTOCOMMIT
- UPDATE

B.4.1 CREATE FILE

Generate an XML file called <file_name> from the default schema data, or from a specific schema.

If the EMPTY option is specified, an empty file with the XML structure specified in the DTD or XSD is generated.

```
CREATE [EMPTY] FILE <file_name> [FROM SCHEMA <schema_name>]
    [JAVA_ENCODING <java_encoding>  XML_ENCODING <xml_encoding>]
    [NO_CLOSING_TAGS] [NO_DEFAULT_NS]
```

Parameters

FROM SCHEMA
Specify the schema in which data will be written in the XML file.

JAVA_ENCODING
Encoding of the generated File.

XML_ENCODING
Encoding generated in the file's xml tag.

Example of generated tag: `<?xml version="1.0" encoding="ISO-8859-1"?>`

Note that Java and XML encoding should always be specified together.

NO_CLOSING_TAG
If this parameter is specified, the driver generates the empty tags with closing tag. By default, the driver generates an empty element as `<element></element>`. with the no_closing_tags parameter, it generates `<element/>`.

NO_DEFAULT_NS
If this parameter is specified, the driver generates the target file without a default namespace entry.

Remarks

- If the file name contains spaces, enclose it in double quotes
- The encoding values should be enclosed in double quotes as they may contain special characters.

B.4.2 CREATE XMLFILE

Generate an XML file called <file_name> from the default schema data, or from a specific schema.

```
CREATE XMLFILE <file_name> [FROM SCHEMA <schema_name>]
    [JAVA_ENCODING <java_encoding> XML_ENCODING <xml_encoding>]
    [NO_CLOSING_TAGS][NO_DEFAULT_NS]
```

Parameters

FROM SCHEMA

Specify the schema in which data will be written in the XML file.

JAVA_ENCODING

Encoding of the generated File.

XML_ENCODING

Encoding generated in the file's xml tag. Example of generated tag: `<?xml version="1.0" encoding="ISO-8859-1"?>.`

Note that Java and XML encoding should always be specified together.

NO_CLOSING_TAG

If this parameter is specified, the driver generates the empty tags with closing tag. By default, the driver generates an empty element as `<element></element>.` with the no_closing_tags parameter, it generates `<element/>`.

NO_DEFAULT_NS

If this parameter is specified, the driver generates the target file without a default namespace entry.

Remarks

- If the file name contains spaces, enclose it in double quotes
- The encoding values should be enclosed in double quotes as they may contain special characters.

B.4.3 CREATE FOREIGNKEYS

Create physically all the foreign keys joining the tables from the relational schema in the database. This command is helpful to enforce integrity constraints on the schema.

> **Note:** When requested, the driver always returns "virtual" foreign keys, corresponding to the relational structure mapping. It does not return the real foreign keys enforced at database level.

```
CREATE FOREIGNKEYS
```

Remarks

After using CREATE FOREIGNKEYS, it is not possible any longer to perform a LOAD FILE.

B.4.4 CREATE SCHEMA

Create in <schema_name> an empty schema or a schema with tables mapping the structure of the description file specified as <dtd/xsd_name>.

```
CREATE SCHEMA <schema_name> [WITH DTD <dtd/xsd_name>] [REPLACE]
   [ROOTELT <root element>] [READONLY]
   [JAVA_ENCODING <java_encoding> XML_ENCODING <xml_encoding>]
```

Parameters

WITH DTD

Specify the description file (DTD or XSD) which structure will be created in the schema.

REPLACE

Specify if an existing schema structure must be replaced with the new one.

ROOTELT

Element in the description file considered as the root of the XML file. This element name is case sensitive.

READONLY

The schema loaded cannot have data inserted, deleted or updated.

JAVA_ENCODING

Encoding of the target XML file(s) generated from schema.

Note: Java and XML encoding should always be specified together.

