

Oracle® Fusion Middleware

Connectivity and Knowledge Modules Guide for Oracle Data Integrator



12c (12.2.1.3.0)

E96498-05

March 2019

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Fusion Middleware Connectivity and Knowledge Modules Guide for Oracle Data Integrator, 12c (12.2.1.3.0)

E96498-05

Copyright © 2010, 2019, Oracle and/or its affiliates. All rights reserved.

Primary Author: Oracle Corporation

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

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

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

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

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

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

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

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

Contents

Preface

Audience	xxvi
Documentation Accessibility	xxvi
Related Documents	xxvi
Conventions	xxvii

1 Introduction

Terminology	1-1
Using This Guide	1-2
Accessing Data in the Relational Structure	1-2
Accessing Data in the Schemas	1-2

Part I Databases, Files, and XML

2 Oracle Database

Introduction	2-1
Concepts	2-1
Knowledge Modules	2-1
Installation and Configuration	2-4
System Requirements and Certifications	2-4
Technology Specific Requirements	2-4
Using the SQL*Loader Utility	2-4
Using External Tables	2-4
Using Oracle Wallet	2-5
Connectivity Requirements	2-6
Setting up the Topology	2-7
Creating an Oracle Data Server	2-7
Creation of the Data Server	2-7
Creating an Oracle Physical Schema	2-9
Setting Up an Integration Project	2-9

Creating and Reverse-Engineering an Oracle Model	2-10
Create an Oracle Model	2-10
Reverse-engineer an Oracle Model	2-10
Setting up Changed Data Capture	2-11
Setting up Data Quality	2-12
Designing a Mapping	2-12
Loading Data from and to Oracle	2-13
Loading Data from Oracle	2-13
Loading Data to Oracle	2-13
Integrating Data in Oracle	2-14
Designing an ETL-Style Mapping	2-15
Troubleshooting	2-18
Troubleshooting Oracle Database Errors	2-18
Common Problems and Solutions	2-19

3 Oracle Autonomous Data Warehouse Cloud

Introduction	3-1
Concepts	3-1
Knowledge Modules	3-1
Prerequisites	3-5
Setting up the Topology	3-7
Creating an Oracle Data Server	3-7
Creating an Oracle Physical Schema	3-10
Creating and Reverse-Engineering an Oracle Model	3-10
Create an Oracle Model	3-10
Reverse Engineer an Oracle Model	3-10
Designing a Mapping	3-10
Loading data	3-11
Loading Data using Oracle KMs	3-11
Loading Data using SQL* Loader KMs	3-11
Loading Data directly into ADWC	3-12
Loading Oracle Object Storage files into ADWC	3-36
Extracting data	3-42
Best Practices for Working with ADWC	3-43
Caching Oracle Sequences in ADWC	3-43

4 Oracle Autonomous Transaction Processing

Introduction	4-1
Concepts	4-1

Knowledge Modules	4-2
Prerequisites	4-6
Setting up the Topology	4-8
Creating an Oracle Data Server	4-9
Creating an Oracle Physical Schema	4-11
Creating and Reverse-Engineering an Oracle Model	4-11
Create an Oracle Model	4-11
Reverse Engineer an Oracle Model	4-11
Designing a Mapping	4-12
Loading data	4-12
Loading Data using Oracle KMs	4-12
Loading Data using SQL* Loader KMs	4-13
Loading Data directly into ATP	4-13
Loading Oracle Object Storage files into ATP	4-37
Extracting data	4-44
Best Practices for Working with ATP	4-44
Caching Oracle Sequences in ATP	4-44

5 Files

Introduction	5-1
Concepts	5-1
Knowledge Modules	5-1
Installation and Configuration	5-2
System Requirements and Certifications	5-2
Technology Specific Requirements	5-2
Connectivity Requirements	5-5
Setting up the Topology	5-5
Creating a File Data Server	5-5
Creation of the Data Server	5-6
Creating a File Physical Schema	5-8
Setting Up an Integration Project	5-8
Creating and Reverse-Engineering a File Model	5-9
Create a File Model	5-9
Reverse-engineer a File Model	5-9
Delimited Files Reverse-Engineering	5-9
Fixed Files Reverse-engineering using the Wizard	5-10
COBOL Copybook reverse-engineering	5-11
Customized Reverse-Engineering	5-12
Designing a Mapping	5-14
Loading Data From Files	5-14

Integrating Data in Files	5-16
IKM SQL to File Append	5-16
IKM File to File (Java)	5-16

6 Generic SQL

Introduction	6-1
Concepts	6-1
Knowledge Modules	6-2
Installation and Configuration	6-4
System Requirements and Certifications	6-5
Technology-Specific Requirements	6-5
Connectivity Requirements	6-5
Setting up the Topology	6-5
Creating a Data Server	6-5
Creating a Physical Schema	6-6
Setting up an Integration Project	6-6
Creating and Reverse-Engineering a Model	6-6
Create a Data Model	6-6
Reverse-engineer a Data Model	6-6
Setting up Changed Data Capture	6-7
Setting up Data Quality	6-7
Designing a Mapping	6-7
Loading Data From and to an ANSI SQL-92 Compliant Technology	6-7
Loading Data from an ANSI SQL-92 Compliant Technology	6-7
Loading Data to an ANSI SQL-92 Compliant Technology	6-8
Integrating Data in an ANSI SQL-92 Compliant Technology	6-8
Designing an ETL-Style Mapping	6-9

7 XML Files

Introduction	7-1
Concepts	7-1
Pre/Post Processing Support for XML Driver	7-1
Knowledge Modules	7-2
Installation and Configuration	7-2
System Requirements	7-2
Technologic Specific Requirements	7-2
Connectivity Requirements	7-5
Setting up the Topology	7-5
Creating an XML Data Server	7-6

Creation of the Data Server	7-6
Creating a Physical Schema for XML	7-7
Setting Up an Integration Project	7-7
Creating and Reverse-Engineering a XML File	7-7
Create an XML Model	7-8
Reverse-Engineering an XML Model	7-8
Designing a Mapping	7-8
Notes about XML Mappings	7-8
Targeting an XML Structure	7-9
Synchronizing XML File and Schema	7-9
Handling Large XML Files	7-9
Loading Data from and to XML	7-10
Loading Data from an XML Schema	7-10
Loading Data to an XML Schema	7-10
Integrating Data in XML	7-11
Troubleshooting	7-11
Detect the Errors Coming from XML	7-11
Common Errors	7-12

8 Complex Files

Introduction	8-1
Concepts	8-1
Pre/Post Processing Support for Complex File Driver	8-2
Knowledge Modules	8-2
Installation and Configuration	8-2
System Requirements	8-3
Technology Specific Requirements	8-3
Connectivity Requirements	8-3
Building a Native Schema Description File Using the Native Format Builder	8-3
Setting up the Topology	8-4
Creating a Complex File Data Server	8-4
Creation of the Data Server	8-4
Creating a Complex File Physical Schema	8-5
Setting Up an Integration Project	8-5
Creating and Reverse-Engineering a Complex File Model	8-5
Create a Complex File Model	8-6
Reverse-engineer a Complex File Model	8-6
Designing a Mapping	8-6

9 Microsoft SQL Server

Introduction	9-1
Concepts	9-1
Knowledge Modules	9-1
Installation and Configuration	9-2
System Requirements and Certifications	9-2
Technology Specific Requirements	9-3
Using the BULK INSERT Command	9-3
Using the BCP Command	9-3
Using Linked Servers	9-3
Connectivity Requirements	9-4
Setting up the Topology	9-4
Creating a Microsoft SQL Server Data Server	9-4
Creation of the Data Server	9-4
Creating a Microsoft SQL Server Physical Schema	9-5
Setting Up an Integration Project	9-5
Creating and Reverse-Engineering a Microsoft SQL Server Model	9-5
Create a Microsoft SQL Server Model	9-6
Reverse-engineer a Microsoft SQL Server Model	9-6
Setting up Changed Data Capture	9-7
Setting up Data Quality	9-7
Designing a Mapping	9-7
Loading Data from and to Microsoft SQL Server	9-7
Loading Data from Microsoft SQL Server	9-8
Loading Data to Microsoft SQL Server	9-8
Integrating Data in Microsoft SQL Server	9-9

10 Microsoft Excel

Introduction	10-1
Concepts	10-1
Knowledge Modules	10-1
Installation and Configuration	10-2
System Requirements and Certifications	10-2
Specific Requirements	10-2
Setting up the Topology	10-3
Creating a Microsoft Excel Data Server	10-3
Creating a Microsoft Excel Physical Schema	10-3
Setting Up an Integration Project	10-3
Creating and Reverse-Engineering a Microsoft Excel Model	10-4
Create a Microsoft Excel Model	10-4

Reverse-engineer a Microsoft Excel Model	10-4
Designing a Mapping	10-4
Loading Data From and to Microsoft Excel	10-4
Loading Data from Microsoft Excel	10-4
Loading Data to Microsoft Excel	10-5
Integrating Data in Microsoft Excel	10-5

11 Microsoft Access

Introduction	11-1
Concepts	11-1
Knowledge Modules	11-1
Specific Requirements	11-2

12 Netezza

Introduction	12-1
Concepts	12-1
Knowledge Modules	12-1
Installation and Configuration	12-2
System Requirements and Certifications	12-2
Technology Specific Requirements	12-2
Connectivity Requirements	12-3
Setting up the Topology	12-3
Creating a Netezza Data Server	12-3
Creation of the Data Server	12-3
Creating a Netezza Physical Schema	12-4
Setting Up an Integration Project	12-4
Creating and Reverse-Engineering a Netezza Model	12-4
Create a Netezza Model	12-4
Reverse-engineer a Netezza Model	12-5
Setting up Data Quality	12-5
Designing a Mapping	12-5
Loading Data from and to Netezza	12-5
Loading Data from Netezza	12-5
Loading Data to Netezza	12-6
Integrating Data in Netezza	12-6

13 Teradata

Introduction	13-1
Concepts	13-1

Knowledge Modules	13-1
Installation and Configuration	13-2
System Requirements and Certifications	13-2
Technology Specific Requirements	13-3
Connectivity Requirements	13-3
Setting up the Topology	13-3
Creating a Teradata Data Server	13-4
Creation of the Data Server	13-4
Creating a Teradata Physical Schema	13-4
Setting Up an Integration Project	13-4
Creating and Reverse-Engineering a Teradata Model	13-5
Create a Teradata Model	13-5
Reverse-engineer a Teradata Model	13-5
Setting up Data Quality	13-6
Designing a Mapping	13-6
Loading Data from and to Teradata	13-7
Loading Data from Teradata	13-7
Loading Data to Teradata	13-7
Integrating Data in Teradata	13-8
Designing an ETL-Style Mapping	13-12
KM Optimizations for Teradata	13-15
Primary Indexes and Statistics	13-16
Support for Teradata Utilities	13-16
Support for Named Pipes	13-17
Optimized Management of Temporary Tables	13-17

14 Hypersonic SQL

Introduction	14-1
Concepts	14-1
Knowledge Modules	14-1
Installation and Configuration	14-2
System Requirements and Certifications	14-2
Technology Specific Requirements	14-2
Connectivity Requirements	14-2
Setting up the Topology	14-3
Creating a Hypersonic SQL Data Server	14-3
Creating a Hypersonic SQL Physical Schema	14-3
Setting Up an Integration Project	14-3
Creating and Reverse-Engineering a Hypersonic SQL Model	14-4
Create a Hypersonic SQL Model	14-4

Reverse-engineer a Hypersonic SQL Model	14-4
Setting up Changed Data Capture	14-4
Setting up Data Quality	14-5
Designing a Mapping	14-5

15 IBM Informix

Introduction	15-1
Concepts	15-1
Knowledge Modules	15-1
Specific Requirements	15-3

16 IBM DB2 for iSeries

Introduction	16-1
Concepts	16-1
Knowledge Modules	16-1
Installation and Configuration	16-2
System Requirements and Certifications	16-2
Technology Specific Requirements	16-2
Setting up the Topology	16-3
Creating a DB2/400 Data Server	16-3
Creation of the Data Server	16-3
Creating a DB2/400 Physical Schema	16-3
Setting Up an Integration Project	16-4
Creating and Reverse-Engineering an IBM DB2/400 Model	16-4
Create an IBM DB2/400 Model	16-4
Reverse-engineer an IBM DB2/400 Model	16-4
Setting up Changed Data Capture	16-5
Setting up Trigger-Based CDC	16-5
Setting up Data Quality	16-5
Designing a Mapping	16-5
Loading Data from and to IBM DB2 for iSeries	16-6
Integrating Data in IBM DB2 for iSeries	16-6
Specific Considerations with DB2 for iSeries	16-6
Alternative Connectivity Methods for iSeries	16-6
Using Client Access	16-6
Using the IBM JT/400 and Native Drivers	16-7
Troubleshooting	16-7
Troubleshooting Error messages	16-7
Common Problems and Solutions	16-8

17 IBM DB2 UDB

Introduction	17-1
Concepts	17-1
Knowledge Modules	17-1
Specific Requirements	17-3

18 Salesforce.com

Introduction	18-1
Concepts	18-1
Knowledge Modules	18-1
Installation and Configuration	18-1
System Requirements and Certifications	18-2
Technology Specific Requirements	18-2
Connectivity Requirements	18-2
Setting up the Topology	18-2
Creating a Salesforce.com Data Server	18-2
Creating a Physical Schema for Salesforce.com Data Server	18-3
Setting Up an Integration Project	18-3
Creating and Reverse-Engineering a Salesforce.com Model	18-4
Create a Salesforce.com Model	18-4
Reverse-engineer a Salesforce.com Model	18-4
Designing a Mapping	18-4
Loading Data from and to Salesforce.com	18-4
Loading Data from Salesforce.com	18-5
Loading Data to Salesforce.com	18-5
Integrating Data in Salesforce.com	18-5

19 Sybase IQ

Introduction	19-1
Concepts	19-1
Knowledge Modules	19-1
Specific Requirements	19-3

Part II Business Intelligence

20 Oracle Business Intelligence Enterprise Edition

Introduction	20-1
Concepts	20-1
Knowledge Modules	20-1
Installation and Configuration	20-2
System Requirements and Certifications	20-2
Technology Specific Requirements	20-2
Connectivity Requirements	20-2
Setting up the Topology	20-3
Creating an Oracle BI Data Server	20-3
Creation of the Data Server	20-3
Creating an Oracle BI Physical Schema	20-4
Setting Up an Integration Project	20-4
Creating and Reverse-Engineering an Oracle BI Model	20-4
Create an Oracle BI Model	20-4
Reverse-engineer an Oracle BI Model	20-4
Setting up Data Quality	20-5
Designing a Mapping	20-5
Loading Data from and to Oracle BI	20-5
Loading Data from Oracle BI	20-5
Loading Data to Oracle BI	20-6
Integrating Data in Oracle BI	20-6

21 Oracle Business Intelligence Cloud Service

Introduction	21-1
Setting up the Topology	21-2
Creating an Oracle BICS Data Server	21-2
Creating an Oracle BICS Physical Schema	21-2
Reverse Engineering a BICS Model	21-3
Designing a Mapping	21-3

22 Oracle Hyperion Planning

Introduction	22-1
Integration Process	22-1
Knowledge Modules	22-1
Installation and Configuration	22-2
System Requirements and Certifications	22-2
Technology Specific Requirements	22-2
Connectivity Requirements	22-2

Setting up Hyperion Planning Adapter	22-2
Setting up Adapter for ODI Studio	22-3
Setting up Adapter for ODI Standalone Agent	22-3
Setting up the Topology	22-3
Creating an Hyperion Planning Data Server	22-3
Creating an Hyperion Planning Physical Schema	22-4
Creating and Reverse-Engineering a Planning Model	22-4
Create a Planning Model	22-4
Reverse-engineer a Planning Model	22-4
Designing a Mapping	22-4
Loading Metadata	22-5
Loading Data	22-6
Load Options	22-7
Datastore Tables and Data Load Columns	22-8
Accounts	22-8
Employee	22-15
Entities	22-21
User-Defined Dimensions	22-26
Attribute Dimensions	22-32
UDA	22-33
Data Load Columns	22-34

23 Oracle Hyperion Essbase

Introduction	23-1
Integration Process	23-1
Knowledge Modules	23-2
Installation and Configuration	23-2
System Requirements and Certifications	23-2
Technology Specific Requirements	23-2
Connectivity Requirements	23-2
Setting up Hyperion Essbase Adapter	23-3
Setting up Adapter for ODI Studio	23-3
Setting up Adapter for ODI Standalone Agent	23-3
Setting up the Topology	23-3
Creating an Hyperion Essbase Data Server	23-3
Creating an Hyperion Essbase Physical Schema	23-4
Creating and Reverse-Engineering an Essbase Model	23-4
Create an Essbase Model	23-4
Reverse-engineer an Essbase Model	23-4
Designing a Mapping	23-6

Loading Metadata	23-6
Loading Data	23-8
Extracting Data	23-12
Data Extraction Methods for Essbase	23-12
Extracting Essbase Data	23-13
Extracting Members from Metadata	23-15

Part III Other Technologies

24 JMS

Introduction	24-1
Concepts	24-1
JMS Message Structure	24-1
Using a JMS Destination	24-2
Knowledge Modules	24-3
Installation and Configuration	24-3
System Requirements and Certifications	24-4
Technology Specific Requirements	24-4
Connectivity Requirements	24-4
Setting up the Topology	24-4
Creating a JMS Data Server	24-4
Creation of the Data Server	24-4
Creating a JMS Physical Schema	24-5
Setting Up an Integration Project	24-5
Creating and Defining a JMS Model	24-5
Create a JMS Model	24-6
Defining the JMS Datastores	24-6
Designing a Mapping	24-7
Loading Data from a JMS Source	24-7
Integrating Data in a JMS Target	24-8
JMS Standard Properties	24-9
Using JMS Properties	24-11
Declaring JMS Properties	24-11
Filtering on the Router	24-11
Filtering on the Client	24-12
Using Property Values as Source Data	24-12
Setting Properties when Sending a Message	24-12

25 JMS XML

Introduction	25-1
Concepts	25-1
JMS Message Structure	25-1
Using a JMS Destination	25-1
Knowledge Modules	25-3
Installation and Configuration	25-3
System Requirements and Certifications	25-3
Technology Specific Requirements	25-3
Connectivity Requirements	25-4
Setting up the Topology	25-4
Creating a JMS XML Data Server	25-4
Creation of the Data Server	25-5
Creating a JMS XML Physical Schema	25-6
Setting Up an Integration Project	25-7
Creating and Reverse-Engineering a JMS XML Model	25-7
Create a JMS XML Model	25-7
Reverse-Engineering a JMS XML Model	25-8
Designing a Mapping	25-8
Loading Data from a JMS XML Source	25-8
Integrating Data in a JMS XML Target	25-9

26 LDAP Directories

Introduction	26-1
Concepts	26-1
Knowledge Modules	26-2
Installation and Configuration	26-2
System Requirements	26-2
Technologic Specific Requirements	26-2
Connectivity Requirements	26-2
Setting up the Topology	26-3
Creating an LDAP Data Server	26-3
Creation of the Data Server	26-3
Creating a Physical Schema for LDAP	26-4
Setting Up an Integration Project	26-4
Creating and Reverse-Engineering an LDAP Directory	26-4
Create an LDAP Model	26-5
Reverse-Engineering an LDAP Model	26-5
Designing a Mapping	26-5
Loading Data from and to LDAP	26-5

Loading Data from an LDAP Directory	26-6
Loading Data to an LDAP Directory	26-6
Integrating Data in an LDAP Directory	26-6
Troubleshooting	26-6