XML_ENCODING

Encoding generated in the target files' XML tag. Example of generated tag: <?xml version="1.0" encoding="ISO-8859-1"?>.

Remarks

- The XML file data is not loaded. This command is similar to LOAD FILE but does not load the XML file data.
- The schema is created in READONLY mode since no XML file is associated with it.
- The connection schema does not automatically switch to the newly created schema.
- If the file name contains spaces, enclose the name id double quotes.
- The encoding values should be enclosed in double quotes as they may contain special characters.

B.4.5 DROP FOREIGNKEYS

Drop all the foreign keys on the tables of the relational schema in the database. This command is helpful to drop all integrity constraints on the schema.

```
DROP FOREIGNKEYS
```

B.4.6 DROP SCHEMA

Drop an existing schema. If <schema_name> is not specified, the current schema is dropped. It is not possible to drop a schema if there are pending connections to this schema. Trying to drop a schema with existing connections causes an exception.

```
DROP SCHEMA [<schema_name>]
```

B.4.7 LOAD FILE

Load the <file_name> XML file into the current relational schema.

```
LOAD FILE <file_name> [WITH DTD <dtd/xsd_name> | INSERT_ONLY] [ON SCHEMA <schema_
name>] [REPLACE] [READONLY] [ROOTELT <root element>] [AUTO_UNLOCK] [DB_PROPS
<external database properties>]
```

Parameters

WITH DTD

Specify the description file (DTD or XSD) which structure will be created in the schema.

INSERT_ONLY

Adds the data from the XML file in the schema if it already exists. The new XML file should have valid description file for the existing schema.

ON SCHEMA

Force the file to be loaded in <schema_name>. Note that the current schema is not set after the command automatically to <schema_name>.

REPLACE

Specify if an existing schema structure with the same name must be replaced with the one that is being loaded.

READONLY

The schema loaded cannot have data inserted, deleted or updated.

ROOTELT

Element in the description file considered as the root of the XML file. This element name is case sensitive.

AUTO_UNLOCK

If the XML file is already locked by another driver instance, an exception occurs unless the AUTO_UNLOCK is specified. This parameter unlocks automatically the file if it is locked.

DB_PROPS

Loads the file in the external database identified by the properties file called <external database properties>.properties.

Remarks

- If the file name contains spaces, enclose the name in double quotes.
- When no schema is specified, the driver automatically generates a schema name from the file name.
- The connection schema does not automatically switch to the loaded schema.

- If the XML file is already open in another schema, an exception occurs.

B.4.8 SET SCHEMA

Set the current schema to <schema_name>.

```
SET SCHEMA <schema_name>
```

Remarks

If no <schema_name> is specified, the schema is set to the default schema.

B.4.9 SYNCHRONIZE

Synchronize data in the schema with the file data.

```
SYNCHRONIZE [ALL | SCHEMA <schema_name>] [FROM FILE/FROM DATABASE]
 [IGNORE CONFLICTS]
```

Parameters

ALL

Synchronizes all schemas

SCHEMA

Synchronizes only <schema_name>

FROM FILE

Forces the data to be loaded from the file to the schema. Erases all changes in the schema.

FROM DATABASE

Forces the data to be loaded from the schema to the file. Erases all changes in the file.

IGNORE CONFLICTS

If FROM FILE/DATABASE are not specified, the driver automatically determines where data have been modified (in the FILE or DATABASE) and updates the unmodified data. If both the FILE and the DATABASE have been modified, the driver issues a Conflict Error. if the IGNORE CONFLICTS parameter is used, no error is issued, and if performing a SYNCHRONIZE ALL, the following schemas will be synchronized.

Note: A schema is marked updated only when a data modification (update, delete, insert, drop) is executed in a connection to that schema. It is not marked as updated, when the order is launched from a connection to another schema.