27 Oracle TimesTen In-Memory Database

Introduction	27-1
Concepts	27-1
Knowledge Modules	27-2
Installation and Configuration	27-2
System Requirements and Certifications	27-2
Technology Specific Requirements	27-2
Connectivity Requirements	27-3
Setting up the Topology	27-3
Creating a TimesTen Data Server	27-3
Creation of the Data Server	27-4
Creating a TimesTen Physical Schema	27-4
Creating and Reverse-Engineering a TimesTen Model	27-4
Create a TimesTen Model	27-4
Reverse-engineer a TimesTen Model	27-4
Setting up Data Quality	27-5
Designing a Mapping	27-5
Loading Data from and to TimesTen	27-5
Loading Data from TimesTen	27-5
Loading Data to TimesTen	27-6
Integrating Data in TimesTen	27-6
Setting Up an Integration Project	27-6

28 Oracle GoldenGate

Introduction	28-1
Overview of the GoldenGate CDC Process	28-1
Knowledge Modules	28-2
Installation and Configuration	28-3
System Requirements and Certifications	28-3
Technology Specific Requirements	28-4
Connectivity Requirements	28-4
Working with the Oracle GoldenGate JKMs	28-4
Define the Topology	28-5
Define the Source Data Server	28-5

Create the Source Physical Schema	28-5
Define the Staging Server	28-5
Create the Staging Physical Schema	28-6
Define the Oracle GoldenGate Data Servers	28-6
Create the Oracle GoldenGate Physical Schemas	28-6
Create the Oracle GoldenGate Logical Schemas	28-8
Create the Replicated Tables	28-9
Set Up an Integration Project	28-9
Configure CDC for the Source Datastores	28-10
Create Oracle GoldenGate Physical Schemas from the model	28-12
Configure and Start Oracle GoldenGate Processes (Offline mode only)	28-13
Design Mappings Using Replicated Data	28-14
Advanced Configuration	28-14
Initial Load Method	28-14
Tuning Replication Performances	28-14
One Source Multiple Staging Configuration (Offline mode only)	28-15
Integrated Capture	28-15
Integrated Capture Deployment Options	28-16
Deciding Which Apply Method to Use	28-17
Nonintegrated Replicat	28-17
Using Different Capture and Apply Modes Together	28-21
Switching to Different Process Mode	28-22
Upgrading GoldenGate Classic Extract to Integrated	28-22

29 Oracle SOA Suite Cross References

Introduction	29-1
Concepts	29-1
General Principles	29-1
Cross Reference Table Structures	29-2
Handling Cross Reference Table Structures	29-3
Knowledge Modules	29-3
Overview of the SOA XREF KM Process	29-4
Loading Phase (LKM)	29-4
Integration and Cross-Referencing Phase (IKM)	29-5
Updating/Deleting Processed Records (LKM)	29-5
Installation and Configuration	29-6
System Requirements and Certifications	29-6
Technology Specific Requirements	29-6
Connectivity Requirements	29-6
Working with XREF using the SOA Cross References KMs	29-6

Defining the Topology	29-7
Setting up the Project	29-7
Designing a Mapping with the Cross-References KMs	29-7
Knowledge Module Options Reference	29-8

30 Oracle Object Storage

Introduction	30-1
Concepts	30-1
Installation and Configuration	30-2
System Requirements & Certifications	30-2
Technology Specific Requirements	30-3
Setting up the Topology	30-3
Creating an Oracle Object Storage Data Server	30-3
Creating an Oracle Object Storage Physical Schema	30-5
Creating and Reverse-Engineering an Oracle Object Storage Model	30-6
Creating an Oracle Object Storage Model	30-6
Reverse Engineering an Oracle Object Storage Model	30-6
Reverse-Engineering Delimited Files from Oracle Object Storage	30-6
Reverse-engineering Fixed Files from Oracle Object Storage	30-7
Reverse-Engineering JSON, Avro and Parquet Storage Formats	30-8
Working with Oracle Object Storage Tools	30-8
Uploading Files/Objects to Oracle Object Storage	30-9
Downloading Files/Objects from Oracle Object Storage	30-11
Deleting Files/Objects from Oracle Object Storage	30-12
Designing a Mapping	30-14
Setting up an Integration Project	30-14
LKM File to Oracle Object Storage	30-14
LKM File to Oracle Object Storage Direct	30-15
LKM SQL to Oracle Object Storage	30-15
LKM SQL to Oracle Object Storage Direct	30-16

31 Oracle Storage Cloud Service

Introduction	31-1
Concepts	31-1
Installation and Configuration	31-2
System Requirements and Certifications	31-2
Technology Specific Requirements	31-2
Setting up the Topology	31-3
Creating an Oracle Storage Cloud Service Data Server	31-3

Creating an Oracle Storage Cloud Service Physical Schema	31-4
Creating and Reverse-Engineering an Oracle Storage Cloud Service Model	31-4
Creating an Oracle Storage Cloud Service Model	31-4
Reverse Engineering an Oracle Storage Cloud Service Model	31-5
Reverse-Engineering Delimited Files from Oracle Storage Cloud Service	31-5
Reverse-engineering Fixed Files from Oracle Storage Cloud Service	31-5
Reverse-Engineering JSON, Avro and Parquet Storage Formats	31-6
Working with Oracle Storage Cloud Service Tools	31-7
Uploading Files/Objects to Oracle Storage Cloud Service	31-7
Downloading File/Objects from Oracle Storage Cloud Service	31-9

Part IV SaaS Applications

32 Oracle Enterprise Resource Planning Cloud

Introduction	32-1
Concepts	32-1
Knowledge Modules	32-1
Prerequisites	32-2
Installation and Configuration	32-2
System Requirements and Certifications	32-2
Technology Specific Requirements	32-2
Connectivity Requirements	32-3
Setting up the Topology	32-3
Creating an Oracle ERP Cloud Data Server	32-3
Creating an Oracle ERP Cloud Physical Schema	32-3
Creating and Reverse-Engineering an Oracle ERP Cloud Datastore	32-4
Creating an Oracle ERP Cloud Model	32-4
Creating an Oracle ERP Cloud Datastore	32-4
Defining Parameters for BI Publisher Report	32-5
Reverse-Engineering an Oracle ERP Cloud Datastore	32-5
Designing a Mapping	32-5
Loading Data from Oracle ERP Cloud	32-5
Troubleshooting	32-7

33 Oracle Marketing Cloud

Introduction	33-1
Concepts	33-1
Knowledge Modules	33-1
Installation and Configuration	33-1

System Requirements and Certifications	33-2
Technology Specific Requirements	33-2
Connectivity Requirements	33-2
Setting up the Topology	33-2
Creating an Oracle Marketing Cloud Data Server	33-2
Creating an Oracle Marketing Cloud Physical Schema	33-3
Creating and Reverse-Engineering an Oracle Marketing Cloud Model	33-3
Creating an Oracle Marketing Cloud Model	33-3
Reverse-engineer an Oracle Marketing Cloud Model	33-3
Designing a Mapping	33-3

34 Oracle Sales Cloud

Introduction	34-1
Concepts	34-1
Knowledge Modules	34-1
Installation and Configuration	34-1
System Requirements and Certifications	34-2
Technology Specific Requirements	34-2
Connectivity Requirements	34-2
Setting up the Topology	34-2
Creating an Oracle Sales Cloud Data Server	34-2
Creating an Oracle Sales Cloud Physical Schema	34-3
Creating and Reverse-Engineering an Oracle Sales Cloud Model	34-3
Creating an Oracle Sales Cloud Model	34-3
Reverse-engineer an Oracle Sales Cloud Model	34-3
Designing a Mapping	34-3

35 Oracle Service Cloud

Introduction	35-1
Concepts	35-1
Knowledge Modules	35-1
Installation and Configuration	35-1
System Requirements and Certifications	35-2
Technology Specific Requirements	35-2
Connectivity Requirements	35-2
Setting up the Topology	35-2
Creating an Oracle Service Cloud Data Server	35-2
Creating an Oracle Service Cloud Physical Schema	35-3
Creating and Reverse-Engineering an Oracle Service Cloud Model	35-3

Creating an Oracle Service Cloud Model	35-3
Reverse-engineer an Oracle Service Cloud Model	35-3
Designing a Mapping	35-3

Part V Appendices

A Oracle Data Integrator Driver for LDAP Reference

Introduction to Oracle Data Integrator Driver for LDAP	A-1
LDAP Processing Overview	A-1
LDAP to Relational Mapping	A-2
General Principle	A-2
Grouping Factor	A-4
Mapping Exceptions	A-5
Reference LDAP Tree	A-5
Managing Relational Schemas	A-6
Relational Schema Storage	A-6
Accessing Data in the Relational Structure	A-7
Installation and Configuration	A-7
Driver Configuration	A-8
Using an External Database to Store the Data	A-13
Passing the Properties in the Driver URL	A-13
Setting the Properties in ODI Studio	A-13
Setting the Properties in a Properties File	A-14
LDAP Directory Connection Configuration	A-16
Table Aliases Configuration	A-16
SQL Syntax	A-18
SQL Statements	A-18
DISCONNECT	A-18
INSERT INTO	A-19
SELECT	A-19
UPDATE	A-19
Expressions, Condition & values	A-19
SQL FUNCTIONS	A-20
JDBC API Implemented Features	A-23

B Oracle Data Integrator Driver for XML Reference

Introduction to Oracle Data Integrator Driver for XML	B-1
XML Processing Overview	B-1
XML to SQL Mapping	B-2

XML Namespaces	B-3
Managing Schemas	B-3
Schema Storage	B-3
Multiple Schemas	B-4
Accessing Data in the Schemas	B-4
Case Sensitivity	B-4
Loading/Synchronizing	B-5
Locking	B-5
XML Schema (XSD) Support	B-5
Installation and Configuration	B-5
Driver Configuration	B-6
Automatically Create Multiple Schemas	B-11
Using an External Database to Store the Data	B-11
Detailed Driver Commands	B-17
CREATE FILE	B-18
CREATE FOREIGNKEYS	B-19
CREATE XMLFILE	B-19
CREATE SCHEMA	B-20
DROP FOREIGNKEYS	B-21
DROP SCHEMA	B-22
LOAD FILE	B-22
SET SCHEMA	B-23
SYNCHRONIZE	B-24
UNLOCK FILE	B-24
TRUNCATE SCHEMA	B-24
VALIDATE	B-25
WRITE MAPPING FILE	B-25
SQL Syntax	B-26
SQL Statements	B-26
COMMIT	B-27
CREATE TABLE	B-27
DELETE	B-27
DISCONNECT	B-28
DROP TABLE	B-28
INSERT INTO	B-28
ROLLBACK	B-28
SELECT	B-28
SET AUTOCOMMIT	B-29
UPDATE	B-29
Expressions, Condition and Values	B-29
SQL FUNCTIONS	B-30

JDBC API Implemented Features	B-32
Rich Metadata	B-33
Supported user-specified types for different databases	B-35
XML Schema Supported Features	B-35
Datatypes	B-36
Supported Elements	B-36
All	B-37
Any	B-37
AnyAttribute	B-37
AnyType	B-38
Attribute	B-38
AttributeGroup	B-38
Choice	B-38
ComplexContent	B-39
ComplexType	B-39
Element	B-39
Extension	B-40
Group	B-40
Import	B-41
Include	B-41
List	B-41
Restriction	B-41
Schema	B-42
Sequence	B-42
SimpleContent	B-42
SimpleType	B-43
Unsupported Features	B-43
Unsupported Elements	B-43
Unsupported Features	B-43
Unsupported Datatypes	B-43

C Oracle Data Integrator Driver for Complex Files Reference

Introduction to Oracle Data Integrator Driver for Complex Files	C-1
Complex Files Processing Overview	C-2
Generating the Native Schema	C-2
XML to SQL Mapping	C-2
JSON Support	C-3
Supported Features	C-3
Driver Configuration	C-3
Detailed Driver Commands	C-5

D Pre/Post Processing Support for XML and Complex File Drivers

Overview	D-1
Configuring the processing stages	D-1
Implementing the processing stages	D-3
Example: Groovy Script for Reading XML Data From Within a ZIP File	D-4
Example: Groovy Script for Transforming XML Data and Writing to a Different Format	D-5
Example: Java Class for Reading Data From HTTP Source Requiring Authentication	D-6
Example: Groovy Code Embedded in Configuration XML File	D-8

Preface

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

This preface contains the following topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Documents](#)
- [Conventions](#)

Audience

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in *Oracle Data Integrator Library*.

- *Release Notes for Oracle Data Integrator*
- *Understanding Oracle Data Integrator*
- *Administering Oracle Data Integrator*
- *Developing Integration Projects with Oracle Data Integrator*
- *Installing and Configuring Oracle Data Integrator*
- *Upgrading Oracle Data Integrator*
- *Application Adapters Guide for Oracle Data Integrator*
- *Developing Knowledge Modules with Oracle Data Integrator*

- *Migrating From Oracle Warehouse Builder to Oracle Data Integrator*
- *Oracle Data Integrator Tools Reference*
- *Data Services Java API Reference for Oracle Data Integrator*
- *Open Tools Java API Reference for Oracle Data Integrator*
- *Getting Started with SAP ABAP BW Adapter for Oracle Data Integrator*
- *Java API Reference for Oracle Data Integrator*
- *Getting Started with SAP ABAP ERP Adapter for Oracle Data Integrator*
- *Oracle Data Integrator 12c Online Help*, which is available in ODI Studio through the JDeveloper Help Center when you press **F1** or from the main menu by selecting **Help**, and then **Search** or **Table of Contents**.

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.
<i>italic</i>	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

The *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* describes how to work with different technologies in Oracle Data Integrator. This book includes the following parts:

- [Databases, Files, and XML](#)
- [Business Intelligence](#)
- [Other Technologies](#)
- [SaaS Applications](#)

Application Adapters are covered in a separate guide. See the *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:

- [Terminology](#)
- [Using This Guide](#)

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 supported 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 *Application Adapters Guide for Oracle Data Integrator* for more information.

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 [Generic SQL](#) and in [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.

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.

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.

Part I

Databases, Files, and XML

It is important to understand how to work with databases, files, and XML files in Oracle Data Integrator.

Part I contains the following chapters:

- [Oracle Database](#)
- [Oracle Autonomous Data Warehouse Cloud](#)
- [Oracle Autonomous Transaction Processing](#)
- [Files](#)
- [Generic SQL](#)
- [XML Files](#)
- [Complex Files](#)
- [Microsoft SQL Server](#)
- [Microsoft Excel](#)
- [Microsoft Access](#)
- [Netezza](#)
- [Teradata](#)
- [Hypersonic SQL](#)
- [IBM Informix](#)
- [IBM DB2 for iSeries](#)
- [IBM DB2 UDB](#)
- [Salesforce.com](#)

2

Oracle Database

It is important to understand how to work with Oracle Database in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering an Oracle Model](#)
- [Setting up Changed Data Capture](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)
- [Troubleshooting](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in the following table 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 [Generic SQL](#) for more information.

Table 2-1 Oracle KMs

Knowledge Module	Description
RKM Oracle	Reverse-engineers tables, views, columns, primary keys, non unique indexes and foreign keys.
JKM Oracle 11g Consistent (Streams)	Creates the journalizing infrastructure for consistent set journalizing on Oracle 11g tables, using Oracle Streams. This KM is deprecated.
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 <i>Last Update Date</i> 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 Online)	Creates and manages the ODI CDC framework infrastructure when using Oracle GoldenGate for CDC. See 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 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 Pull (DB Link)	Loads data from an Oracle source database to an Oracle staging area database using database links. It does not create a view in the source database. It also does not creates the synonym in the staging database. Built-in KM.
LKM Oracle to Oracle Push (DB Link)	Loads and integrates data into Oracle target table using database links. It does not create the synonym in the staging database. Any settings in the IKM would be ignored. Built-in KM.
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.

Table 2-1 (Cont.) Oracle KMs

Knowledge Module	Description
LKM SAP BW to Oracle (SQLLDR)	Loads data from SAP BW systems to an Oracle staging using SQL*Loader utilities. See the <i>Application Adapters Guide for Oracle Data Integrator</i> 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 <i>Application Adapters Guide for Oracle Data Integrator</i> for more information.
IKM Oracle Incremental Update	Integrates data in an Oracle target table in incremental update mode. Supports Flow Control.
IKM Oracle Incremental Update (MERGE)	Integrates data in an Oracle target table in incremental update mode, using a MERGE statement. Supports Flow Control.
IKM Oracle Incremental Update (PL SQL)	Integrates data in an Oracle target table in incremental update mode using PL/SQL. Supports Flow Control.
IKM Oracle Insert	Integrates data into an Oracle target table in append mode. The data is loaded directly in the target table with a single INSERT SQL statement. Built-in KM.
IKM Oracle Update	Integrates data into an Oracle target table in incremental update mode. The data is loaded directly into the target table with a single UPDATE SQL statement. Built-in KM.
IKM Oracle Merge	Integrates data into an Oracle target table in incremental update mode. The data is loaded directly into the target table with a single MERGE SQL statement. Built-in KM.
IKM Oracle Multi-Insert	Integrates data from one source into one or many Oracle target tables in append mode, using a multi-table insert statement (MTI). This IKM can be utilized in a single mapping to load multiple targets. Built-in KM.
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). Supports Flow Control.
IKM Oracle Slowly Changing Dimension	Integrates data in an Oracle target table used as a Type II Slowly Changing Dimension. Supports Flow Control.
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. Supports Flow Control.
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. Supports Flow Control. This IKM is typically used for ETL configurations: source and target tables are on different Oracle instances and the mapping'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. For information about how to use this SKM, see <i>Generating and Deploying Data Services</i> in the <i>Administering Oracle Data Integrator</i> .

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

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.

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.

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.

Using Oracle Wallet

Oracle Wallet provides a simple and easy method to manage database credentials across multiple domains.

NOT_SUPPORTED:

This section applies only to Data Integration Platform Cloud.

This section describes the requirements that are necessary for creating and managing wallet in an Oracle database environment.

Note:

For more details on creating and managing wallets, refer to [Managing Oracle Wallets](#) section of *Enterprise User Security Administrator's Guide*.

This environment provides all the necessary commands and libraries, including `$ORACLE_HOME/oracle_common/bin/mkstore` command.

- Oracle recommends you to create and manage Wallet in a database environment. Often this task is completed by a database administrator and provided for use to the client. You can also install the Oracle Client Runtime package to provide the necessary commands and libraries to create and manage Oracle Wallet.

Create a wallet on the client using the command: `mkstore -wrl <wallet_location> -create`, where: `wallet_location` is the path to the directory where you want to create and store the wallet.

For Example — `mkstore -wrl /scratch/ewallet -createCredential jdbc:oracle:thin:@kkm00ebs.in.oracle.com:1523:oditest odiUser odi where`

- `kkm00ebs.in.oracle.com` is the host name
- `1523` is the port number
- `oditest` is the SID
- `odiUser` denotes the user name
- `odi` denotes the password for the user

Note:

You can store multiple credentials for multiple databases in a single client wallet. You cannot store multiple credentials (for logging into multiple schema) for the same database in the same wallet. If you have multiple login credentials for the same database, then they must be stored in separate wallets.

- To add database login credentials to an existing client wallet, use the command:
`mkstore -wrl <wallet_location> -createCredential <db_connect_string>
<username> <password> where`
 - `wallet_location` is the path to the directory where you have created the wallet
 - `db_connect_string` must be identical to the connection string that you specify in the URL used in the datasource definition (the part of the string that follows the @).

It can be either the short form or the long form of the URL.

* For Example — `jdbc:oracle:thin:@kkm00ebs.in.oracle.com:1523/
oditest`

or

* `(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)
(HOST=myhost-scan)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME=myservice)))`

For Example — `(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)
(HOST=<hostname or ipaddress>)(PORT=<port>))
(CONNECT_DATA=(SERVICE_NAME=<db service>))
(security=(ssl_server_cert_dn="<certificate_info>")))`



Note:

Oracle supports two types of wallets: 1. Password protected wallets (`ewallet.p12`) and 2. Auto login wallets (`cwallet.sso`) is used for retrieving connection details to establish secure connection to an instance by verifying certificate details associated with instance for secure connection.