B.4.10 UNLOCK FILE

Unlocks <file_name> if it is locked by another instance of the driver.

```
UNLOCK FILE <file_name>
```

B.4.11 TRUNCATE SCHEMA

Clears all data from the current schema, or from <schema_name>.

```
TRUNCATE SCHEMA [<schema_name>]
```

B.4.12 VALIDATE

Validates an XML file <file_name> content against the XML Schema (XSD) referenced in the XML file. This command returns an exception if the file is not valid.

```
VALIDATE [FILE <file_name>] [ERROR_ON_WARNING|IGNORE_ON_WARNING]
   [ERROR_ON_ERROR|IGNORE_ON_ERROR]
   [ERROR_ON_FATAL_ERROR|IGNORE_ON_FATAL_ERROR] [VERBOSE]
```

Parameters

FILE <file_name>
Name of the XML file to validate.

ERROR_ON_WARNING | IGNORE_ON_WARNING
Ignore or generate errors on XSD validation warnings, such as values out of range. The default value is IGNORE_ON_WARNING.

ERROR_ON_ERROR | IGNORE_ON_ERROR
Ignore or generate errors on XSD validation errors, such as non conform attribute or element. The default value is ERROR_ON_ERROR.

ERROR_ON_FATAL_ERROR | IGNORE_ON_FATAL_ERROR
Ignore or generate errors on XSD validation fatal errors, such as malformed XML. The default value is ERROR_ON_FATAL_ERROR.

VERBOSE
Displays on the Java console the detailed errors and number of the line causing the error. Nothing is displayed by default on the console.

B.5 SQL Syntax

The following statements are available when using the built-in engine to store the XML schema. They enable the management of the data and data structure in the schema through Standard SQL Syntax.

This section contains the following topics:

- SQL Statements
- SQL FUNCTIONS

> **Note:** If you are using an external database, you may use the database engine querying syntax instead of this one.

B.5.1 SQL Statements

Any number of commands may be combined. You can optionally use the semicolon character (;) to separate each command.

This section details the following commands:

- COMMIT
- CREATE TABLE
- DELETE
- DISCONNECT
- DROP TABLE
- INSERT INTO
- ROLLBACK
- SELECT
- SET AUTOCOMMIT
- UPDATE
- Expressions, Condition and Values

B.5.1.1 COMMIT

Ends a transaction on the schema and makes the changes permanent.

```
COMMIT [WORK]
```

B.5.1.2 CREATE TABLE

Create a tables and its constraints in the relational schema.

```
CREATE TABLE <table_name>
  ( <columnDefinition> [, ...] [, <constraintDefinition>...])

<columnDefinition> ::=
    <column_name> <datatype> [(anything)] [[NOT] NULL] [IDENTITY] [PRIMARY KEY]

<constraintDefinition> ::=
[ CONSTRAINT <constraint_name> ]
    UNIQUE ( <column_name> [,<column>...] ) |
    PRIMARY KEY ( <column_name> [,<column_name>...] ) |
    FOREIGN KEY ( <column_name> [,<column_name>...] )
    REFERENCES <referenced_table> ( <column_name> [,<column_name>...] )
```

Remarks

- IDENTITY columns are automatically incremented integer columns. The last inserted value into an identity column for a connection is available using the IDENTITY() function.
- Valid datatypes are: BIT, TINYINT, BIGINT, LONGVARBINARY, VARBINARY, BINARY, LONGVARCHAR, CHAR, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE, VARCHAR, DATE, TIME, TIMESTAMP, OBJECT

B.5.1.3 DELETE

Remove rows in a table in the relational schema. This function uses a standard SQL Syntax.

```
DELETE FROM <table_name> [ WHERE <expression> ]
```

B.5.1.4 DISCONNECT

Closes this connection.

```
DISCONNECT
```

Remarks

- It is not required to call this command when using the JDBC interface: it is called automatically when the connection is closed.
- After disconnecting, it is not possible to execute other queries with this connection.

B.5.1.5 DROP TABLE

Remove a table, the data and indexes from the relational schema.

```
DROP TABLE <table_name>
```

B.5.1.6 INSERT INTO

Insert one or more new rows of data into a table.

```
INSERT INTO <table_name> [ ( <column_name> [,...] ) ]
    { VALUES (<expression> [,...]) | <SELECT Statement> }
```

B.5.1.7 ROLLBACK

Undo the changes made since the last COMMIT or ROLLBACK.

```
ROLLBACK
```

B.5.1.8 SELECT

Retrieves information from one or more tables in the schema.

```
SELECT [DISTINCT] { <select_expression> | <table_name>.* | * } [, ... ]
[  INTO <new_table> ]
   FROM <table_list>
[  WHERE <expression> ]
[  GROUP BY <expression> [, ...] ]
[  ORDER BY <order_expression> [, ...] ]
[  { UNION [ALL] | {MINUS|EXCEPT} | INTERSECT } <select_statement> ]

<table_list> ::=
     <table_name> [ { INNER | LEFT [OUTER] } JOIN <table_name>
     ON <expression> ]  [, ...]

<select_expression> ::=
      { <expression> | COUNT(*) | {COUNT | MIN | MAX | SUM | AVG}
       (<expression>) <column_alias>}

<order_expression> ::=
      { <column_number> | <column_alias> | <select_expression> } [ ASC | DESC ]
```

B.5.1.9 SET AUTOCOMMIT

Switches on or off the connection's auto-commit mode. If switched on, then all statements will be committed as individual transactions. Otherwise, the statements are grouped into transactions that are terminated by either COMMIT or ROLLBACK. By default, new connections are in auto-commit mode.

```
SET AUTOCOMMIT { TRUE | FALSE }
```

B.5.1.10 UPDATE

Modifies data of a table in the database.

```
UPDATE table SET column = <expression> [, ...] [WHERE <expression>]
```

B.5.1.11 Expressions, Condition and Values

```
<expression> ::=
     [NOT] <condition> [ { OR | AND } <condition> ]

<condition> ::=
     { <value> [ || <value> ]
      | <value> { = | < | <= | > | >= | <> | != | IS [NOT] } <value>
      | EXISTS(<select_statement>)
      | <value> BETWEEN <value> AND <value>
      | <value> [NOT] IN ( {<value> [, ...] | selectStatement } )
      | <value> [NOT] LIKE <value> [ESCAPE] value }

<value> ::=
      [ + | - ] { term [ { + | - | * | / } term ]
      | ( condition )
      | function ( [parameter] [,...] )
      | selectStatement_giving_one_value

<term> ::=
      { 'string' | number | floatingpoint | [table.]column | TRUE | FALSE | NULL }
```

<string> ::=

- Starts and ends with a single '. In a string started with ' use '' to create a '.
- LIKE uses '%' to match any (including 0) number of characters, and '_' to match exactly one character. To search for '%' itself, '\%' must be used, for '_' use '_'; or any other escaping character may be set using the ESCAPE clause.

<name> ::=

- A name starts with a letter and is followed by any number of letters or digits. Lowercase is changed to uppercase except for strings and quoted identifiers. Names are not case-sensitive.
- Quoted identifiers can be used as names (for example for tables or columns). Quoted identifiers start and end with ". In a quoted identifier use "" to create a ". With quoted identifiers it is possible to create mixed case table and column names. Example: CREATE TABLE "Address" ("Nr" INTEGER, "Name" VARCHAR); SELECT * FROM "Address". Quoted identifiers are not strings.

<values> ::=

- A 'date' value starts and ends with ', the format is yyyy-mm-dd (see java.sql.Date).
- A 'time' value starts and ends with ', the format is hh:mm:ss (see java.sql.Time).
- Binary data starts and ends with ', the format is hexadecimal. '0004ff' for example is 3 bytes, first 0, second 4 and last 255 (0xff).