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) or Service Name

- Login and password of an Oracle User.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating an Oracle Data Server](#)
2. [Creating an Oracle Physical Schema](#)

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.

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 Administering Oracle Data Integrator*. This section details only the fields required or specific for defining an Oracle data server:

1. In the Definition tab:

Data Server

- **Name:** Name of the data server that will appear in Oracle Data Integrator.
- **Instance/dblink(Data Server):** TNS Alias used for this Oracle instance. It will be used to identify the Oracle instance when using database links and SQL*Loader.

Connection

- **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.
- **JNDI Connection:** Select this check-box to configure the JNDI connection settings. Navigate to the **JNDI** tab, and fill in the required fields.
- **Use Credential File:** Select this check box to upload the connection details directly from a pre-configured wallet file¹ or credential zip file.

Credential Details

NOT_SUPPORTED:

This section applies only to Data Integration Platform Cloud.

¹ In ODI, you can directly upload connection details from a credential file (auto login wallet) or a password protected wallet file (ewallet.p12)

- **Credential File:** Click the browse icon present beside the **Credential File** text box, to browse for the location of the required wallet file containing the connection details.

By default, the credential location is populated to point to the same wallet file (ewallet.p12) or credential zip file (auto login wallet) that is used for login which is present in `<homedir>/<odi>/oracledi/ewallet`. You can also browse for any other location where this credential file is stored with the required database connection details.

Based on your selection, the following options appear:

- a. If you have selected a password protected wallet file (ewallet.p12),
 - The **Credential File Password** text box appears. It represents the password that you configured during wallet creation and it is required to open the wallet file. Enter the relevant password of the wallet file in this text box.

 **Note:**

ODI supervisor can distribute ewallet.p12 and wallet password to other ODI users.

- **Connection Details** – If the entered password is valid, ODI retrieves a list of all available database connection details that are present in the wallet and displays it. Click the **Connection Details** drop down arrow to choose the required database connection from the list.
- b. If you have selected a credential zip file (auto login wallet),
 - The **Connection Details** text box appears. Click the **Connection Details** drop down arrow to choose the required connection URL from the list of available connection URLs retrieved from `tnsnames.ora`.
- Upon selecting the required connection URL, JDBC Driver, JDBC URL, Username and Password fields are disabled and the associated username, password, jdbc URL and jdbc driver details are auto-populated with credentials retrieved from the pre-configured wallet file or credential zip file. You can change them only through the list of connections available in the wallet file (ewallet.p12) or credential zip (`tnsnames.ora`).
 - Click **Save**, to save the Data Server details
 - Click **Test Connection**, to test the established connection
2. If the data server supports JNDI access, fill-in the details of the JNDI tab:
 - **JNDI authentication:** Select the required option from the following-
 - **None-** Anonymous access to the naming or directory service
 - **Simple:** Authenticated access, non-encrypted
 - **CRAM-MD5:** Authenticated access, encrypted MD5
 - **<other value>:** authenticated access, encrypted according to <other value>
 - **JNDI User/Password:** User/password connecting to the JNDI directory
 - **JNDI Protocol:** Protocol used for the connection

 **Note:**

Please note that only the most common protocols are listed here. This is not an exhaustive list.

- **LDAP:** Access to an LDAP directory
 - **SMQP:** Access to a SwiftMQ MOM directory
 - **<other value>:** access following the sub-protocol <other value>
 - **JNDI Driver:** The driver allowing the JNDI connection
For Example — Sun LDAP directory: `com.sun.jndi.ldap.LdapCtxFactory`
 - **JNDI URL:** The URL allowing the JNDI connection
For example: `ldap://suse70:389/o=linuxfocus.org`
 - **JNDI Resource:** The directory element containing the connection parameters
For example: `cn=sampled`
3. If the data server supports JDBC access, fill-in the details of the JDBC tab:
- **JDBC Driver:** Name of the JDBC driver used for connecting to the data server
`oracle.jdbc.OracleDriver`
 - **JDBC URL:** URL allowing you to connect to the data server.
`jdbc:oracle:thin:@<network name or ip address of the Oracle machine>:<port of the Oracle listener>:<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)))`

Creating an Oracle Physical Schema

Create an Oracle physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using the Oracle Database follows the standard procedure. See *Creating an Integration Project of the Developing Integration Projects with 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

Creating and Reverse-Engineering an Oracle Model

This section contains the following topics:

- [Create an Oracle Model](#)
- [Reverse-engineer an Oracle Model](#)

Create an Oracle Model

Create an Oracle Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

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 Developing Integration Projects with 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 Developing Integration Projects with Oracle Data Integrator*. This section details only the fields specific to the Oracle technology:

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

Setting up Changed Data Capture

The ODI Oracle Knowledge Modules support the Changed Data Capture feature. See Using Changed Data of *Developing Integration Projects with 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 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 <i>Last Update Date</i> 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 [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

CDC Group	Naming Convention
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.

Setting up Data Quality

Oracle Data Integrator provides the CKM Oracle for checking data integrity against constraints defined on an Oracle table. See Flow Control and Static Control in *Developing Integration Projects with Oracle Data Integrator*.

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 [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

Designing a Mapping

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

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

Loading Data from and to Oracle

Oracle can be used as a source, target or staging area of a mapping. The LKM choice in the Mapping's Loading Knowledge Module tab to load data between Oracle and another type of data server is essential for the performance of a mapping.

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 Push (DB Link)	Creates a view on the source server, but does not create synonyms on this view on the target server. This KM ignores any settings on the IKM. Built-in KM.
Oracle	LKM Oracle to Oracle Pull (DB Link)	Does not create a view on the source server, or the synonyms on this view on the target server. Built-in KM.
Oracle	LKM Oracle to Oracle (datapump)	Uses external tables in the datapump format.

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.
Oracle	LKM Oracle to Oracle Push (DB Link)	Views not created on the source server, synonyms created on the target. Built-in KM.
Oracle	LKM Oracle to Oracle Pull (DB Link)	Views not created on the source server, synonyms not created on the target. Built-in KM.
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.

Source or Staging Area Technology	KM	Notes
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)

Integrating Data in Oracle

The data integration strategies in Oracle are numerous and cover several modes. The IKM choice in the Mapping's Physical diagram 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. Supports Flow Control.
Update	IKM Oracle Update	Optimized for Oracle. Oracle UPDATE statement KM. Built-in KM.
Update	IKM Oracle Merge	Optimized for Oracle. Oracle MERGE statement KM. Built-in KM.
Update	IKM Oracle Spatial Incremental Update	Supports SDO_GEOMETRY datatypes. Supports Flow Control.
Update	IKM Oracle Incremental Update (MERGE)	Recommended for very large volumes of data because of bulk set-based MERGE feature. Supports Flow Control.
Update	IKM Oracle Incremental Update (PL SQL)	Use PL/SQL and supports long and blobs in incremental update mode. Supports Flow Control.
Specific	IKM Oracle Slowly Changing Dimension	Supports type 2 Slowly Changing Dimensions. Supports Flow Control.
Specific	IKM Oracle Multi Table Insert	Supports multi-table insert statements. Supports Flow Control.
Append	IKM Oracle to Oracle Control Append (DBLINK)	Optimized for Oracle using DB*Links. Supports Flow Control.
Append	IKM Oracle Insert	Optimized for Oracle. Oracle INSERT statement KM. Built-in KM. Supports Flow Control.

Mode	KM	Note
Append	IKM Oracle Multi-Insert	Optimized for Oracle. Oracle multi-target INSERT statement KM, applied to each target. Built-in KM.

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 mappings that are sequenced in a Package. This Package must meet the following conditions:

- The first mapping of the Package must have a temporary target and the KM option *DEFINE_QUERY* set to *YES*.
This first mapping defines the structure of the SELECT clause of the multi-table insert statement (that is the source flow).
- Subsequent mappings must source from this temporary datastore and have the KM option *IS_TARGET_TABLE* set to *YES*.
- The last mapping 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 Mapping as Derived Table (Sub-Select)* to *true* on any of the mappings.

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_GEOMETRY* 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.

Designing an ETL-Style Mapping

See *Creating a Mapping in Developing Integration Projects with Oracle Data Integrator* for generic information on how to design mappings. This section describes how to design an ETL-style mapping where the staging area is Oracle database or any ANSI-92 compliant database and the target on Oracle database.

In an ETL-style mapping, 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 [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 mapping:

1. Create a mapping 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 a Mapping of Developing Integration Projects with Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. Change the staging area for the mapping to the logical schema of the source tables or a third schema. See *Configuring Execution Locations of Developing Integration Projects with Oracle Data Integrator* for information about how to change the staging area.
3. In the Physical diagram, select an access point. The Property Inspector opens for this object.
4. In the Loading Knowledge Module tab, select an LKM 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 Physical diagram, select the Target by clicking its title. The Property Inspector opens for this object.

In the Integration Knowledge Module tab, select an ETL multi-connection IKM 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 Check Knowledge Module 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 mapping. 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 mapping:

1. Create a mapping with the staging area and target on Oracle using the standard procedure as described in *Creating a Mapping of Developing Integration Projects with Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. Change the staging area for the mapping to the logical schema of the source tables or a third schema. See *Configuring Execution Locations of Developing Integration Projects with Oracle Data Integrator* for information about how to change the staging area.
3. In the Physical diagram, select an access point. The Property Inspector opens for this object.
4. In the Loading Knowledge Module tab, select an LKM 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 access point for the Staging Area. The Property Inspector for this object appears.
7. In the Loading Knowledge Module tab, select an LKM to load from the staging area to the target. See [Table 2-5](#) to determine the LKM you can use.
8. Optionally, modify the KM options.
9. Select the Target by clicking its title. The Property Inspector opens for this object.

In the Integration Knowledge Module tab, select a standard mono-connection IKM to update the target. See [Table 2-5](#) to determine the IKM you can use.

Table 2-5 KM Guidelines for ETL-Style Mappings 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: <ul style="list-style-type: none"> • Perform control append • Use DB*Links for performance reasons Supports flow and static control.

Table 2-5 (Cont.) KM Guidelines for ETL-Style Mappings with Oracle Data

Source	Staging Area	Target	Exporting LKM	IKM	KM Strategy	Comment
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	<p>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.</p> <p>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.</p> <p>Supports flow and static control</p>
Oracle	Oracle	Oracle	LKM to Oracle to Oracle (DBLINK)	IKM Oracle Slowly Changing Dimension	LKM + standard IKM	na
Oracle	Oracle	Oracle	LKM to Oracle to Oracle (DBLINK)	IKM Oracle Incremental Update	LKM + standard IKM	na
Oracle	Oracle	Oracle	LKM to Oracle to Oracle (DBLINK)	IKM Oracle Incremental Update (MERGE)	LKM + standard IKM	na

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](#)

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.SQLException` 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 it 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 sent to the database to find the source of the error. The code is displayed in the description tab of the erroneous task.

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 mapping, or typed in a procedure is invalid for Oracle. This is usually related to an input error in the mapping, filter or 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 mapping 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

Oracle Autonomous Data Warehouse Cloud

This chapter describes how to work with Autonomous Data Warehouse Cloud (ADWC) in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Prerequisites](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Oracle Model](#)
- [Designing a Mapping](#)
- [Best Practices for Working with ADWC](#)

Introduction

Autonomous Data Warehouse Cloud (ADWC) is a fully-managed, high-performance elastic cloud service providing analytical capability over data stored in the database and Oracle Object Storage.

Oracle Data Integrator (ODI) seamlessly integrates with ADWC. By integrating ODI with ADWC, you can get the full performance of Oracle Database, in a fully-managed environment that is tuned and optimized for data warehouse workloads.

Concepts

The Oracle ADWC concepts map the Oracle Data Integrator concepts as follows: An Oracle ADWC 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 ADWC instance. All connections to the ADWC require the use of an Oracle Wallet to manage public key security credentials. ODI users can use a Thin or OCI connection to access ADWC.

Knowledge Modules

Oracle Data Integrator provides the following Knowledge Modules (KM) for loading data into ADWC. The KMs use Oracle specific features. It is also possible to use the generic SQL KMs with ADWC.

Table 3-1 ADWC Knowledge Modules

Knowledge Module	Description
LKM SQL to Oracle (Built-In)	Loads data from any ANSI SQL-92 source database to an Oracle staging area.
LKM SQL Multi-Connect	Enables the use of multi-connect IKM for target table. Built-in IKM.
LKM File to Oracle (SQL*Loader)	Loads data from a file to an Oracle staging area using the SQL*Loader command line utility.
IKM SQL to File Append	Integrates data in a target file from any ANSI SQL-92 compliant staging area in replace mode.
IKM Oracle Insert	Integrates data into an Oracle target table in append mode. The data is loaded directly in the target table with a single INSERT SQL statement. Built-in KM.
IKM Oracle Update	Integrates data into an Oracle target table in incremental update mode. The data is loaded directly into the target table with a single UPDATE SQL statement. Built-in KM.
IKM Oracle Merge	Integrates data into an Oracle target table in incremental update mode. The data is loaded directly into the target table with a single MERGE SQL statement. Built-in KM.
IKM Oracle Multi-Insert	Integrates data from one source into one or many Oracle target tables in append mode, using a multi-table insert statement (MTI). This IKM can be utilized in a single mapping to load multiple targets. Built-in KM.
RKM Oracle	Reverse-engineers tables, views, columns and creates data models to use as targets or sources in Oracle Data Integrator mappings.
LKM SQL to ADWC External Table	Loads data from SQL source to Oracle ADWC using Oracle Object Storage as intermediate staging. You have to use this LKM in combination with Oracle or generic SQL IKM.
	<p>NOT_SUPPORTED:</p> <p>This KM applies only to Data Integration Platform Cloud.</p>
LKM SQL to ADWC Copy	Loads data from SQL source to Oracle ADWC using Oracle Object Storage as intermediate staging. You have to use this LKM in combination with Oracle or generic SQL IKM.

 **Note:**

This KM applies only to Data Integration Platform Cloud.

Table 3-1 (Cont.) ADWC Knowledge Modules









Knowledge Module	Description
LKM SQL to ADWC Copy Direct	Loads data from SQL source to Oracle ADWC using Oracle Object Storage as intermediate staging. You can use this LKM as a standalone KM as you do not need any IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM File to ADWC External Table	Loads data from local or HDFS file source to Oracle ADWC using Oracle Object Storage as intermediate staging. You have to use this LKM in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM File to ADWC Copy	Loads data from local or HDFS file source to Oracle ADWC using Oracle Object Storage as intermediate staging. You have to use this LKM in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM File to ADWC Copy Direct	Loads data from local or HDFS file source to Oracle ADWC using Oracle Object Storage as intermediate staging. You can use this LKM as a standalone KM as you do not need an IKM for its implementation.
	 Note: This KM applies only to Data Integration Platform Cloud.

Table 3-1 (Cont.) ADWC Knowledge Modules

Knowledge Module	Description
LKM Oracle to ADWC Datapump	Loads data from Oracle On-premises products to Oracle ADWC.. You can use it in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM Object Storage to ADWC Copy	Loads data from Oracle Cloud Object Storage to Oracle ADWC. You can use it in combination with Oracle or generic SQL IKM
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM Object Storage to ADWC Copy Direct	Loads data from Oracle Cloud Object Storage to Oracle ADWC. You can use this LKM to move files/objects from Oracle Object Storage to an ADWC table.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM Object Storage to ADWC External Table	Loads data from Oracle Object Storage to Oracle ADWC using External Table method. You can use this LKM in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.

Prerequisites

The following prerequisites are essential for connecting to ADWC environment. Make sure you go through the following prerequisites, before connecting to ADWC environment.

 **Note:**

The following prerequisites are common for both ODI Studio and ODI Agent.

Wallet Configuration

All connections to ADWC require the use of an Oracle Wallet to manage public key security credentials. A wallet is a password-protected container used to store authentication and signing credential, including private key, certificates, and trusted certificates needed by SSL. Oracle Wallet provides a simple and easy method to manage database credentials across multiple domains. It allows you to update database credentials by updating the Wallet instead of having to change individual data source definitions. To connect to the ADWC, applications need access to the Oracle wallet.

For more details on wallet, refer to **Securing Passwords in Application Design** section of [Managing Security for Application Developers](#) in Database Security Guide.

The JDBC properties require a Wallet file location.

- Get the wallet zip file from ADWC wallet location and place it in a local directory accessible to both ODI Studio and ODI Agent. The default location of the wallet file is `<homedir>/.odi/oracledi/ewallet`. For auto login wallet, upload the zip as wallet file.

Java Security configuration

 **Note:**

Steps listed below are not required for JDK1.8.0_u161 or later versions.

Update the file `java.security`, from the location `JDK_HOME\jre\lib\security` and add the following lines of code, as shown below:

- `security.provider.10=sun.security.mscaapi.SunMSCAPI`
- `security.provider.11=oracle.security.pki.OraclePKIProvider`

```
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=sun.security.ec.SunEC
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
```



```
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=sun.security.mscaapi.SunMSCAPI
security.provider.11=oracle.security.pki.OraclePKIProvider
```

Preparing for OCI Connection

Follow the below steps, to create a OCI Connection :

1. Download and install Oracle Instant Client: Version 12.2.0.1.0. For more instructions, see <http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html>
2. After installing the Oracle Instant Client, add its folder in the environment variable called LD_LIBRARY_PATH: `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/oracle/instantclient_12_2`
3. Unzip the credential file into a new directory: i.e. `/home/oracle/wallet`
4. Navigate to the new wallet directory, and modify the `sqlnet.ora` file as:
 - specify the wallet location.

For Example —

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
(DIRECTORY=/home/oracle/wallet)))
```

- add the following two SSL parameters in the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH= TRUE
SSL_VERSION = 1.2
```

5. Change or set `TNS_ADMIN` to the location or directory where the unzipped credential files are located. For instance, if your wallet file was unzipped to a directory called `/home/oracle/wallet` then, set `TNS_ADMIN` parameter as follows:

```
TNS_ADMIN=/home/oracle/wallet
```

Agent Configurations for OCI

Perform the following configurations to use OCI, to connect ODI Agents (Standalone and J2EE) with ADWC:

- In Local (No Agent), ODI Standalone agent and J2EE Agent, set the instant client folder for the `LD_LIBRARY_PATH` parameter before launching ODI Studio. For example, `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/oracle/instantclient_12_2`

Note:

The existing value of jars or directories mentioned in the `LD_LIBRARY_PATH` parameter should be consistent to the current version of instant client jars . If not, you may get an error.

- For ODI Standalone Agent, configure the instant client library path in `ODI_ADDITIONAL_JAVA_OPTIONS` variable, present in `instance.sh/cmd` of the

standalone domain. The `instance.sh` file is located at : `<DOMAIN_HOME>config/fmwconfig/components/ODI/<agent-name>/bin` . For Example,

```
ODI_ADDITIONAL_JAVA_OPTIONS="{ODI_ADDITIONAL_JAVA_OPTIONS} -  
Djava.library.path={ODI_HOME}/../sdk/lib:/home/oracle/  
instantclient_12_2"
```

- For ODI J2EE Agent, set `JAVA_OPTIONS` parameter to `$JAVA_OPTIONS -Djava.library.path=/home/oracle/instantclient_12_2`. For example, export `JAVA_OPTIONS="$JAVA_OPTIONS -Djava.library.path=/home/oracle/instantclient_12_2"`

Setting up the Topology

Note:

Please note the ADWC data server is created under **Oracle Technology**.

Setting up the Topology consists of:

- [Creating an Oracle Data Server](#)
- [Creating an Oracle Physical Schema](#)

Creating an Oracle Data Server

Create a data server for ADWC using the standard procedure, as described in [Creating a Data Server of Administering Oracle Data Integrator](#). This section details only the properties required to be set in the data server created under Oracle technology for ADWC:

1. In the **Definition** tab:

Data Server

- **Name** : Enter a name for the data server definition.
- **Instance / dblink (Data Server)** : TNS Alias used for this Oracle instance. It will be used to identify the Oracle instance when using database links and SQL*Loader.

Connection

- **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.
- **JNDI Connection** : Select this check-box to configure the JNDI connection settings. Navigate to the JNDI tab, and fill in the required fields.

 **Note:**

JNDI connection field is not applicable for ADWC.

- **Use Credential File**

 **Note:**

Please note it is required to use a credential file with ADWC.

Select this check box to upload the connection details directly from a pre-configured wallet file¹ or credential zip file. The above JNDI Connection check-box is disabled and the following fields with respect to Wallet feature appear:

Credential Details

 **Note:**

This section applies only to Data Integration Platform Cloud.

- **Credential File:** Click the browse icon present beside the **Credential File** text box, to browse for the location of the required wallet file containing the connection details.

By default, the credential location is populated to point to the same wallet file (ewallet.p12) or credential zip file (auto login wallet) that is used for login which is present in `<homedir>/.odi/oracledi/ewallet`. You can also browse for any other location where this credential file is stored with the required database connection details.

 **Note:**

ODI supervisor can distribute the password protected wallet and credential zip file to other ODI users.

- Click the **Connection Details** drop down arrow to choose the required connection URL from the list of available connection URLs retrieved from `tnsnames.ora`.
- Upon selecting the required connection URL, JDBC Driver, JDBC URL, Username and Password fields are disabled and the associated username, password, jdbc URL and jdbc driver details are auto-populated with credentials retrieved from the pre-configured wallet file or credential zip file. You can change them only through the list of connections available in the wallet file (ewallet.p12) or credentials zip (tnsnames.ora).
- Click **Save**, to save the Data Server details

¹ In ODI, you can directly upload connection details from a credential file (cwallet.sso) or a password protected wallet file (ewallet.p12).

- Click **Test Connection**, to test the established connection
2. The ADWC Data Server (using the Oracle Technology) supports JDBC. Fill-in the following details of the **JDBC** tab:

ODI users can use a Thin or OCI connection to access ADWC.

 **Note:**

The JDBC tab will be auto populated with the JDBC information contained in the credential file. You can review the auto-populated information in the JDBC tab and make edits (only if required).

- **JDBC Driver** : Name of the JDBC driver used for connecting to the data server. `oracle.jdbc.OracleDriver`
- **JDBC URL** : URL allowing you to connect to the data server.

Format of the 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>`

- a. To connect an ADWC instance with the Oracle JDBC thin driver, use a database URL as shown below:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)
(HOST=<hostname or ipaddress>)(PORT=<port>))
(CONNECT_DATA=(SERVICE_NAME=<db service>))
(security=(ssl_server_cert_dn="<certificate_info>")) )
```

For Example — `(description= (address=(protocol=tcps)(port=1522)
(host=129.146.11.169))
(connect_data=(service_name="<dbservice.oracle.com>"))
(security=(ssl_server_cert_dn="<certificate_info>"))`

- b. To connect an ADWC instance with JDBC OCI, use a database URL as shown below:

`jdbc:oracle:oci8:@tnsname` , where `tnsname` is the network name to identify a server or sid or port combination.

For Example — `jdbc:oracle:oci8:@adwc_high`

- **JDBC Properties: (Wallet Specific Properties)**

- `oracle.net.ssl_server_dn_match = true`
- `oracle.net.wallet_location =(SOURCE=(METHOD=file)
(METHOD_DATA=(DIRECTORY=<wallet_directory>))`

For Example — `WALLET_LOCATION = (SOURCE = (METHOD = file)
(METHOD_DATA = (DIRECTORY="/network/admin/ewallet")))`

This property is used to force the distinguished name (dn) of the server to match with its service name.

Creating an Oracle Physical Schema

Create an Oracle physical schema for ADWC using the standard procedure, as described in [Creating a Physical Schema in Administering Oracle Data Integrator](#).

Create a logical schema for this physical schema using the standard procedure, as described in [Creating a Logical Schema in Administering Oracle Data Integrator](#) and associate it in a given context.

Creating and Reverse-Engineering an Oracle Model

This section contains the following topics:

- [Create an Oracle Model](#)
- [Reverse Engineer an Oracle Model](#)

Create an Oracle Model

Create an Oracle Model using the standard procedure, as described in [Creating a Model of Developing Integration Projects with Oracle Data Integrator](#).

Reverse Engineer an Oracle Model

An Oracle model for ADWC 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 and references.

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

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on an Oracle model for ADWC, use the usual procedure, as described in [Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator](#).

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on Oracle model for ADWC with a RKM, use the usual procedure, as described in [Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator](#).

This section details only the fields specific to the Oracle technology. In the Reverse Engineer tab of the Oracle Model, select the KM — RKM Oracle.<project name>.

Designing a Mapping

You can use Oracle ADWC as a source or a target of a mapping. It is also possible to create ETL-style mappings based on the Oracle technology for ADWC.

The recommendations in this section help in the selection of the KM for different situations concerning an Oracle Database Server.

- [Loading Data](#)
- [Extracting data](#)

Loading data

ADWC can be used as a source or target of a mapping. The choice of LKM used in the mappings for loading Knowledge Module, to load data between Oracle and another type of data server determines the working and performance of a mapping.

The following KMs implement optimized methods for loading data from an Oracle database to a target database which is ADWC. Our primary goal is to load data into ADWC. In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technologies involved.

- [Loading Data using Oracle KMs](#)
- [Loading Data using SQL* Loader KMs](#)
- [Loading Data directly into ADWC](#)
- [Loading Oracle Object Storage files into ADWC](#)

Loading Data using Oracle KMs

You can load data into Oracle tables using the following Oracle KMs by designing a mapping where ADWC Oracle data stores can be the target. KMs that can be used for mapping are:

- LKM SQL to Oracle (Built-In)
- IKM Oracle Insert
- IKM Oracle Update
- IKM Oracle Merge
- IKM Oracle Multi-Insert

Loading Data using SQL* Loader KMs

You can also load data into Oracle tables for ADWC using the SQL* Loader KM. LKM File to Oracle (SQLLDR) loads data into Oracle tables from files. You can design a mapping that uses the data stores for an Oracle Schema for ADWC as the target of the mapping, where the data is loaded using the SQL*Loader KM.

Connection Setup for SQL* Loader KMs

Make sure your tnsnames.ora and sqlnet.ora properties are configured to use a Wallet file.

For Example:

1. sqlnet.ora :

```

WALLET_LOCATION=(SOURCE = (METHOD = file)
(METHOD_DATA = (DIRECTORY="<wallet_directory>")))
SSL_SERVER_DN_MATCH=yes

```

2. tnsnames.ora :

- <db_name>_DB_high=(description=(address=(protocol=tcps) (port=<port>)(host=<hostname or ipaddress>)) (connect_data=(service_name=<db_service>)) (security=(ssl_server_cert_dn="<certificate_info>")))
- <db_name>_DB_medium=(description=(address=(protocol=tcps) (port=<port>)(host=<hostname or ipaddress>)) (connect_data=(service_name=<db_service>)) (security=(ssl_server_cert_dn="<certificate_info>")))
- <db_name>_DB_low=(description=(address=(protocol=tcps) (port=<port>)(host=<hostname or ipaddress>)) (connect_data=(service_name=<db_service>)) (security=(ssl_server_cert_dn="<certificate_info>")))

For more details, refer to [Use of an External Password Store to Secure Passwords](#) section of Database Security Guide.

Loading Data directly into ADWC

**Note:**

This section applies only to Data Integration Platform Cloud.

You can use the following knowledge modules for loading data directly into Oracle ADWC.

- [LKM SQL to ADWC External Table](#)
- [LKM SQL to ADWC Copy](#)
- [LKM SQL to ADWC Copy Direct](#)
- [LKM File to ADWC External Table](#)
- [LKM File to ADWC Copy](#)
- [LKM File to ADWC Copy Direct](#)
- [LKM Oracle to ADWC Datapump](#)

For loading data directly into ADWC set the Source of the mapping to be File/HDFS or SQL technology such as Oracle. Target of the mapping has to be Oracle technology, more specifically ADWC database. Object storage staging area is used to temporary store files and this is defined by setting KM option `TEMP_OBJECT_STORAGE_SCHEMA`(logical schema). Some properties such as user/ password are retrieved from Oracle Object Storage data server attached to `TEMP_OBJECT_STORAGE_SCHEMA`logical schema. If you need to create temporary local file in case of SQL source, you have to define its location as `TEMP_FILE_SCHEMA`KM option. If you use transform components, they can be included in the source execution unit in

case of SQL as a source and/or on the target ADWC execution unit. File, as a source does not support source transformations. Direct Copy KMs do not support transformations on the target.

LKM SQL to ADWC External Table

This LKM helps in loading data from SQL source to Oracle ADWC using Oracle Object Storage as intermediate staging. The result from SQL query is first loaded into Oracle Object Storage staging area and then External Table method is used to pull data from Oracle Object Storage. You must set the `TEMP_OBJECT_STORAGE_SCHEMA` and `TEMP_FILE_SCHEMA` KM options to designate temporary storage locations.

You have to use this LKM in combination with Oracle or generic SQL IKM.

The KM invokes the ODI tool `OdiSqlUnload` to unload SQL query data to a local file.

For Example –

```
OdiSqlUnload "-FILE=/tmp/odi_e3a779f8-ddd3-49d2-9575-
a9f3c675b0f6_FILTER_AP.txt" "-DRIVER=oracle.jdbc.OracleDriver" "-
URL=jdbc:oracle:thin:@//slc03sap:1521/flex" "-USER=system" "-
PASS=<@=odiRef.getInfo("SRC_ENCODED_PASS") @>" "-FILE_FORMAT=VARIABLE" "-
FIELD_SEP=," "-ROW_SEP=" "-DATE_FORMAT=MM-DD-YYYY" "-
CHARSET_ENCODING=ISO8859_1"
```

```
SELECT
  PER.PID AS PID ,
  PER.PNAME AS PNAME
FROM
  UT_TD_D_1.PERSON PER
WHERE (PER.PID = 2)
```

The KM then calls ODI tool `OdiObjectStorageUpload` to upload the local file to Oracle Object Storage.

```
OdiObjectStorageUpload "-TRG_LOGICAL_SCHEMA=Object Storage - SRC1" "-
SRC_LOGICAL_SCHEMA=FILE_GENERIC_TMP" "-FILE_NAMES_FILTER=odi_e3a779f8-
ddd3-49d2-9575-a9f3c675b0f6_FILTER_AP.txt" "-OVERWRITE=true"
```

The KM creates a temporary staging external table to pull the data from Oracle Object Storage.

```
BEGIN
  dbms_cloud.create_external_table(
    table_name => 'C$_0FILTER_EXT',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/odi_e3a779f8-
ddd3-49d2-9575-a9f3c675b0f6_FILTER_AP.txt',
    column_list => 'PID NUMBER(2), PNAME VARCHAR2(20)',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  )
```



```
);
END;
```

The KM optionally creates the credential object to connect to Object Storage, but can also re-use an existing one.

```
dbms_cloud.create_credential(
  credential_name => 'ODI_FLEX',
  username => 'tenant15',
  password => 'xxxxxxxx'
);
END;
```

Note:

LKM SQL to ADWC External Table has two different KM options for formatting date/time datatypes:

1. `UNLOAD_DATE_FORMAT`– Use this KM option while unloading SQL query into text file. The formatting syntax is in conform to Java Date and Time Patterns.
2. `DATE_FORMAT`– Use this KM option while loading object storage text file into ADWC. The formatting syntax is in conform to Oracle database syntax for formatting DATE datatype.

If you are loading into `TIMESTAMP` column, use Advanced option `ADD_FORMAT_PROPERTIES` to add the timestampformat property. It is very important to synchronize `UNLOAD_DATE_FORMAT` and `DATE_FORMAT`/timestampformat to use identical format settings, even though they are using different syntax.

Here is an example of loading from source `TIMESTAMP` column into target `TIMESTAMP` column.

```
UNLOAD_DATE_FORMAT = yyyy-MM-dd
UNLOAD_TIMESTAMP_FORMAT = HH:mm:ss.SSS
ADD_FORMAT_PROPERTIES = 'timestampformat' VALUE 'YYYY-MM-DD HH24:MI:SS.FF3'
```

Also, make sure that `TIMESTAMP` column precision allows to store the given format.

KM Options

This KM has the following options:

- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL`- It creates new credentials. If set to False, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST` — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it required by Fixed file format.

- `TEMP_OBJECT_STORAGE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that will be stored in Oracle Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- `CLEANUP_TEMPORARY_OBJECTS`— Set this property to True, if you want temporary objects to be automatically cleaned up.
- `ADD_COMPRESSION` — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.
- `COMPRESSION_TYPE`- It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES` — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

 **Note:**

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` - It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.

- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. Select the required trim option from the provided list of trim options.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties. Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Cleanup

- `CLEANUP_EXTERNAL_TABLE`— Set this property to True, if you want the external table to be automatically cleaned up at the end of the every execution.
- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

SQL Unload Options

- `TEMP_FILE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that is stored before uploading to Oracle Object Storage. This must be a File technology logical schema. The temporary file is stored on local file system where the ODI agent is running.
- `UNLOAD_DATE_FORMAT`— It specifies the output format used for date datatypes. This format follows Java date format standards.
- `UNLOAD_TIMESTAMP_FORMAT`— It specifies the output format used for time datatypes. This format follows Java time format standards.
- `CHARSET_ENCODING` — Use this for character set encoding.
- `FETCH_SIZE` — It denotes the number of rows (records read) requested by ODI agent on each communication with the data server.

LKM SQL to ADWC Copy

This LKM helps in loading data from SQL source to Oracle ADWC using Oracle Object Storage as intermediate staging. The result from SQL query is first loaded into Oracle Object Storage staging area. Then we use `dbms_cloud.copy_data ()` to load data from

Oracle Object Storage into staging table on ADWC. The user must set `TEMP_OBJECT_STORAGE_SCHEMA` and `TEMP_FILE_SCHEMA` KM options to designate temporary storage locations.

You can use this LKM in combination with Oracle or generic SQL IKM.

`OdiSqlUnload` and `OdiObjectStorageUpload` follows the same steps as described in LKM SQL to ADWC External Table. LKM SQL to ADWC Copy is then creating a temporary staging table on ADWC execution unit and pulls the data from Object Store into it using `dbms_cloud.copy_data()` function.

For Example

```
create table STAR.C$_OFILTER
(
  PID NUMBER(2),
  PNAME VARCHAR2(20)
)
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'STAR',
    table_name => 'C$_OFILTER',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/odi_d0b58fb8-
bde5-4ce5-89da-0d258c98380e_FILTER_AP.txt',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  );
END;
```

The KM optionally creates the credential object to connect to Oracle Object Storage, but can also re-use an existing one:

```
BEGIN
  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxx'
  );
END;
```

 **Note:**

LKM SQL to ADWC Copy has two different KM options for formatting date/time datatypes:

- `UNLOAD_DATE_FORMAT`- Use this KM option while unloading SQL query into text file. The formatting syntax is conform to Java Date Patterns.
- `DATE_FORMAT` — Use this KM option while loading object storage text file into ADWC. The formatting syntax is conform to Oracle database syntax for formatting DATE datatype.

If you are loading into `TIMESTAMP` column use Advanced option `ADD_FORMAT_PROPERTIES` to add timestampformat property. It is very important to synchronize `UNLOAD_DATE_FORMAT`and `DATE_FORMAT/timestampformat`to use identical format settings, even though they are using different syntax.

Here is an example of loading from source `TIMESTAMP` column into target `TIMESTAMP` column.

- `UNLOAD_DATE_FORMAT = yyyy-MM-dd HH:mm:ss.SSS`
- `ADD_FORMAT_PROPERTIES = 'timestampformat' VALUE 'YYYY-MM-DD HH24:MI:SS.FF3'`

Also, make sure that `TIMESTAMP` column precision allows the given format to be stored.

KM Options

This KM has the following KM options:

- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL` — It creates new credentials. If set to False, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST` — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `TEMP_OBJECT_STORAGE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that will be stored in Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- `ADD_COMPRESSION` — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE`and `KEEP_SOURCE_FILES`define compression preferences.

- `COMPRESSION_TYPE` — It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES` — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

**Note:**

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J  
MM-DD-YYYYBC  
MM-DD-YYYY  
YYYYMMDD HHMISS  
YYMMDD HHMISS  
YYYY.DDD  
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to True, it trims the specified spaces.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` - This option allows adding custom format properties.

Use the following syntax: <prop1> VALUE '<value1>', '<prop2>' VALUE '<value2>' ...

- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Cleanup

- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want temporary objects to be automatically cleaned up.
- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

SQL Unload Options

- `TEMP_FILE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that is stored before uploading to Object Storage. This must be a File technology logical schema. The temporary file is stored on local file system where the ODI agent is running.
- `UNLOAD_DATE_FORMAT` — It specifies the output format used for date datatypes. This format follows Java date/time format standards.
- `UNLOAD_TIMESTAMP_FORMAT` — It specifies the output format used for time datatypes. This format follows Java time format standards.
- `CHARSET_ENCODING` — Use this for character set encoding.
- `FETCH_SIZE` — It denotes the number of rows (records read) requested by ODI agent on each communication with the data server.

LKM SQL to ADWC Copy Direct

This LKM loads data from SQL source to Oracle ADWC using Oracle Object Storage as intermediate staging. The result from SQL query is first loaded into Oracle Object Storage staging area. Then we use `dbms_cloud.copy_data()` function to load data from Oracle Object Storage directly into ADWC target table. All target columns are loaded irrespective of whether they are mapped or not. Only transformations on source are supported. The user must set `TEMP_OBJECT_STORAGE_SCHEMA` and `TEMP_FILE_SCHEMA` KM options to designate temporary storage locations.

You can use this LKM as a standalone KM as you do not need any IKM.

`OdiSqlUnload` and `OdiObjectStorageUploadSteps` are the same as described in LKM SQL to ADWC External Table. LKM SQL to ADWC Copy Direct pulls the data from Object Store and loads data directly into target using `dbms_cloud.copy_data()` function.

For Example

```
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'STAR',
    table_name => 'PERSON',
```

```

        credential_name =>'ODI',
        file_uri_list =>'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/
odi_16a97fd6-8a61-4303-8f83-81988222b8cb_FILTER_AP.txt',
        format => json_object(
            'type' VALUE 'CSV',
            'skipheaders' VALUE '0',
            'dateformat' VALUE 'AUTO')
    );
END;

```

The KM optionally creates the credential object to connect to Object Storage, but can also re-use an existing one.

```

BEGIN

    dbms_cloud.create_credential(
        credential_name => 'ODI_FLEX',
        username => 'tenant15',
        password => 'xxxxxxxx'
    );

END;

```

Note:

LKM SQL to ADWC Copy Direct has two different KM options for formatting date/time Data types:

- `UNLOAD_DATE_FORMAT` is used while unloading SQL query into text file. The formatting syntax is conform to Java Date and Time Patterns.
- `DATE_FORMAT` is used while loading object storage text file into ADWC. The formatting syntax is conform to Oracle database syntax for formatting DATE data type.

If you are loading into `TIMESTAMP` column use Advanced option

`ADD_FORMAT_PROPERTIES` to add `timestampformat` property. It is very important to synchronize `UNLOAD_DATE_FORMAT` and `DATE_FORMAT/timestampformat` to use identical format settings, even though they are using different syntax.

Here is an example of loading from source `TIMESTAMP` column into target `TIMESTAMP` column.