B.5.2 SQL FUNCTIONS

Table B–4 lists the numerical functions.

Table B–4 Numerical Functions

Function	Description
ABS(d)	returns the absolute value of a double value
ACOS(d)	returns the arc cosine of an angle
ASIN(d)	returns the arc sine of an angle
ATAN(d)	returns the arc tangent of an angle
ATAN2(a,b)	returns the tangent of a/b
CEILING(d)	returns the smallest integer that is not less than d
COS(d)	returns the cosine of an angle
COT(d)	returns the cotangent of an angle
DEGREES(d)	converts radians to degrees
EXP(d)	returns e (2.718...) raised to the power of d
FLOOR(d)	returns the largest integer that is not greater than d
LOG(d)	returns the natural logarithm (base e)
LOG10(d)	returns the logarithm (base 10)
MOD(a,b)	returns a modulo b
PI()	returns pi (3.1415...)
POWER(a,b)	returns a raised to the power of b
RADIANS(d)	converts degrees to radians
RAND()	returns a random number x bigger or equal to 0.0 and smaller than 1.0
ROUND(a,b)	rounds a to b digits after the decimal point
SIGN(d)	returns -1 if d is smaller than 0, 0 if d==0 and 1 if d is bigger than 0
SIN(d)	returns the sine of an angle
SQRT(d)	returns the square root
TAN(d)	returns the trigonometric tangent of an angle
TRUNCATE(a,b)	truncates a to b digits after the decimal point
BITAND(a,b)	return a & b
BITOR(a,b)	returns a | b

Table B–5 lists the string functions.

Table B–5 String Functions

Function	Description
ASCII(s)	returns the ASCII code of the leftmost character of s
CHAR(c)	returns a character that has the ASCII code c
CONCAT(str1,str2)	returns str1 + str2
DIFFERENCE(s1,s2)	returns the difference between the sound of s1 and s2
INSERT(s,start,len,s2)	returns a string where len number of characters beginning at start has been replaced by s2

Table B–5 (Cont.) String Functions

Function	Description
LCASE(s)	converts s to lower case
LEFT(s,count)	returns the leftmost count of characters of s
LENGTH(s)	returns the number of characters in s
LOCATE(search,s,[start])	returns the first index (1=left, 0=not found) where search is found in s, starting at start
LTRIM(s)	removes all leading blanks in s
REPEAT(s,count)	returns s repeated count times
REPLACE(s,replace,s2)	replaces all occurrences of replace in s with s2
RIGHT(s,count)	returns the rightmost count of characters of s
RTRIM(s)	removes all trailing blanks
SOUNDEX(s)	returns a four character code representing the sound of s
SPACE(count)	returns a string consisting of count spaces
SUBSTRING(s,start[,len])	returns the substring starting at start (1=left) with length len
UCASE(s)	converts s to upper case
LOWER(s)	converts s to lower case
UPPER(s)	converts s to upper case

Table B–6 lists the date/time functions.

Note that a *date* value starts and ends with a single quote ('), the format is `yyyy-mm-dd` (see java.sql.Date). A *time* value starts and ends with a single quote ('), the format is `hh:mm:ss` (see java.sql.Time).

Table B–6 Date/Time Functions

Function	Description
CURDATE()	returns the current date
CURTIME()	returns the current time
DAYNAME(date)	returns the name of the day
DAYOFMONTH(date)	returns the day of the month (1-31)
DAYOFWEEK(date)	returns the day of the week (1 means Sunday)
DAYOFYEAR(date)	returns the day of the year (1-366)
HOUR(time)	return the hour (0-23)
MINUTE(time)	returns the minute (0-59)
MONTH(date)	returns the month (1-12)
MONTHNAME(date)	returns the name of the month
NOW()	returns the current date and time as a timestamp
QUARTER(date)	returns the quarter (1-4)
SECOND(time)	returns the second (0-59)
WEEK(date)	returns the week of this year (1-53)
YEAR(date)	returns the year

Table B–7 lists the system functions.