```

UNLOAD_DATE_FORMAT = yyyy-MM-dd HH:mm:ss.SSS
ADD_FORMAT_PROPERTIES = 'timestampformat' VALUE 'YYYY-MM-DD HH24:MI:SS.FF3'

```

Also, make sure that `TIMESTAMP` column precision allows for the given format to be stored.

KM Options

This KM has the following options:

- `CREDENTIAL_NAME`— It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL`— It creates new credentials. If set to False, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST`— If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `TEMP_OBJECT_STORAGE_SCHEMA`— It specifies the name of logical schema defining the location of the temporary file that will be stored in Oracle Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want temporary objects to be automatically cleaned up.
- `ADD_COMPRESSION`- It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.
- `COMPRESSION_TYPE`- It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES`— Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

 **Note:**

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT`- It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source data store File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
```

```
YYMMDD HHMISS  
YYYY.DDD  
YYYY-MM-DD
```

- **REJECT_LIMIT** — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- **CONVERSION_ERRORS** — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- **TRIM_SPACES** — It helps to trim the leading and trailing spaces of the fields. If set to True it trims the specified spaces.
- **IGNORE_BLANK_LINES** — If set to True, the blank lines are ignored without throwing any error.
- **IGNORE MISSING COLUMNS** — If there are more columns in the field_list than the source files, the extra columns will be stored as null.
- **TRUNCATE_COLUMNS** — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- **ADD_FORMAT_PROPERTIES** — This option allows adding custom format properties.
Use the following syntax: '<prop1>' VALUE '<value1>', '<prop2>' VALUE '<value2>' ...
- **OVERWRITE_FIELD_LIST** — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as field_list parameter of dbms_cloud.create_external_tablefunction call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Target

- **CREATE_TARG_TABLE**— It helps you to create target table. Set this option to True, if you want to create target table before loading.
- **TRUNCATE_TARG_TABLE**— It helps to truncate target table. Set this KM option to True if you want to truncate target table before loading it.
- **DELETE_TARG_TABLE**— It allows you to delete the target table. Set this KM option to True if you want to delete data from target table before loading.

Cleanup

- **CLEANUP_CREDENTIAL** — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if CREATE_CREDENTIALoption is also set to True.

SQL Unload Options

- **TEMP_FILE_SCHEMA**— It specifies the name of logical schema defining the location of the temporary file that is stored before uploading to Object Storage. This must be a File technology logical schema. The temporary file is stored on local file system where the ODI agent is running.

- UNLOAD_DATE_FORMAT— It specifies the output format used for date data types. This format follows Java date/time format standards.
- UNLOAD_TIMESTAMP_FORMAT— It specifies the output format used for time data types. This format follows Java time format standards.
- CHARSET_ENCODING— Use this for character set encoding.
- FETCH_SIZE- It denotes the number of rows (records read) requested by ODI agent on each communication with the data server.

LKM File to ADWC External Table

This LKM helps to load data from local or HDFS file source to Oracle ADWC using Oracle Object Storage as intermediate staging. The source file is first loaded into Oracle Object Storage staging area. Then we use External Table method to pull data from Object Storage. You must set TEMP_OBJECT_STORAGE_SCHEMA KM option to designate temporary storage location.

You have to use this LKM in combination with Oracle or generic SQL IKM.

The KM invokes ODI tool OdiObjectStorageUpload to upload the local file to Object Storage.

For Example

```
OdiObjectStorageUpload "-TRG_LOGICAL_SCHEMA=Object Storage - SRC1" "-
SRC_LOGICAL_SCHEMA=FILE_GENERIC_TMP" "-
FILE_NAMES_FILTER=person_no_header.csv" "-OVERWRITE=true"
```

The KM then creates a temporary staging external table to pull the data from Oracle Object Storage.

```
BEGIN
  dbms_cloud.create_external_table(
    table_name =>'C$_OPER_EXT',
    credential_name =>'ODI',
    file_uri_list =>'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/person_no_header.csv',
    column_list => 'PID NUMBER(2,0), PNAME VARCHAR2(20)',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  );
END;
```

The KM optionally creates the credential object to connect to Oracle Object Storage, but can also re-use an existing one.

```
BEGIN
  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxxx'
```

```
);  
END;
```

KM Options

This KM has the following options:

- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL` — It creates new credentials. If set to False, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST` — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

Note:

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `TEMP_OBJECT_STORAGE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that will be stored in Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- `ADD_COMPRESSION` — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.
- `COMPRESSION_TYPE` — It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES` — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

Note:

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source data store File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and AUTO implies compression type is auto-detected.

- `DATE_FORMAT` — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to `True`, it trims the specified spaces.
- `IGNORE_BLANK_LINES` — It set to `True` the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Cleanup

- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to `True`, if you want temporary objects to be automatically cleaned up.
- `CLEANUP_CREDENTIAL` — Set this property to `True`, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to `True`.

LKM File to ADWC Copy

This LKM helps to load data from local or HDFS file source to Oracle ADWC using Oracle Object Storage as intermediate staging. The source file is first loaded into Oracle Object Storage staging area. Then we use `dbms_cloud.copy_data()` to load

data from Oracle Object Storage into staging table on ADWC. You must set `TEMP_OBJECT_STORAGE_SCHEMA` KM option to designate temporary storage location.

You can use this LKM in combination with Oracle or generic SQL IKM.

`OdiObjectStorageUpload` happens similar to LKM File to ADWC External Table.

LKM File to ADWC Copy then creates a temporary staging table on ADWC execution unit and pulls the data from Object Store into it using `dbms_cloud.copy_data()` function.

For Example

```
create table STAR.C$_OPER
(
  PID NUMBER(2,0),
  PNAME VARCHAR2(20)
)
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'STAR',
    table_name => 'C$_OPER',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/person_no_header.csv',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  );
END;
```

The KM optionally creates the credential object to connect to Object Storage, but can also re-use an existing one.

```
BEGIN

  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxx'
  );
END;
```

KM Options

This KM has the following options:

- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL` — It creates new credentials. If set to False, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST` — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `TEMP_OBJECT_STORAGE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that will be stored in Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- `ADD_COMPRESSION` — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.
- `COMPRESSION_TYPE` — It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES` — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

 **Note:**

`gzip` supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.

- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to `True`, it trims the specified spaces.
- `IGNORE_BLANK_LINES` — If set to `True`, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: `'<prop1>' VALUE '<value1>', '<prop2>' VALUE '<value2>' ...`
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Cleanup

- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to `True`, if you want temporary objects to be automatically cleaned up.
- `CLEANUP_CREDENTIAL` — Set this property to `True`, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to `True`.

LKM File to ADWC Copy Direct

This KM is helpful to load data from local or HDFS file source to Oracle ADWC using Oracle Object Storage as intermediate staging. The source file is first loaded into Oracle Object Storage staging area. Then we use `dbms_cloud.copy_data()` function to load data from Object Storage directly into ADWC target table. Source attributes must match target column names. Either use data entities with matching attributes or set option `GENERATE_FIELD_LIST` to `false`. All target columns are loaded, irrespective of their mapping.

You must set `TEMP_OBJECT_STORAGE_SCHEMA` KM option to designate temporary storage location. LKM File to ADWC Copy Direct does not support any transformations.

You can use this LKM as a standalone KM as you do not need an IKM for its implementation. `OdiObjectStorageUpload` happens similar to LKM File to ADWC External Table. LKM SQL to ADWC Copy Direct pulls the data from Oracle Object Storage and loads data directly into target using `dbms_cloud.copy_data()` function.

For Example

```
BEGIN
    dbms_cloud.copy_data(
        schema_name => 'STAR',
```



```

        table_name =>'PERSON',
        credential_name =>'ODI',
        file_uri_list =>'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/person_no_header.csv',
        format => json_object(
            'type' VALUE 'CSV',
            'skipheaders' VALUE '0',
            'dateformat' VALUE 'AUTO')
    );
END;
```

The KM optionally creates the credential object to connect to Oracle Object Storage, but can also re-use an existing one.

```

BEGIN

    dbms_cloud.create_credential(
        credential_name => 'ODI_FLEX',
        username => 'tenant15',
        password => 'xxxxxxxx'
    );

END;
```

KM Options

This KM has the following options:

- **CREDENTIAL_NAME** — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- **CREATE_CREDENTIAL** — It creates new credentials. If set to False, ODI reuses the existing credentials.
- **GENERATE_FIELD_LIST** — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

Note:

You have to always generate the `field_list` clause as it is required by Fixed file format.

- **TEMP_OBJECT_STORAGE_SCHEMA** — It specifies the name of logical schema defining the location of the temporary file that will be stored in Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- **CLEANUP_TEMPORARY_OBJECTS** — Set this property to True, if you want temporary objects to be automatically cleaned up.
- **ADD_COMPRESSION** — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.

- `COMPRESSION_TYPE` — It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES` — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

**Note:**

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source data store File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types, or auto. Empty value implies no compression and AUTO implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option AUTO searches for the following formats

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to True, it trims the specified spaces.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.

Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...

- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Target

- `CREATE_TARG_TABLE` — It helps you to create target table. Set this option to True, if you want to create target table before loading.
- `TRUNCATE_TARG_TABLE` — It helps to truncate target table. Set this KM option to True if you want to truncate target table before loading it.
- `DELETE_TARG_TABLE` — It allows you to delete the target table. Set this KM option to True if you want to delete data from target table before loading.

Cleanup

- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

LKM Oracle to ADWC Datapump

This KM is helpful to load data from Oracle On-premises products to Oracle ADWC. You can use it in combination with Oracle or generic SQL IKM.

Working

First source table is exported into local Datapump dump file. The export file is created in a local directory specified by `TEMP_FILE_SCHEMA` logical schema. A unique Export file name is generated as `odi_<ODI_session_id>_<mapping_node>.dmp`. Export log name is `odi_<ODI_session_id>_<mapping_node>_exp.log`. The export file is then uploaded to Oracle Object storage bucket specified by `TEMP_OBJECT_STORAGE_SCHEMA` logical schema. It is then imported to ADWC database into temporary staging table `C$_alias`. Import log name is `odi_<ODI_session_id>_<mapping_node>_imp.log`. Finally, the mapping IKM integrates the staging table into target. Transformations on target execution unit are supported.

Note:

- ODI agent has to run on the same host as the source database to be able to access the dump/log files.
- Currently the KM does not support multi file loading from Oracle Object Storage. Hence you cannot set `FILESIZE` and `PARALLEL` options. `dbms_datapump` package is used for export and import.

Example of Export code

```
declare
  h1 number;
```

```

j_status varchar2(200);
begin
  dbms_output.enable();
  h1 := dbms_datapump.open('EXPORT','SCHEMA',NULL,'ODI_EXPORT','LATEST');
  dbms_datapump.add_file(
    handle => h1,
    filename => 'odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP.dmp',
    directory => 'ODI_DIR',
    filetype => dbms_datapump.KU$_FILE_TYPE_DUMP_FILE,
    reusefile => 1);
  dbms_datapump.add_file(
    handle => h1,
    filename => 'odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP_exp.log',
    directory => 'ODI_DIR',
    filetype => dbms_datapump.KU$_FILE_TYPE_LOG_FILE);
  dbms_datapump.metadata_filter( h1, 'SCHEMA_LIST', q'|'UT_TD_D_1'|' );
  dbms_datapump.metadata_filter( h1, 'NAME_LIST', q'|'PERSON_SRC'|',
'TABLE' );
  dbms_datapump.metadata_filter( h1, 'INCLUDE_PATH_LIST', q'|'TABLE'|' );
  dbms_datapump.set_parameter(h1, 'COMPRESSION', 'METADATA_ONLY');
  dbms_datapump.set_parameter(h1, 'COMPRESSION_ALGORITHM', 'BASIC');
  dbms_datapump.start_job(h1);
  dbms_datapump.wait_for_job(h1, j_status);
exception
  when others then
    ....
end;
```

Example of Import code

```

declare
  h1 number;
  j_status varchar2(200);
begin
  h1 := dbms_datapump.open('IMPORT','FULL',NULL,'ODI_IMPORT','LATEST');
  dbms_datapump.add_file(
    handle => h1,
    filename => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/
odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP.dmp',
    directory => 'ODI',
    filetype => dbms_datapump.KU$_FILE_TYPE_URIDUMP_FILE);
  dbms_datapump.add_file(
    handle => h1,
    filename => 'odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP_imp.log',
    directory => 'DATA_PUMP_DIR',
    filetype => dbms_datapump.KU$_FILE_TYPE_LOG_FILE);
  dbms_datapump.metadata_remap( h1, 'REMAP_SCHEMA', 'UT_TD_D_1', 'STAR');
  dbms_datapump.metadata_remap( h1, 'REMAP_TABLE', 'PERSON_SRC',
'CS_OPER');
  dbms_datapump.set_parameter(h1,'TABLE_EXISTS_ACTION','SKIP');
  dbms_datapump.set_parameter(h1,'PARTITION_OPTIONS','MERGE');
  dbms_datapump.metadata_transform( h1, 'SEGMENT_ATTRIBUTES', 0);
  dbms_datapump.start_job(h1);
```

```

dbms_datapump.wait_for_job(h1, j_status);
dbms_cloud.put_object(
    credential_name => 'ODI',
    object_uri => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/
odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP_imp.log',
    directory_name => 'DATA_PUMP_DIR',
    file_name =>
'odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP_imp.log');
exception
    when others then
        ....
end;

```

KM Options

This KM has the following options:

Source Export

- **TEMP_FILE_SCHEMA**— It specifies the name of logical schema defining the location of the temporary export file that is stored before uploading to Oracle Object Storage. This must be a File technology logical schema. Oracle directory is created based on this location. The temporary file is stored on a file system accessible by the source database.
- **ORACLE_DIRECTORY_NAME** — It specifies the name of Oracle directory used by Datapump to store export file and logs on Source. The default value is ODI_DIR.

Note:

Use only uppercase for directory names.

- **COMPRESSION** — It specifies which data to compress before writing to the dump file set. You can specify any of the following compression options:
 - **ALL** — enables compression for the entire export operation. You must enable **Oracle Advanced Compression** option for making this type of compression.
 - **DATA_ONLY** — When you select this option, entire data is written to the dump file in a compressed format. You must enable Oracle Advanced Compression option for making this type of compression.
 - **METADATA_ONLY** — When you select this option, entire metadata is written to the dump file in compressed format. This is the default option for **COMPRESSION**.
 - **NONE** — It disables compression for the entire export operation.
- **COMPRESSION_ALGORITHM**— It specifies the compression algorithm to be used when compressing dump file data. You can specify one of the following compression algorithm options:
 - **BASIC** — Offers a good combination of compression ratios and speed; the algorithm used is the same as in previous versions of Oracle Datapump.
 - **LOW** — Least impact on export throughput and suited for environments where CPU resources are the limiting factor.

- MEDIUM — Recommended for most environments. This option is similar to the BASIC option that provides a good combination of compression ratios and speed, but it uses a different algorithm when compared to BASIC algorithm.
- HIGH — Best suited for situations in which dump files are copied over slower networks where the limiting factor is the network speed.
- CREATE_VIEW_ON_SOURCE — It allows you to create view on source. Set this property to False only if you have a single source with no transformations or a simple filter.
- CREATE_VIEW_ON_SOURCE_TEMP_SCHEMA — It allows you to create view on source work schema. Set this property to True if you want to create view on source Work Schema.

 **Note:**

- Work Schema user should have select privileges to all the source tables participating in the view select statement for the view to be valid. If set to False the view is created in the schema of the currently connected user.
- You cannot remap a SYSTEM schema, which means if set to False you must not be connected as SYSTEM. Because of restrictions in both the cases, we provide this option.

- USE_GROUP_PARTITION_TABLE_DATA — Use parameter DATA_OPTIONS=GROUP_PARTITION_TABLE_DATA on export. GROUP_PARTITION_TABLE_DATA value for DATA_OPTIONS parameter is only available for Oracle 12.2 versions or later.

 **Note:**

Set this option to False, for any previous Oracle versions.

Target Import

- CREDENTIAL_NAME — It provides credential name to connect to Object Storage. The default value is ODI.
- CREATE_CREDENTIAL — It helps to create new credential. If set to False ODI will reuse existing credentials.
- TEMP_OBJECT_STORAGE_SCHEMA — It denotes the logical schema for temporary Oracle Object Storage location. Specify the name of logical schema defining the location of the temporary export file that will be stored in Oracle Object Storage staging area. This must be an Object Storage technology logical schema.

Cleanup

- CLEANUP_TEMPORARY_OBJECTS — Set this property to True, if you want to clean up the temporary objects automatically.
- CLEANUP_CREDENTIAL — Set this property to True, if you want to clean up the credential object automatically at the end of the every execution. Cleanup will happen only if CREATE_CREDENTIALoption is also set to true.

Loading Oracle Object Storage files into ADWC



Note:

This section applies only to Data Integration Platform Cloud.

You can use the following knowledge modules for loading Oracle Object Storage files into Oracle ADWC:

- [LKM Object Storage to ADWC Copy](#)
- [LKM Object Storage to ADWC Copy Direct](#)
- [LKM Object Storage to ADWC External Table](#)

When you load data from Oracle Object Storage to ADWC, target of the mapping has to be Oracle technology as there is no specific technology for ADWC. Object Storage Bucket represents Object Storage physical schema. Some properties such as Heading and Delimiter are retrieved from source Data Store. Other properties such as user name and password are retrieved from Object Storage Data Server. If you use transform components, they need to be moved to the target (Oracle) execution unit. No transformations on source are supported.

LKM Object Storage to ADWC Copy

This KM helps to load data from Oracle Cloud Object Storage to Oracle ADWC. It is using `dbms_cloud.copy_data()` function to load data into a staging table. LKM Object Storage to ADWC Copy is assigned to an AP node. It is used in moving data from Oracle Object Storage file to an ADWC table. You can use it in combination with Oracle or generic SQL IKM.

For Example

LKM Object Storage to ADWC External Table is creating a temporary staging external table to pull the data from Object Store.

```
BEGIN
  dbms_cloud.create_external_table(
    table_name => 'C$_OPER_EXT',
    credential_name => 'ODI_FLEX',
    file_uri_list => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsprod/bucket_tenant15/person.csv',
    column_list => 'PID NUMBER(2,0),
PNAME VARCHAR2(20)',
    field_list => 'PID, PNAME',
    format => json_object('type' VALUE 'CSV', 'skipheaders' VALUE '1')
  );
END;
```

It also optionally creates the credential object, but can also re-use an existing one.

```
BEGIN
```

```

dbms_cloud.create_credential(
  credential_name => 'ODI_FLEX',
  username => 'tenant15',
  password => 'xxxxxxx'
);
END;

```

KM Options

This KM has the following options:

- **CREDENTIAL_NAME** — It provides credential name to connect to Object Storage. The default value is ODI.
- **CREATE_CREDENTIAL** — It helps to create new credential. If set to False ODI will reuse existing credentials.
- **GENERATE_FIELD_LIST** — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.



Note:

You have to always generate the `field_list` clause as it is required by Fixed file format.

- **DELIMITED_FILE_FORMAT** — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- **COMPRESSION** — It specifies the compression method of the source file. It can have values `nil` or `auto`. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- **DATE_FORMAT** — It helps to set specific date format. The default format option `AUTO` searches for the following formats:


```

J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD

```
- **REJECT_LIMIT** — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- **CONVERSION_ERRORS** — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- **TRIM_SPACES** — It helps to trim the leading and trailing spaces of the fields. If set to `True` it trims the specified spaces.
- **IGNORE_BLANK_LINES** — It set to `True` the blank lines are ignored without throwing any error.

- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.
- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: `<'prop1>' VALUE '<value1>', '<prop2>' VALUE '<value2>' ...`
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.
For more details, refer to [DBMS_CLOUD](#) package documentation for more information.
- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want temporary objects to be automatically cleaned up.
- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

LKM Object Storage to ADWC Copy Direct

This LKM is helpful to load data from Oracle Object Storage to Oracle ADWC. It is using `dbms_cloud.copy_data()` function. This LKM can be used as standalone as no IKM is needed. It loads data directly into target table. Source attributes must match target column names. Either use data entities with matching attributes or set option `GENERATE_FIELD_LIST = false`. All target columns are loaded irrespective of whether they are mapped or not.



Note:

LKM Object Storage to ADWC Copy Direct does not support any transformations.

The LKM Object Storage to ADWC Copy Direct is assigned to an AP node. This is used for moving files/objects from Oracle Object Storage to an ADWC table.

For Example

LKM Object Store to ADWC Copy Direct pulls the data from Oracle Object Storage and loads data directly into target using `dbms_cloud.copy_data()` function.

```
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'ODI',
    table_name => 'PERSON',
    credential_name => 'ODI_FLEX',
    file_uri_list => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsprod/bucket_tenant15/person.csv',
    field_list => 'PID, PNAME',
```

```

        format => json_object('type' VALUE 'CSV', 'skipheaders' VALUE '1')
    );
END;
```

It also optionally creates the credential object, but can also re-use an existing one.

```

BEGIN

    dbms_cloud.create_credential(
        credential_name => 'ODI_FLEX',
        username => 'tenant15',
        password => 'xxxxxxxx'
    );
END;
```

KM Options

This KM has the following options:

- **CREATE_TARG_TABLE** — It helps you to create target table. Set this option to True, if you want to create target table before loading.
- **CREDENTIAL_NAME** — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- **CREATE_CREDENTIAL** — It creates new credentials. If set to False, ODI reuses the existing credentials.
- **TRUNCATE_TARG_TABLE** — It helps to truncate target table. Set this KM option to True if you want to truncate target table before loading it.
- **DELETE_TARG_TABLE** — It allows you to delete the target table. Set this KM option to True if you want to delete data from target table before loading.
- **GENERATE_FIELD_LIST** — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

Note:

You have to always generate the `field_list` clause as it is required by Fixed file format.

- **DELIMITED_FILE_FORMAT** — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- **COMPRESSION** — It specifies the compression method of the source file. It can have values `nil` or `auto`. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- **DATE_FORMAT** — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```

J
MM-DD-YYYYBC
```

```
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- **REJECT_LIMIT** — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- **CONVERSION_ERRORS** — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- **TRIM_SPACES** — It helps to trim the leading and trailing spaces of the fields. If set to True it trims the specified spaces.
- **IGNORE_BLANK_LINES** — If set to True, the blank lines are ignored without throwing any error.
- **IGNORE MISSING COLUMNS** — If there are more columns in the field_list than the source files, the extra columns will be stored as null.
- **TRUNCATE_COLUMNS** — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.
- **ADD_FORMAT_PROPERTIES** — This option allows adding custom format properties.

Use the following syntax: '**<prop1>**' VALUE '**<value1>**', '**<prop2>**' VALUE '**<value2>**' ...

- **OVERWRITE_FIELD_LIST** — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as field_list parameter of dbms_cloud.create_external_table function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

- **CLEANUP_TEMPORARY_OBJECTS** — Set this property to True, if you want temporary objects to be automatically cleaned up.
- **CLEANUP_CREDENTIAL** — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if **CREATE_CREDENTIAL** option is also set to True.

LKM Object Storage to ADWC External Table

This KM is helpful to load data from Oracle Object Storage to Oracle ADWC using External Table method. You can use this LKM in combination with Oracle or generic SQL IKM.

The LKM is assigned to an AP node, so that it can move Oracle Object Storage file to Oracle ADWC table.

For Example —

LKM Object Storage to ADWC External Table is creating a temporary staging external table to pull the data from Oracle Object Storage.

```
BEGIN
  dbms_cloud.create_external_table(
```

```

        table_name => 'C$_OPER_EXT',
        credential_name => 'ODI_FLEX',
        file_uri_list => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsprod/bucket_tenant15/person.csv',
        column_list => 'PID NUMBER(2,0),
PNAME VARCHAR2(20)',
        field_list => 'PID, PNAME',
        format => json_object('type' VALUE 'CSV', 'skipheaders' VALUE '1')
    );
END;

```

It also optionally creates the credential object, but can also re-use an existing one.

```

BEGIN

    dbms_cloud.create_credential(
        credential_name => 'ODI_FLEX',
        username => 'tenant15',
        password => 'xxxxxxxx'
    );

END;

```

KM Options

This KM has the following options:

- **CREDENTIAL_NAME** — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- **CREATE_CREDENTIAL** — It creates new credentials. If set to False, ODI reuses the existing credentials.
- **GENERATE_FIELD_LIST** — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

Note:

You have to always generate the `field_list` clause as it is required by Fixed file format.

- **DELIMITED_FILE_FORMAT** — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- **COMPRESSION** — It specifies the compression method of the source file. It can have values `nil` or `auto`. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- **DATE_FORMAT** — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```

J
MM-DD-YYYYBC

```

MM-DD-YYYY
 YYYYMMDD HHMISS
 YYMMDD HHMISS
 YYYY.DDD
 YYYY-MM-DD

- **REJECT_LIMIT** — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- **CONVERSION_ERRORS** — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- **TRIM_SPACES** — It helps to trim the leading and trailing spaces of the fields. If set to True it trims the specified spaces.
- **IGNORE_BLANK_LINES** — It set to True the blank lines are ignored without throwing any error.
- **IGNORE MISSING COLUMNS** — If there are more columns in the field_list than the source files, the extra columns will be stored as null.
- **TRUNCATE_COLUMNS**— If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.
- **ADD_FORMAT_PROPERTIES** — This option allows adding custom format properties.

Use the following syntax: '<prop1>' VALUE '<value1>', '<prop2>' VALUE '<value2>' ...

- **OVERWRITE_FIELD_LIST** — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as field_list parameter of dbms_cloud.create_external_table function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

- **CLEANUP_TEMPORARY_OBJECTS** — Set this property to True, if you want temporary objects to be automatically cleaned up.
- **CLEANUP_CREDENTIAL**— Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if **CREATE_CREDENTIAL** option is also set to True.

Extracting data

In ODI, you can extract data from ADWC via JDBC using Oracle KMs.

You can design a mapping that uses the Data Stores for an ADWC Schema as the source of the mapping, where you can extract data via JDBC using the following Oracle KMs:

- Extract data from ADWC and load to on-premise Oracle Table through
 - LKM SQL to Oracle (Built-In)
 - IKM Oracle Insert
- Extract data from ADWC and load to on-premise File through
 - LKM SQL Multi-Connect

4

Oracle Autonomous Transaction Processing

This chapter describes how to work with Autonomous Transaction Processing (ATP) in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Prerequisites](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Oracle Model](#)
- [Designing a Mapping](#)
- [Best Practices for Working with ATP](#)

Introduction

Oracle Autonomous Transaction Processing Cloud Service is a fully managed, pre-configured database environment.

You do not need to configure or manage any hardware, or install any software. After provisioning, you can scale the number of CPU cores or the storage capacity of the database at any time without impacting availability or performance. Autonomous Transaction Processing handles creating the database, along with performing the maintenance tasks such as backing up, patching, upgrading and tuning the database.

Oracle Data Integrator (ODI) seamlessly integrates with ATP. By integrating ODI with ATP, you can get a self-driving, self-securing, self-repairing database service that can instantly scale to meet demands of mission critical applications.

Concepts

The Oracle ATP concepts map the Oracle Data Integrator concepts as follows: An Oracle ATP 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 ATP instance. All connections to the ATP require the use of a credential file to manage public key security credentials. ODI users can use a Thin or OCI connection to access ATP.

Knowledge Modules

Oracle Data Integrator provides the following Knowledge Modules (KM) for loading data into Oracle ATP. The KMs use Oracle specific features. It is also possible to use the generic SQL KMs with Oracle ATP. ODI uses the same optimized KMs for both Oracle ADWC and Oracle ATP cloud services.

Table 4-1 ATP Knowledge Modules

Knowledge Module	Description
LKM SQL to Oracle (Built-In)	Loads data from any ANSI SQL-92 source database to an Oracle staging area.
LKM SQL Multi-Connect	Enables the use of multi-connect IKM for target table. Built-in IKM.
LKM File to Oracle (SQLLDR)	Loads data from a file to an Oracle staging area using the SQL*Loader command line utility.
IKM SQL to File Append	Integrates data in a target file from any ANSI SQL-92 compliant staging area in replace mode.
IKM Oracle Insert	Integrates data into an Oracle target table in append mode. The data is loaded directly in the target table with a single INSERT SQL statement. Built-in KM.
IKM Oracle Update	Integrates data into an Oracle target table in incremental update mode. The data is loaded directly into the target table with a single UPDATE SQL statement. Built-in KM.
IKM Oracle Merge	Integrates data into an Oracle target table in incremental update mode. The data is loaded directly into the target table with a single MERGE SQL statement. Built-in KM.
IKM Oracle Multi-Insert	Integrates data from one source into one or many Oracle target tables in append mode, using a multi-table insert statement (MTI). This IKM can be utilized in a single mapping to load multiple targets. Built-in KM.
RKM Oracle	Reverse-engineers tables, views, columns and creates data models to use as targets or sources in Oracle Data Integrator mappings.

Table 4-1 (Cont.) ATP Knowledge Modules




Knowledge Module	Description
LKM SQL to ADWC External Table	Loads data from SQL source to Oracle ADWC and Oracle ATP using Oracle Object Storage as intermediate staging. You have to use this LKM in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM SQL to ADWC Copy	Loads data from SQL source to Oracle ADWC and Oracle ATP using Oracle Object Storage as intermediate staging. You have to use this LKM in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM SQL to ADWC Copy Direct	Loads data from SQL source to Oracle ADWC and Oracle ATP using Oracle Object Storage as intermediate staging. You can use this LKM as a standalone KM as you do not need any IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.

Table 4-1 (Cont.) ATP Knowledge Modules




Knowledge Module	Description
LKM File to ADWC External Table	Loads data from local or HDFS file source to Oracle ADWC and Oracle ATP using Oracle Object Storage as intermediate staging. You have to use this LKM in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM File to ADWC Copy	Loads data from local or HDFS file source to Oracle ADWC and Oracle ATP using Oracle Object Storage as intermediate staging. You have to use this LKM in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM File to ADWC Copy Direct	Loads data from local or HDFS file source to Oracle ADWC and Oracle ATP using Oracle Object Storage as intermediate staging. You can use this LKM as a standalone KM as you do not need an IKM for its implementation.
	 Note: This KM applies only to Data Integration Platform Cloud.

Table 4-1 (Cont.) ATP Knowledge Modules




Knowledge Module	Description
LKM Oracle to ADWC Datapump	Loads data from Oracle On-premises products to Oracle ADWC and Oracle ATP. You can use it in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM Object Storage to ADWC Copy	Loads data from Oracle Cloud Object Storage to Oracle ADWC and Oracle ATP. You can use it in combination with Oracle or generic SQL IKM.
	 Note: This KM applies only to Data Integration Platform Cloud.
LKM Object Storage to ADWC Copy Direct	Loads data from Oracle Cloud Object Storage to Oracle ADWC and Oracle ATP. You can use this LKM to move files/objects from Oracle Object Storage to an ADWC or ATP table.
	 Note: This KM applies only to Data Integration Platform Cloud.

Table 4-1 (Cont.) ATP Knowledge Modules

Knowledge Module	Description
LKM Object Storage to ADWC External Table	Loads data from Oracle Object Storage to Oracle ADWC and Oracle ATP using External Table method. You can use this LKM in combination with Oracle or generic SQL IKM.

 **Note:**

This KM applies only to Data Integration Platform Cloud.

Prerequisites

The following prerequisites are essential for connecting to ATP environment. Make sure you go through the following prerequisites, before connecting to ATP environment.

 **Note:**

The following prerequisites are common for both ODI Studio and ODI Agent.

Wallet Configuration

All connections to ATP require the use of a credential file to manage public key security credentials. This file is used to store authentication and signing credential, including private key, certificates, and trusted certificates needed by SSL. Oracle Wallet (ewallet.p12) or credential zip file (auto login wallet) provides a simple and easy method to manage database credentials across multiple domains. It allows you to update database credentials by updating the Wallet instead of having to change individual data source definitions. To connect to the ATP, applications need access to the Oracle wallet.

For more details on wallet, refer to **Securing Passwords in Application Design** section of [Managing Security for Application Developers](#) in Database Security Guide.

The JDBC properties require a Wallet or Credential file location.

- Get the wallet file (ewallet.p12) or credential zip file (auto login wallet) from ATP wallet location and place it in a local directory accessible to both ODI Studio and ODI Agent. The default location of the wallet file is `<homedir>/ .odi/oracledi/ewallet`. For auto login wallet, upload the credential zip as wallet file.

Java Security configuration

 **Note:**

Steps listed below are not required for JDK1.8.0_u161 or later versions.

Update the file `java.security`, from the location `JDK_HOME\jre\lib\security` and add the following lines of code, as shown below:

- `security.provider.10=sun.security.mscaapi.SunMSCAPI`
- `security.provider.11=oracle.security.pki.OraclePKIProvider`

```

security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=sun.security.ec.SunEC
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=sun.security.mscaapi.SunMSCAPI
security.provider.11=oracle.security.pki.OraclePKIProvider

```

Preparing for OCI Connection

Follow the below steps, to create a OCI Connection :

1. Download and install Oracle Instant Client: Version 12.2.0.1.0. For more instructions, see <http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html>
2. After installing the Oracle Instant Client, add its folder in the environment variable called `LD_LIBRARY_PATH`: `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/oracle/instantclient_12_2`
3. Unzip the credential file into a new directory: i.e. `/home/oracle/wallet`
4. Navigate to the new wallet directory, and modify the `sqlnet.ora` file as:
 - specify the wallet location.

For Example —

```

WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
(DIRECTORY=/home/oracle/wallet)))

```

- add the following two SSL parameters in the `sqlnet.ora` file:

```

SSL_SERVER_DN_MATCH= TRUE
SSL_VERSION = 1.2

```

- Change or set `TNS_ADMIN` to the location or directory where the unzipped credential files are located. For instance, if your wallet file was unzipped to a directory called `/home/oracle/wallet` then, set `TNS_ADMIN` parameter as follows:

```
TNS_ADMIN=/home/oracle/wallet
```

Agent Configurations for OCI

Perform the following configurations to use OCI, to connect ODI Agents (Standalone and J2EE) with ATP:

- In Local (No Agent), ODI Standalone agent and J2EE Agent, set the instant client folder for the `LD_LIBRARY_PATH` parameter before launching ODI Studio. For example, export `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/oracle/instantclient_12_2`

Note:

The existing value of jars or directories mentioned in the `LD_LIBRARY_PATH` parameter should be consistent to the current version of instant client jars . If not, you may get an error.

- For ODI Standalone Agent, configure the instant client library path in `ODI_ADDITIONAL_JAVA_OPTIONS` variable, present in `instance.sh/cmd` of the standalone domain. The `instance.sh` file is located at : `<DOMAIN_HOME>config/fmwconfig/components/ODI/<agent-name>/bin` . For Example,

```
ODI_ADDITIONAL_JAVA_OPTIONS="${ODI_ADDITIONAL_JAVA_OPTIONS} -
Djava.library.path=${ODI_HOME}/../sdk/lib:/home/oracle/
instantclient_12_2"
```

- For ODI J2EE Agent, set `JAVA_OPTIONS` parameter to `$JAVA_OPTIONS -Djava.library.path=/home/oracle/instantclient_12_2`. For example, export `JAVA_OPTIONS="$JAVA_OPTIONS -Djava.library.path=/home/oracle/instantclient_12_2"`

Setting up the Topology

Note:

Please note the ATP data server is created under Oracle Technology.

Setting up the Topology consists of:

- [Creating an Oracle Data Server](#)
- [Creating an Oracle Physical Schema](#)

Creating an Oracle Data Server

Create a data server for ATP using the standard procedure, as described in Creating a Data Server of Administering Oracle Data Integrator. This section details only the properties required to be set in the data server created under Oracle technology for ATP:

1. In the **Definition** tab:

Data Server

- **Name** : Enter a name for the data server definition.
- **Instance / dblink (Data Server)** : TNS Alias used for this Oracle instance. It will be used to identify the Oracle instance when using database links and SQL*Loader.

Connection

- **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.
- **JNDI Connection** : Select this check-box to configure the JNDI connection settings. Navigate to the JNDI tab, and fill in the required fields.

 **Note:**

JNDI connection field is not applicable for ATP.

- **Use Credential File**

 **Note:**

Please note it is required to use a credential file with ATP.

Select this check box to upload the connection details directly from a pre-configured wallet file¹ or credential zip file. The above JNDI Connection check-box is disabled and the following fields with respect to Wallet feature appear:

Credential Details

 **Note:**

This section applies only to Data Integration Platform Cloud.

¹ In ODI, you can directly upload connection details from a credential file (cwallet.sso) or a password protected wallet file (ewallet.p12).

- **Credential File:** Click the browse icon present beside the **Credential File** text box, to browse for the location of the required wallet file containing the connection details.

By default, the credential location is populated to point to the same wallet file (ewallet.p12) or credential zip file (auto login wallet) that is used for login which is present in `<homedir>/<odi>/oracledi/ewallet`. You can also browse for any other location where this credential file is stored with the required database connection details.

 **Note:**

ODI supervisor can distribute the password protected wallet and credential zip file to other ODI users.

- Click the **Connection Details** drop down arrow to choose the required connection URL from the list of available connection URLs retrieved from `tnsnames.ora`.
 - Upon selecting the required connection URL, JDBC Driver, JDBC URL, Username and Password fields are disabled and the associated username, password, jdbc URL and jdbc driver details are auto-populated with credentials retrieved from the pre-configured wallet file or credential zip file. You can change them only through the list of connections available in the wallet file (ewallet.p12) or credentials zip (tnsnames.ora).
 - Click **Save**, to save the Data Server details
 - Click **Test Connection**, to test the established connection
2. The ATP Data Server (using the Oracle Technology) supports JDBC. Fill-in the following details of the **JDBC** tab:

ODI users can use a Thin or OCI connection to access ATP.

 **Note:**

The JDBC tab will be auto populated with the JDBC information contained in the credential file. You can review the auto-populated information in the JDBC tab and make edits (only if required).

- **JDBC Driver** : Name of the JDBC driver used for connecting to the data server. `oracle.jdbc.OracleDriver`
- **JDBC URL** : URL allowing you to connect to the data server.

Format of the 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>`

- a. To connect an ATP instance with the Oracle JDBC thin driver, use a database URL as shown below:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)
(HOST=<hostname or ipaddress>)(PORT=<port>))
```



```
(CONNECT_DATA=(SERVICE_NAME=<db service>))
(security=(ssl_server_cert_dn="<certificate_info>")) )
```

For Example —(description= (address=(protocol=tcps)(port=1522)
(host=129.146.11.169))
(connect_data=(service_name="<dbservice.oracle.com>"))
(security=(ssl_server_cert_dn="<certificate_info>"))

- b. To connect an ATP instance with JDBC OCI, use a database URL as shown below:

jdbc:oracle:oci8:@tnsname , where tnsname is the network name to identify a server or sid or port combination.

For Example — jdbc:oracle:oci8:@adwc_high

- **JDBC Properties: (Wallet Specific Properties)**

- oracle.net.ssl_server_dn_match = true
- oracle.net.wallet_location =(SOURCE=(METHOD=file)
(METHOD_DATA=(DIRECTORY=<wallet_directory>))

For Example —WALLET_LOCATION = (SOURCE = (METHOD = file)
(METHOD_DATA = (DIRECTORY="/network/admin/ewallet")))

This property is used to force the distinguished name (dn) of the server to match with its service name.

Creating an Oracle Physical Schema

Create an Oracle physical schema for ATP using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*.

Create a logical schema for this physical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator* and associate it in a given context.

Creating and Reverse-Engineering an Oracle Model

This section contains the following topics:

- [Create an Oracle Model](#)
- [Reverse Engineer an Oracle Model](#)

Create an Oracle Model

Create an Oracle Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Reverse Engineer an Oracle Model

An Oracle model for ATP 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 and references.

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

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on an Oracle model for ATP, use the usual procedure, as described in Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on Oracle model for ATP with a RKM, use the usual procedure, as described in Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator.

This section details only the fields specific to the Oracle technology. In the Reverse Engineer tab of the Oracle Model, select the KM — RKM Oracle.<project name>.

Designing a Mapping

You can use Oracle ATP as a source or a target of a mapping. It is also possible to create ETL-style mappings based on the Oracle technology for ATP. The recommendations in this section help in the selection of the KM for different situations concerning an Oracle Database Server.

- [Loading Data](#)
- [Extracting data](#)

Loading data

ATP can be used as a source or target of a mapping. The choice of LKM used in the mappings for loading Knowledge Module, to load data between Oracle and another type of data server determines the working and performance of a mapping.

The following KMs implement optimized methods for loading data from an Oracle database to a target database which is ATP. Our primary goal is to load data into ATP. In addition to these KMs, you can also use the Generic SQL KMs or the KMs specific to the other technologies involved.

- [Loading Data using Oracle KMs](#)
- [Loading Data using SQL* Loader KMs](#)
- [Loading Data directly into ATP](#)
- [Loading Oracle Object Storage files into ATP](#)

Loading Data using Oracle KMs

You can load data into Oracle tables using the following Oracle KMs by designing a mapping where ADWC or ATP Oracle data stores can be the target. KMs that can be used for mapping are:

- LKM SQL to Oracle (Built-In)
- IKM Oracle Insert
- IKM Oracle Update
- IKM Oracle Merge
- IKM Oracle Multi-Insert

Loading Data using SQL* Loader KMs

You can also load data into Oracle tables for ADWC or ATP using the SQL* Loader KM. LKM File to Oracle (SQLLDR) loads data into Oracle tables from files. You can design a mapping that uses the data stores for an Oracle Schema for ADWC or ATP as the target of the mapping, where the data is loaded using the SQL*Loader KM.

Connection Setup for SQL* Loader KMs

Make sure your `tnsnames.ora` and `sqlnet.ora` properties are configured to use a Wallet file.

For Example:

1. `sqlnet.ora` :

```
WALLET_LOCATION=(SOURCE = (METHOD = file)
(METHOD_DATA = (DIRECTORY="<wallet_directory>")))
SSL_SERVER_DN_MATCH=yes
```

2. `tnsnames.ora` :

- `<db_name>_DB_high=(description=(address=(protocol=tcps) (port=<port>)(host=<hostname or ipaddress>)) (connect_data=(service_name=<db_service>)) (security=(ssl_server_cert_dn="<certificate_info>"))))`
- `<db_name>_DB_medium=(description=(address=(protocol=tcps) (port=<port>)(host=<hostname or ipaddress>)) (connect_data=(service_name=<db_service>)) (security=(ssl_server_cert_dn="<certificate_info>"))))`
- `<db_name>_DB_low=(description=(address=(protocol=tcps) (port=<port>) (host=<hostname or ipaddress>)) (connect_data=(service_name=<db_service>)) (security=(ssl_server_cert_dn="<certificate_info>"))))`

For more details, refer to [Use of an External Password Store to Secure Passwords](#) section of Database Security Guide.

Loading Data directly into ATP

Note:

This section applies only to Data Integration Platform Cloud.

You can use the following knowledge modules for loading data directly into Oracle ATP. ODI uses the same optimized KMs for both Oracle ADWC and Oracle ATP cloud services.

- [LKM SQL to ADWC External Table](#)
- [LKM SQL to ADWC Copy](#)
- [LKM SQL to ADWC Copy Direct](#)
- [LKM File to ADWC External Table](#)
- [LKM File to ADWC Copy](#)
- [LKM File to ADWC Copy Direct](#)
- [LKM Oracle to ADWC Datapump](#)

For loading data directly into ATP set the Source of the mapping to be File/HDFS or SQL technology such as Oracle. Target of the mapping has to be Oracle technology, more specifically ATP database. Object storage staging area is used to temporary store files and this is defined by setting KM option `TEMP_OBJECT_STORAGE_SCHEMA` (logical schema). Some properties such as user/password are retrieved from Oracle Object Storage data server attached to `TEMP_OBJECT_STORAGE_SCHEMA` logical schema. If you need to create temporary local file in case of SQL source, you have to define its location as `TEMP_FILE_SCHEMA` KM option. If you use transform components, they can be included in the source execution unit in case of SQL as a source and/or on the target ATP execution unit. File, as a source does not support source transformations. Direct Copy KMs do not support transformations on the target.

LKM SQL to ADWC External Table

This LKM helps in loading data from SQL source to Oracle ADWC or Oracle ATP using Oracle Object Storage as intermediate staging. The result from SQL query is first loaded into Oracle Object Storage staging area and then External Table method is used to pull data from Oracle Object Storage. You must set the `TEMP_OBJECT_STORAGE_SCHEMA` and `TEMP_FILE_SCHEMA` KM options to designate temporary storage locations.

You have to use this LKM in combination with Oracle or generic SQL IKM.

The KM invokes the ODI tool `OdiSqlUnload` to unload SQL query data to a local file.

For Example –

```
OdiSqlUnload "-FILE=/tmp/odi_e3a779f8-ddd3-49d2-9575-
a9f3c675b0f6_FILTER_AP.txt" "-DRIVER=oracle.jdbc.OracleDriver" "-
URL=jdbc:oracle:thin:@//slc03sap:1521/flex" "-USER=system" "-
PASS=<@=odiRef.getInfo("SRC_ENCODED_PASS") @>" "-FILE_FORMAT=VARIABLE" "-
FIELD_SEP=," "-ROW_SEP=" "-DATE_FORMAT=MM-DD-YYYY" "-
CHARSET_ENCODING=ISO8859_1"
```

```
SELECT
  PER.PID AS PID ,
  PER.PNAME AS PNAME
FROM
  UT_TD_D_1.PERSON PER
WHERE (PER.PID = 2)
```

The KM then calls ODI tool `OdiObjectStorageUpload` to upload the local file to Oracle Object Storage.

```
OdiObjectStorageUpload "-TRG_LOGICAL_SCHEMA=Object Storage - SRC1" "-
SRC_LOGICAL_SCHEMA=FILE_GENERIC_TMP" "-FILE_NAMES_FILTER=odi_e3a779f8-
ddd3-49d2-9575-a9f3c675b0f6_FILTER_AP.txt" "-OVERWRITE=true"
```

The KM creates a temporary staging external table to pull the data from Oracle Object Storage.

```
BEGIN
  dbms_cloud.create_external_table(
    table_name => 'C$_0FILTER_EXT',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/odi_e3a779f8-
ddd3-49d2-9575-a9f3c675b0f6_FILTER_AP.txt',
    column_list => 'PID NUMBER(2), PNAME VARCHAR2(20)',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  );
END;
```

The KM optionally creates the credential object to connect to Object Storage, but can also re-use an existing one.

```
dbms_cloud.create_credential(
  credential_name => 'ODI_FLEX',
  username => 'tenant15',
  password => 'xxxxxxxx'
);
END;
```

Note:

LKM SQL to ADWC External Table has two different KM options for formatting date/time datatypes:

1. `UNLOAD_DATE_FORMAT`– Use this KM option while unloading SQL query into text file. The formatting syntax is in conform to Java Date and Time Patterns.
2. `DATE_FORMAT`– Use this KM option while loading object storage text file into ADWC or ATP. The formatting syntax is in conform to Oracle database syntax for formatting DATE datatype.

If you are loading into `TIMESTAMP` column, use Advanced option `ADD_FORMAT_PROPERTIES` to add the timestamp format property. It is very important to

synchronize UNLOAD_DATE_FORMAT and DATE_FORMAT/timestampformat to use identical format settings, even though they are using different syntax.

Here is an example of loading from source TIMESTAMP column into target TIMESTAMP column.

```
UNLOAD_DATE_FORMAT = yyyy-MM-dd
UNLOAD_TIMESTAMP_FORMAT = HH:mm:ss.SSS
ADD_FORMAT_PROPERTIES = 'timestampformat' VALUE 'YYYY-MM-DD HH24:MI:SS.FF3'
```

Also, make sure that TIMESTAMP column precision allows to store the given format.

KM Options

This KM has the following options:

- CREDENTIAL_NAME — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- CREATE_CREDENTIAL- It creates new credentials. If set to False, ODI reuses the existing credentials.
- GENERATE_FIELD_LIST — If this KM option is set to False, then the field_list clause is skipped and the default settings of ORACLE_LOADER access driver is applied.

Note:

You have to always generate the field_list clause as it required by Fixed file format.

- TEMP_OBJECT_STORAGE_SCHEMA — It specifies the name of logical schema defining the location of the temporary file that will be stored in Oracle Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- CLEANUP_TEMPORARY_OBJECTS— Set this property to True, if you want temporary objects to be automatically cleaned up.
- ADD_COMPRESSION — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options COMPRESSION_TYPEand KEEP_SOURCE_FILESdefine compression preferences.
- COMPRESSION_TYPE- It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- KEEP_SOURCE_FILES — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

Note:

gzip supports KEEP_SOURCE_FILESoption, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` - It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. Select the required trim option from the provided list of trim options.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Cleanup

- `CLEANUP_EXTERNAL_TABLE`— Set this property to True, if you want the external table to be automatically cleaned up at the end of the every execution.
- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

SQL Unload Options

- `TEMP_FILE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that is stored before uploading to Oracle Object Storage. This must be a File technology logical schema. The temporary file is stored on local file system where the ODI agent is running.
- `UNLOAD_DATE_FORMAT`— It specifies the output format used for date datatypes. This format follows Java date format standards.
- `UNLOAD_TIMESTAMP_FORMAT`— It specifies the output format used for time datatypes. This format follows Java time format standards.
- `CHARSET_ENCODING` — Use this for character set encoding.
- `FETCH_SIZE` — It denotes the number of rows (records read) requested by ODI agent on each communication with the data server.

LKM SQL to ADWC Copy

This LKM helps in loading data from SQL source to Oracle ADWC or Oracle ATP using Oracle Object Storage as intermediate staging. The result from SQL query is first loaded into Oracle Object Storage staging area. Then we use `dbms_cloud.copy_data()` to load data from Oracle Object Storage into staging table on ADWC or ATP. The user must set `TEMP_OBJECT_STORAGE_SCHEMA` and `TEMP_FILE_SCHEMA` KM options to designate temporary storage locations.

You can use this LKM in combination with Oracle or generic SQL IKM.

`OdiSqlUnload` and `OdiObjectStorageUpload` follows the same steps as described in LKM SQL to ADWC External Table. LKM SQL to ADWC Copy is then creating a temporary staging table on ADWC execution unit and pulls the data from Object Store into it using `dbms_cloud.copy_data()` function.

For Example

```
create table STAR.C$_0FILTER
(
  PID NUMBER(2),
  PNAME VARCHAR2(20)
)
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'STAR',
    table_name => 'C$_0FILTER',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/odi_d0b58fb8-
bde5-4ce5-89da-0d258c98380e_FILTER_AP.txt',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  );
END;
```


The KM optionally creates the credential object to connect to Oracle Object Storage, but can also re-use an existing one:

```
BEGIN
  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxxx'
  );
END;
```

 **Note:**

LKM SQL to ADWC Copy has two different KM options for formatting date/time datatypes:

- `UNLOAD_DATE_FORMAT`- Use this KM option while unloading SQL query into text file. The formatting syntax is conform to Java Date Patterns.
- `DATE_FORMAT` — Use this KM option while loading object storage text file into ADWC or ATP. The formatting syntax is conform to Oracle database syntax for formatting DATE datatype.

If you are loading into `TIMESTAMP` column use Advanced option `ADD_FORMAT_PROPERTIES` to add `timestampformat` property. It is very important to synchronize `UNLOAD_DATE_FORMAT` and `DATE_FORMAT/timestampformat` to use identical format settings, even though they are using different syntax.

Here is an example of loading from source `TIMESTAMP` column into target `TIMESTAMP` column.

- `UNLOAD_DATE_FORMAT = yyyy-MM-dd HH:mm:ss.SSS`
- `ADD_FORMAT_PROPERTIES = 'timestampformat' VALUE 'YYYY-MM-DD HH24:MI:SS.FF3'`

Also, make sure that `TIMESTAMP` column precision allows the given format to be stored.

KM Options

This KM has the following KM options:

- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL` — It creates new credentials. If set to False, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST` — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `TEMP_OBJECT_STORAGE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that will be stored in Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- `ADD_COMPRESSION` — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.
- `COMPRESSION_TYPE` — It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES` — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

 **Note:**

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.

- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to True, it trims the specified spaces.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` - This option allows adding custom format properties.
Use the following syntax: `<prop1>' VALUE '<value1>', '<prop2>' VALUE '<value2>' ...`
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Cleanup

- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want temporary objects to be automatically cleaned up.
- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

SQL Unload Options

- `TEMP_FILE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that is stored before uploading to Object Storage. This must be a File technology logical schema. The temporary file is stored on local file system where the ODI agent is running.
- `UNLOAD_DATE_FORMAT` — It specifies the output format used for date datatypes. This format follows Java date/time format standards.
- `UNLOAD_TIMESTAMP_FORMAT` — It specifies the output format used for time datatypes. This format follows Java time format standards.
- `CHARSET_ENCODING` — Use this for character set encoding.
- `FETCH_SIZE` — It denotes the number of rows (records read) requested by ODI agent on each communication with the data server.

LKM SQL to ADWC Copy Direct

This LKM loads data from SQL source to Oracle ADWC or Oracle ATP using Oracle Object Storage as intermediate staging. The result from SQL query is first loaded into Oracle Object Storage staging area. Then we use `dbms_cloud.copy_data ()` function to load data from Oracle Object Storage directly into ADWC or Oracle ATP target table. All target columns are loaded irrespective of whether they are mapped or not. Only transformations on source are supported. The user must set

TEMP_OBJECT_STORAGE_SCHEMA and TEMP_FILE_SCHEMA KM options to designate temporary storage locations.

You can use this LKM as a standalone KM as you do not need any IKM.

OdiSqlUnload and OdiObjectStorageUploadSteps are the same as described in LKM SQL to ADWC External Table. LKM SQL to ADWC Copy Direct pulls the data from Object Store and loads data directly into target using dbms_cloud.copy_data() function.

For Example

```
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'STAR',
    table_name => 'PERSON',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcdemo/odi_bojana/
odi_16a97fd6-8a61-4303-8f83-81988222b8cb_FILTER_AP.txt',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  );
END;
```

The KM optionally creates the credential object to connect to Object Storage, but can also re-use an existing one.

```
BEGIN

  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxxx'
  );

END;
```

Note:

LKM SQL to ADWC Copy Direct has two different KM options for formatting date/time Data types:

- UNLOAD_DATE_FORMAT is used while unloading SQL query into text file. The formatting syntax is conform to Java Date and Time Patterns.
- DATE_FORMAT is used while loading object storage text file into ADWC or ATP. The formatting syntax is conform to Oracle database syntax for formatting DATE data type.

If you are loading into TIMESTAMP column use Advanced option ADD_FORMAT_PROPERTY to add timestampformatproperty. It is very important to

synchronize UNLOAD_DATE_FORMAT and DATE_FORMAT/timestampformat to use identical format settings, even though they are using different syntax.

Here is an example of loading from source TIMESTAMP column into target TIMESTAMP column.

```
UNLOAD_DATE_FORMAT = yyyy-MM-dd HH:mm:ss.SSS  
ADD_FORMAT_PROPERTIES = 'timestampformat' VALUE 'YYYY-MM-DD HH24:MI:SS.FF3'
```

Also, make sure that TIMESTAMP column precision allows for the given format to be stored.

KM Options

This KM has the following options:

- **CREDENTIAL_NAME**— It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- **CREATE_CREDENTIAL**— It creates new credentials. If set to False, ODI reuses the existing credentials.
- **GENERATE_FIELD_LIST**— If this KM option is set to False, then the field_list clause is skipped and the default settings of ORACLE_LOADER access driver is applied.

Note:

You have to always generate the field_list clause as it is required by Fixed file format.

- **TEMP_OBJECT_STORAGE_SCHEMA**— It specifies the name of logical schema defining the location of the temporary file that will be stored in Oracle Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- **CLEANUP_TEMPORARY_OBJECTS** — Set this property to True, if you want temporary objects to be automatically cleaned up.
- **ADD_COMPRESSION**- It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options **COMPRESSION_TYPE** and **KEEP_SOURCE_FILES** define compression preferences.
- **COMPRESSION_TYPE**- It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- **KEEP_SOURCE_FILES**— Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

Note:

gzip supports **KEEP_SOURCE_FILES** option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT`- It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to True it trims the specified spaces.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Target

- `CREATE_TARG_TABLE`— It helps you to create target table. Set this option to True, if you want to create target table before loading.
- `TRUNCATE_TARG_TABLE`— It helps to truncate target table. Set this KM option to True if you want to truncate target table before loading it.

- `DELETE_TARG_TABLE`— It allows you to delete the target table. Set this KM option to True if you want to delete data from target table before loading.

Cleanup

- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

SQL Unload Options

- `TEMP_FILE_SCHEMA`— It specifies the name of logical schema defining the location of the temporary file that is stored before uploading to Object Storage. This must be a File technology logical schema. The temporary file is stored on local file system where the ODI agent is running.
- `UNLOAD_DATE_FORMAT`— It specifies the output format used for date data types. This format follows Java date/time format standards.
- `UNLOAD_TIMESTAMP_FORMAT`— It specifies the output format used for time data types. This format follows Java time format standards.
- `CHARSET_ENCODING`— Use this for character set encoding.
- `FETCH_SIZE`- It denotes the number of rows (records read) requested by ODI agent on each communication with the data server.

LKM File to ADWC External Table

This LKM helps to load data from local or HDFS file source to Oracle ADWC or Oracle ATP using Oracle Object Storage as intermediate staging. The source file is first loaded into Oracle Object Storage staging area. Then we use External Table method to pull data from Object Storage. You must set `TEMP_OBJECT_STORAGE_SCHEMA` KM option to designate temporary storage location.

You have to use this LKM in combination with Oracle or generic SQL IKM.

The KM invokes ODI tool `OdiObjectStorageUpload` to upload the local file to Object Storage.

For Example

```
OdiObjectStorageUpload "-TRG_LOGICAL_SCHEMA=Object Storage - SRC1" "-
SRC_LOGICAL_SCHEMA=FILE_GENERIC_TMP" "-
FILE_NAMES_FILTER=person_no_header.csv" "-OVERWRITE=true"
```

The KM then creates a temporary staging external table to pull the data from Oracle Object Storage.

```
BEGIN
  dbms_cloud.create_external_table(
    table_name => 'C$_OPER_EXT',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcdemo/odi_bojana/person_no_header.csv',
    column_list => 'PID NUMBER(2,0), PNAME VARCHAR2(20)',
    format => json_object(
      'type' VALUE 'CSV',
```

```

        'skipheaders' VALUE '0',
        'dateformat' VALUE 'AUTO')
    );
END;
```

The KM optionally creates the credential object to connect to Oracle Object Storage, but can also re-use an existing one.

```

BEGIN
    dbms_cloud.create_credential(
        credential_name => 'ODI_FLEX',
        username => 'tenant15',
        password => 'xxxxxxxxx'
    );
END;
```

KM Options

This KM has the following options:

- **CREDENTIAL_NAME** — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- **CREATE_CREDENTIAL** — It creates new credentials. If set to False, ODI reuses the existing credentials.
- **GENERATE_FIELD_LIST** — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

Note:

You have to always generate the `field_list` clause as it is required by Fixed file format.