Table B–7 System Functions

Function	Description
IFNULL(exp,value)	if exp is null, value is returned else exp
CASEWHEN(exp,v2,v2)	if exp is true, v1 is returned, else v2
CONVERT(term,type)	converts exp to another data type
CAST(term AS type)	converts exp to another data type

B.6 JDBC API Implemented Features

Table B–8 lists the JDBC API features that are implemented in the Oracle Data Integrator Driver for XML:

Table B–8 JDBC API Features

Feature Groups	JDBC Version	Support
Batch Update	2.0 Core	Yes
Blob/Clob	2.0 Core	Yes
JNDI DataSources	2.0 Optional	Yes
Failover support	-	Yes
Transaction SavePoints	3.0	Yes
Unicode support	-	No
Distributed Transaction	2.0 Optional	No
Connection Pooling	2.0 Optional	No
Cluster support	-	No

Table B–9 lists JDBC Java classes.

Table B–9 JDBC Java Classes

JDBC Class	JDBC Version	Support
Array	2.0 Core	No
Blob	2.0 Core	Yes
CallableStatement	1.0	Yes
Clob	2.0 Core	Yes
Connection	1.0	Yes
ConnectionPoolDataSource	2.0 Optional	No
DatabaseMetaData	1.0	Yes
DataSource	2.0 Optional	No
Driver	1.0	Yes
Ref	2.0 Core	No
ResultSet	1.0	Yes
ResultSetMetaData	1.0	Yes

Table B–9 (Cont.) JDBC Java Classes

JDBC Class	JDBC Version	Support
RowSet	2.0 Optional	No
Statement	1.0	Yes
Struct	2.0 Core	No
PreparedStatement	1.0	Yes
XAConnection	2.0 Optional	No
XADataSource	2.0 Optional	No

B.7 XML Schema Supported Features

The driver supports part of the XML Schema (XSD) specification. Supported elements are listed in this section.

For more information on the XML Schema specification, see the W3C specification at `http://www.w3.org/TR/xmlschema-1/`.

This section contains the following topics:

- Datatypes
- Supported Elements
- Unsupported Features

B.7.1 Datatypes

The following datatypes are supported:

- These datatypes are converted to String columns: string, normalizedString, token, nmtoken, nmtokens, anyUri, id, idref, date, datetime, time, hexBinary
- These datatypes are converted to Integer columns: int, positiveInteger, negativeInteger, nonNegativeInteger, onPositiveInteger, long, unsignedLong, unsignedInt, short, unsignedShort, byte, unsignedByte, boolean (Boolean are converted to a numeric column with 0 or 1, but they can take "true" or "false" values from the input files)
- These datatypes are converted to Decimal (with 2 decimal places) columns: decimal, float, double

B.7.2 Supported Elements

This section lists all schema elements. Supported syntax elements are shown in **bold**. Unsupported syntax elements are shown in regular font. They are ignored by the driver.

This section details the following schema elements:

- All
- Attribute
- AttributeGroup
- Choice
- ComplexContent
- ComplexType

- Element
- Extension
- Group
- Import
- Include
- List
- Restriction
- Schema
- Sequence
- SimpleContent
- SimpleType

> **Note:** XML files generated or updated using the XML driver should ideally be validated against their corresponding XSD files using the VALIDATE command after generation.

B.7.2.1 All

This element specifies that child elements can appear in any order and that each child element can occur zero or one time.

Note that child elements mandatory properties (minOccurs=1) are not managed by the driver. This should be handled by checks on the data, and by validating the XML contents against the XSD.

```
<all
    id=ID
    maxOccurs=1
    minOccurs=0|1
    any attributes
>
(annotation?,element*)
</all>
```

B.7.2.2 Attribute

This element defines an attribute.

```
<attribute
    default=string
    id=ID
    name=NCName
    type=QName
    use=optional|prohibited|required
    ref=QName
    fixed=string
    form=qualified|unqualified
    any attributes
>
(annotation?,(simpleType?))
</attribute>
```

Note that the `use` attribute of this element defines the column mapped by the driver for the attribute as mandatory or not.

B.7.2.3 AttributeGroup

This element defines a set of attributes.

```
<attributeGroup
    id=ID
    name=NCName
    ref=QName
    any attributes
>
(annotation?),((attribute|attributeGroup)*,anyAttribute?))
</attributeGroup>
```

B.7.2.4 Choice

This element allows one and only of the elements to be present within the containing element.

```
<choice
    id=ID
    maxOccurs=nonNegativeInteger|unbounded
    minOccurs=nonNegativeInteger
    any attributes
>
(annotation?,(element|group|choice|sequence|any)*)
</choice>
```

Note that the child element's unique nature are not managed by the driver. This should be handled by checks on the data, and by validating the XML contents against the XSD.

B.7.2.5 ComplexContent

This element defines extensions or restrictions on a complex type.

```
<complexContent
    id=ID
    mixed=true|false
    any attributes
>
(annotation?,(restriction|extension))
</complexContent>
```

B.7.2.6 ComplexType

This element defines a complex type.

```
<complexType
    name=NCName
    id=ID
    abstract=true|false
    mixed=true|false
    block=(#all|list of (extension|restriction))
    final=(#all|list of (extension|restriction))
    any attributes
>
(annotation?,(simpleContent|complexContent|((group|all|choice|sequence)?,((attribu
te|attributeGroup)*,anyAttribute?))))
</complexType>
```

B.7.2.7 Element

This element defines an element of the XML file.

```
<element
    name=NCName
    maxOccurs=nonNegativeInteger|unbounded
    minOccurs=nonNegativeInteger
    type=QName
    id=ID
    ref=QName
    substitutionGroup=QName
    default=string
    fixed=string
    form=qualified|unqualified
    nillable=true|false
    abstract=true|false
    block=(#all|list of (extension|restriction))
    final=(#all|list of (extension|restriction))
    any attributes
>
annotation?,((simpleType|complexType)?,(unique|key|keyref)*))
</element>
```

> **Note:** The maxOccurs and minOccurs attributes of the element are used in the XML-to-SQL mapping. If a child element is of a simple type and is monovalued (one occurrence only), then this element is mapped to a simple column in the table corresponding to its parent element. Otherwise, a table linked to the parent element's table is created.
>
> Note that if no reference to either minOccurs or maxOccurs is mentioned in an element then the element is consider as monovalued and is transformed to a column. This behavior can be changed using the `useImplicitMaxValue` URL property. When this property is set to yes, an elements for which maxOccurs is not specified in the XSD is considered as multivalued (maxOccurs ="unbounded").

> **Note:** Using different sub-elements with the same name but with different types is not supported by XML driver. An XSD with such a structure will not be processed correctly.

B.7.2.8 Extension

This element extends an existing simpleType or complexType element

```
<extension
    id=ID
    base=QName
    any attributes
>
(annotation?,((group|all|choice|sequence)?,((attribute|attributeGroup)*,anyAttribu
te?)))
</extension>
```

B.7.2.9 Group

The group element is used to define a group of elements to be used in complex type definitions.

```
<group
    id=ID
    name=NCName
    ref=QName
    maxOccurs=nonNegativeInteger|unbounded
    minOccurs=nonNegativeInteger
    any attributes
>
(annotation?,(all|choice|sequence)?)
</group>
```

B.7.2.10 Import

This element is used to add multiple schemas with different target namespace to a document.

```
<import
    id=ID
    namespace=anyURI
    schemaLocation=anyURI
    any attributes
>
(annotation?)
</import>
```

B.7.2.11 Include

This element is used to add multiple schemas with the same target namespace to a document.

```
<include
    id=ID
    schemaLocation=anyURI
    any attributes
>
(annotation?)
</include>
```

B.7.2.12 List

This element defines a simple type element as a list of values of a specified data type.

```
<list
    id=ID
    itemType=QName
    any attributes
>
(annotation?,(simpleType?))
</list>
```

B.7.2.13 Restriction

This element defines restrictions on a simpleType, simpleContent, or a complexContent.

```
<restriction
     id=ID
     base=QName
```

```
     any attributes
>
Content for simpleType:
(annotation?,(simpleType?,(minExclusive|minInclusive|maxExclusive|maxInclusive|
totalDigits|fractionDigits|length|minLength|maxLength|enumeration|whiteSpace|
pattern)*))
Content for simpleContent:
(annotation?,(simpleType?,(minExclusive|minInclusive|maxExclusive|maxInclusive|
totalDigits|fractionDigits|length|minLength|maxLength|enumeration|whiteSpace|
pattern)*)?, ((attribute|attributeGroup)*,anyAttribute?))
Content for complexContent:
(annotation?,(group|all|choice|sequence)?,
((attribute|attributeGroup)*,anyAttribute?))
</restriction>
```

B.7.2.14 Schema

This element defines the root element of a schema.

```
<schema
     id=ID
     attributeFormDefault=qualified|unqualified
     elementFormDefault=qualified|unqualified
     blockDefault=(#all|list of (extension|restriction|substitution))
     finalDefault=(#all|list of (extension|restriction|list|union))
     targetNamespace=anyURI
     version=token
     xmlns=anyURI
     any attributes
>
((include|import|redefine|annotation)*,(((simpleType|complexType|group|
attributeGroup)|element|attribute|notation),annotation*)*)
</schema>
```

B.7.2.15 Sequence

This element specifies that the child elements must appear in a sequence. Each child element can occur 0 or more times.

```
<sequence
     id=ID
     maxOccurs=nonNegativeInteger|unbounded
     minOccurs=nonNegativeInteger
     any attributes
>
(annotation?,(element|group|choice|sequence|any)*)
</sequence>
```

Note the following:

- The Sequence order is not managed by the driver. The sequence order should be handled by loading the xxx_ORDER column generated by the driver.
- The maxOccurs and minOccurs attributes are not managed by the driver. This should be handled by checks on the data, and by validating the XML contents against the XSD.

B.7.2.16 SimpleContent

This element contains extensions or restrictions on a text-only complex type or on a simple type as content.

```
<simpleContent
     id=ID
     any attributes
>
(annotation?,(restriction|extension))
</simpleContent>
```

B.7.2.17 SimpleType

This element defines a simple type element.

```
<simpleType
     name=NCName
     id=ID
     any attributes
>
(annotation?,(restriction|list|union))
</simpleType>
```

B.7.3 Unsupported Features

The following elements and features are not supported or implemented by the XML driver.

B.7.3.1 Unsupported Elements

The following schema elements are not supported by the XML driver.

- **Annotation**: The annotation is an element for defining comments and inline documentation. This element and all its child elements (appInfo, documentation) are ignored.
- **Any/anyAttribute**: The any and anyAttribute elements enables you to extend the XML document with any element or attribute, even if it is not specified by the schema. These elements are not supported.
- **Key/keyRef/Unique**: These elements allow the definition of constraints in the schema. These elements and their child elements (selector, field) are ignored.
- **Union**: The union element defines a list of values from specified simple data types. This element is not supported.
- **Redefine**: The redefine element redefines simple and complex types, groups, and attribute groups from an external schema. This element is not supported.

> **WARNING: Elements and attributes allowed in an XML file due to an Any or AnyAttribute clause in the XSD may cause errors when the file is loaded.**

B.7.3.2 Unsupported Features

Multipass parsing is not implemented in the ODI XML driver.