- **TEMP_OBJECT_STORAGE_SCHEMA** — It specifies the name of logical schema defining the location of the temporary file that will be stored in Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- **ADD_COMPRESSION** — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.
- **COMPRESSION_TYPE** — It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- **KEEP_SOURCE_FILES** — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

Note:

`gzip` supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and AUTO implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option AUTO searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to True, it trims the specified spaces.
- `IGNORE_BLANK_LINES` — It set to True the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: `'<prop1>' VALUE '<value1>', '<prop2>' VALUE '<value2>' ...`
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Cleanup

- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want temporary objects to be automatically cleaned up.

- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

LKM File to ADWC Copy

This LKM helps to load data from local or HDFS file source to Oracle ADWC or Oracle ATP using Oracle Object Storage as intermediate staging. The source file is first loaded into Oracle Object Storage staging area. Then we use `dbms_cloud.copy_data()` to load data from Oracle Object Storage into staging table on ADWC or ATP. You must set `TEMP_OBJECT_STORAGE_SCHEMA` KM option to designate temporary storage location.

You can use this LKM in combination with Oracle or generic SQL IKM.

OdiObjectStorageUpload happens similar to LKM File to ADWC External Table.

LKM File to ADWC Copy then creates a temporary staging table on ADWC execution unit and pulls the data from Object Store into it using `dbms_cloud.copy_data()` function.

For Example

```
create table STAR.C$_OPER
(  PID NUMBER(2,0),
   PNAME VARCHAR2(20)
)
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'STAR',
    table_name => 'C$_OPER',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/person_no_header.csv',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  );
END;
```

The KM optionally creates the credential object to connect to Object Storage, but can also re-use an existing one.

```
BEGIN

  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxxx'
  );
END;
```

KM Options

This KM has the following options:

- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL` — It creates new credentials. If set to False, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST` — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `TEMP_OBJECT_STORAGE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that will be stored in Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- `ADD_COMPRESSION` — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.
- `COMPRESSION_TYPE` — It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES` — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

 **Note:**

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types or auto. Empty value implies no compression and AUTO implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option AUTO searches for the following formats:

J
MM-DD-YYYYBC

MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to True, it trims the specified spaces.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE MISSING COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Cleanup

- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want temporary objects to be automatically cleaned up.
- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

LKM File to ADWC Copy Direct

This KM is helpful to load data from local or HDFS file source to Oracle ADWC or Oracle ATP using Oracle Object Storage as intermediate staging. The source file is first loaded into Oracle Object Storage staging area. Then we use `dbms_cloud.copy_data()` function to load data from Object Storage directly into ADWC or ATP target table. Source attributes must match target column names. Either use data entities with matching attributes or set option `GENERATE_FIELD_LIST` to false. All target columns are loaded, irrespective of their mapping.

You must set `TEMP_OBJECT_STORAGE_SCHEMA` KM option to designate temporary storage location. LKM File to ADWC Copy Direct does not support any transformations.

You can use this LKM as a standalone KM as you do not need an IKM for its implementation. `OdiObjectStorageUpload` happens similar to LKM File to ADWC External Table. LKM SQL to ADWC Copy Direct pulls the data from Oracle Object Storage and loads data directly into target using `dbms_cloud.copy_data()` function.

For Example

```
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'STAR',
    table_name => 'PERSON',
    credential_name => 'ODI',
    file_uri_list => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/person_no_header.csv',
    format => json_object(
      'type' VALUE 'CSV',
      'skipheaders' VALUE '0',
      'dateformat' VALUE 'AUTO')
  );
END;
```

The KM optionally creates the credential object to connect to Oracle Object Storage, but can also re-use an existing one.

```
BEGIN

  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxxx'
  );

END;
```

KM Options

This KM has the following options:

- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL` — It creates new credentials. If set to False, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST` — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `TEMP_OBJECT_STORAGE_SCHEMA` — It specifies the name of logical schema defining the location of the temporary file that will be stored in Object Storage staging area. This must be an Oracle Object Storage technology logical schema.
- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want temporary objects to be automatically cleaned up.
- `ADD_COMPRESSION` — It compresses data before loading. Set this property to True, if you want to compress source data before loading into Oracle Object Storage. Additional KM options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` define compression preferences.
- `COMPRESSION_TYPE` — It allows you to configure the required compression type. Select the type of compression you want to apply on source data before loading into Oracle Object Storage.
- `KEEP_SOURCE_FILES` — Use this KM option to retain the original source files after compression. Set this property to True (by default), if you wish to compress the target files and retain the original files.

 **Note:**

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

Formatting

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source data store File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. There are 3 possible compression types, or auto. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option `AUTO` searches for the following formats

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.

- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to True, it trims the specified spaces.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.

Advanced

- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

Target

- `CREATE_TARG_TABLE` — It helps you to create target table. Set this option to True, if you want to create target table before loading.
- `TRUNCATE_TARG_TABLE` — It helps to truncate target table. Set this KM option to True if you want to truncate target table before loading it.
- `DELETE_TARG_TABLE` — It allows you to delete the target table. Set this KM option to True if you want to delete data from target table before loading.

Cleanup

- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

LKM Oracle to ADWC Datapump

This KM is helpful to load data from Oracle On-premises products to Oracle ADWC or Oracle ATP. You can use it in combination with Oracle or generic SQL IKM.

Working

First source table is exported into local Datapump dump file. The export file is created in a local directory specified by `TEMP_FILE_SCHEMA` logical schema. A unique Export file name is generated as `odi_<ODI_session_id>_<mapping_node>.dmp`. Export log name is `odi_<ODI_session_id>_<mapping_node>_exp.log`. The export file is then uploaded to Oracle Object storage bucket specified by `TEMP_OBJECT_STORAGE_SCHEMA` logical schema. It is then imported to ADWC or ATP database into temporary staging table `C$_alias`. Import log name is

odi_<ODI_session_id>_<mapping_node>_imp.log. Finally, the mapping IKM integrates the staging table into target. Transformations on target execution unit are supported.

 **Note:**

- ODI agent has to run on the same host as the source database to be able to access the dump/log files.
- Currently the KM does not support multi file loading from Oracle Object Storage. Hence you cannot set FILESIZE and PARALLEL options. dbms_datapump package is used for export and import.

Example of Export code

```
declare
  h1 number;
  j_status varchar2(200);
begin
  dbms_output.enable();
  h1 := dbms_datapump.open('EXPORT','SCHEMA',NULL,'ODI_EXPORT','LATEST');
  dbms_datapump.add_file(
    handle => h1,
    filename => 'odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP.dmp',
    directory => 'ODI_DIR',
    filetype => dbms_datapump.KU$_FILE_TYPE_DUMP_FILE,
    reusefile => 1);
  dbms_datapump.add_file(
    handle => h1,
    filename => 'odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP_exp.log',
    directory => 'ODI_DIR',
    filetype => dbms_datapump.KU$_FILE_TYPE_LOG_FILE);
  dbms_datapump.metadata_filter( h1, 'SCHEMA_LIST', q'|'UT_TD_D_1'|' );
  dbms_datapump.metadata_filter( h1, 'NAME_LIST', q'|'PERSON_SRC'|',
'TABLE' );
  dbms_datapump.metadata_filter( h1, 'INCLUDE_PATH_LIST', q'|'TABLE'|' );
  dbms_datapump.set_parameter(h1, 'COMPRESSION', 'METADATA_ONLY');
  dbms_datapump.set_parameter(h1, 'COMPRESSION_ALGORITHM', 'BASIC');
  dbms_datapump.start_job(h1);
  dbms_datapump.wait_for_job(h1, j_status);
exception
  when others then
    ....
end;
```

Example of Import code

```
declare
  h1 number;
  j_status varchar2(200);
begin
  h1 := dbms_datapump.open('IMPORT','FULL',NULL,'ODI_IMPORT','LATEST');
```



```

dbms_datapump.add_file(
    handle => h1,
    filename => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/
odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP.dmp',
    directory => 'ODI',
    filetype => dbms_datapump.KU$FILE_TYPE_URIDUMP_FILE);
dbms_datapump.add_file(
    handle => h1,
    filename => 'odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP_imp.log',
    directory => 'DATA_PUMP_DIR',
    filetype => dbms_datapump.KU$FILE_TYPE_LOG_FILE);
dbms_datapump.metadata_remap( h1, 'REMAP_SCHEMA', 'UT_TD_D_1', 'STAR');
dbms_datapump.metadata_remap( h1, 'REMAP_TABLE', 'PERSON_SRC',
'CS_OPER');
dbms_datapump.set_parameter(h1, 'TABLE_EXISTS_ACTION', 'SKIP');
dbms_datapump.set_parameter(h1, 'PARTITION_OPTIONS', 'MERGE');
dbms_datapump.metadata_transform( h1, 'SEGMENT_ATTRIBUTES', 0);
dbms_datapump.start_job(h1);
dbms_datapump.wait_for_job(h1, j_status);
dbms_cloud.put_object(
    credential_name => 'ODI',
    object_uri => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsdemo/odi_bojana/
odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP_imp.log',
    directory_name => 'DATA_PUMP_DIR',
    file_name =>
'odi_de73e691-2ef5-4fff-8d98-697cf8e123cf_PER_AP_imp.log');
exception
    when others then
        ....
end;

```

KM Options

This KM has the following options:

Source Export

- **TEMP_FILE_SCHEMA**— It specifies the name of logical schema defining the location of the temporary export file that is stored before uploading to Oracle Object Storage. This must be a File technology logical schema. Oracle directory is created based on this location. The temporary file is stored on a file system accessible by the source database.
- **ORACLE_DIRECTORY_NAME** — It specifies the name of Oracle directory used by Datapump to store export file and logs on Source. The default value is ODI_DIR.

Note:

Use only uppercase for directory names.

- **COMPRESSION** — It specifies which data to compress before writing to the dump file set. You can specify any of the following compression options:

- ALL — enables compression for the entire export operation. You must enable **Oracle Advanced Compression** option for making this type of compression.
- DATA_ONLY — When you select this option, entire data is written to the dump file in a compressed format. You must enable Oracle Advanced Compression option for making this type of compression.
- METADATA_ONLY — When you select this option, entire metadata is written to the dump file in compressed format. This is the default option for COMPRESSSION.
- NONE — It disables compression for the entire export operation.
- COMPRESSION_ALGORITHM— It specifies the compression algorithm to be used when compressing dump file data. You can specify one of the following compression algorithm options:
 - BASIC — Offers a good combination of compression ratios and speed; the algorithm used is the same as in previous versions of Oracle Datapump.
 - LOW — Least impact on export throughput and suited for environments where CPU resources are the limiting factor.
 - MEDIUM — Recommended for most environments. This option is similar to the BASIC option that provides a good combination of compression ratios and speed, but it uses a different algorithm when compared to BASIC algorithm.
 - HIGH — Best suited for situations in which dump files are copied over slower networks where the limiting factor is the network speed.
- CREATE_VIEW_ON_SOURCE — It allows you to create view on source. Set this property to False only if you have a single source with no transformations or a simple filter.
- CREATE_VIEW_ON_SOURCE_TEMP_SCHEMA — It allows you to create view on source work schema. Set this property to True if you want to create view on source Work Schema.

 **Note:**

- Work Schema user should have select privileges to all the source tables participating in the view select statement for the view to be valid. If set to False the view is created in the schema of the currently connected user.
 - You cannot remap a SYSTEM schema, which means if set to False you must not be connected as SYSTEM. Because of restrictions in both the cases, we provide this option.
- USE_GROUP_PARTITION_TABLE_DATA — Use parameter DATA_OPTIONS=GROUP_PARTITION_TABLE_DATA on export. GROUP_PARTITION_TABLE_DATA value for DATA_OPTIONS parameter is only available for Oracle 12.2 versions or later.

**Note:**

Set this option to False, for any previous Oracle versions.

Target Import

- `CREDENTIAL_NAME` — It provides credential name to connect to Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL` — It helps to create new credential. If set to False ODI will reuse existing credentials.
- `TEMP_OBJECT_STORAGE_SCHEMA` — It denotes the logical schema for temporary Oracle Object Storage location. Specify the name of logical schema defining the location of the temporary export file that will be stored in Oracle Object Storage staging area. This must be an Object Storage technology logical schema.

Cleanup

- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want to clean up the temporary objects automatically.
- `CLEANUP_CREDENTIAL` — Set this property to True, if you want to clean up the credential object automatically at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to true.

Loading Oracle Object Storage files into ATP

**Note:**

This section applies only to Data Integration Platform Cloud.

You can use the following knowledge modules for loading Oracle Object Storage files into Oracle ATP:

- [LKM Object Storage to ADWC Copy](#)
- [LKM Object Storage to ADWC Copy Direct](#)
- [LKM Object Storage to ADWC External Table](#)

When you load data from Oracle Object Storage to ATP, target of the mapping has to be Oracle technology as there is no specific technology for ATP. Object Storage Bucket represents Object Storage physical schema. Some properties such as Heading and Delimiter are retrieved from source Data Store. Other properties such as user name and password are retrieved from Object Storage Data Server. If you use transform components, they need to be moved to the target (Oracle) execution unit. No transformations on source are supported.

LKM Object Storage to ADWC Copy

This KM helps to load data from Oracle Cloud Object Storage to Oracle ADWC or Oracle ATP. It is using `dbms_cloud.copy_data()` function to load data into a staging table. LKM Object Storage to ADWC Copy is assigned to an AP node. It is used in moving data from Oracle Object Storage file to an ADWC table. You can use it in combination with Oracle or generic SQL IKM.

For Example

LKM Object Storage to ADWC External Table is creating a temporary staging external table to pull the data from Object Store.

```
BEGIN
  dbms_cloud.create_external_table(
    table_name => 'C$_OPER_EXT',
    credential_name => 'ODI_FLEX',
    file_uri_list => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsprod/bucket_tenant15/person.csv',
    column_list => 'PID NUMBER(2,0),
PNAME VARCHAR2(20)',
    field_list => 'PID, PNAME',
    format => json_object('type' VALUE 'CSV', 'skipheaders' VALUE '1')
  );
END;
```

It also optionally creates the credential object, but can also re-use an existing one.

```
BEGIN

  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxxx'
  );
END;
```

KM Options

This KM has the following options:

- **CREDENTIAL_NAME** — It provides credential name to connect to Object Storage. The default value is ODI.
- **CREATE_CREDENTIAL** — It helps to create new credential. If set to False ODI will reuse existing credentials.
- **GENERATE_FIELD_LIST** — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it is required by Fixed file format.

- **DELIMITED_FILE_FORMAT** — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.

- **COMPRESSION** — It specifies the compression method of the source file. It can have values `nil` or `auto`. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- **DATE_FORMAT** — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- **REJECT_LIMIT** — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- **CONVERSION_ERRORS** — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- **TRIM_SPACES** — It helps to trim the leading and trailing spaces of the fields. If set to `True` it trims the specified spaces.
- **IGNORE_BLANK_LINES** — It set to `True` the blank lines are ignored without throwing any error.
- **IGNORE_MISSING_COLUMNS** — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- **TRUNCATE_COLUMNS** — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.
- **ADD_FORMAT_PROPERTIES** — This option allows adding custom format properties.
Use the following syntax: `<'prop1>' VALUE '<value1>', '<prop2>' VALUE '<value2>' ...`
- **OVERWRITE_FIELD_LIST** — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.
For more details, refer to [DBMS_CLOUD](#) package documentation for more information.
- **CLEANUP_TEMPORARY_OBJECTS** — Set this property to `True`, if you want temporary objects to be automatically cleaned up.
- **CLEANUP_CREDENTIAL** — Set this property to `True`, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to `True`.

LKM Object Storage to ADWC Copy Direct

This KM is helpful to load data from Oracle Object Storage to Oracle ADWC or Oracle ATP. It is using `dbms_cloud.copy_data ()` function. This LKM can be used as standalone as no IKM is needed. It loads data directly into target table. Source attributes must match target column names. Either use data entities with matching

attributes or set option `GENERATE_FIELD_LIST = false`. All target columns are loaded irrespective of whether they are mapped or not.



Note:

LKM Object Storage to ADWC Copy Direct does not support any transformations.

The LKM Object Storage to ADWC Copy Direct is assigned to an AP node. This is used for moving files/objects from Oracle Object Storage to an ADWC or ATP table.

For Example

LKM Object Store to ADWC Copy Direct pulls the data from Oracle Object Storage and loads data directly into target using `dbms_cloud.copy_data()` function.

```
BEGIN
  dbms_cloud.copy_data(
    schema_name => 'ODI',
    table_name => 'PERSON',
    credential_name => 'ODI_FLEX',
    file_uri_list => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsprod/bucket_tenant15/person.csv',
    field_list => 'PID, PNAME',
    format => json_object('type' VALUE 'CSV', 'skipheaders' VALUE '1')
  );
END;
```

It also optionally creates the credential object, but can also re-use an existing one.

```
BEGIN

  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxxx'
  );
END;
```

KM Options

This KM has the following options:

- `CREATE_TARG_TABLE` — It helps you to create target table. Set this option to True, if you want to create target table before loading.
- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.
- `CREATE_CREDENTIAL` — It creates new credentials. If set to False, ODI reuses the existing credentials.
- `TRUNCATE_TARG_TABLE` — It helps to truncate target table. Set this KM option to True if you want to truncate target table before loading it.

- `DELETE_TARG_TABLE` — It allows you to delete the target table. Set this KM option to True if you want to delete data from target table before loading.
- `GENERATE_FIELD_LIST` — If this KM option is set to False, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.



Note:

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be CSV (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to Delimited.
- `COMPRESSION` — It specifies the compression method of the source file. It can have values nil or auto. Empty value implies no compression and AUTO implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option AUTO searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to True it trims the specified spaces.
- `IGNORE_BLANK_LINES` — If set to True, the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.
- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input data. The details that you enter here is used as `field_list` parameter of `dbms_cloud.create_external_table` function call.

For more details, refer to [DBMS_CLOUD](#) package documentation for more information.

- `CLEANUP_TEMPORARY_OBJECTS` — Set this property to True, if you want temporary objects to be automatically cleaned up.
- `CLEANUP_CREDENTIAL` — Set this property to True, if you want the credential object to be automatically cleaned up at the end of the every execution. Cleanup will happen only if `CREATE_CREDENTIAL` option is also set to True.

LKM Object Storage to ADWC External Table

This KM is helpful to load data from Oracle Object Storage to Oracle ADWC or Oracle ATP using External Table method. You can use this LKM in combination with Oracle or generic SQL IKM.

The LKM is assigned to an AP node, so that it can move Oracle Object Storage file to Oracle ADWC or ATP table.

For Example —

LKM Object Storage to ADWC External Table is creating a temporary staging external table to pull the data from Oracle Object Storage.

```
BEGIN
  dbms_cloud.create_external_table(
    table_name => 'C$_OPER_EXT',
    credential_name => 'ODI_FLEX',
    file_uri_list => 'https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/dwcsprod/bucket_tenant15/person.csv',
    column_list => 'PID NUMBER(2,0),
PNAME VARCHAR2(20)',
    field_list => 'PID, PNAME',
    format => json_object('type' VALUE 'CSV', 'skipheaders' VALUE '1')
  );
END;
```

It also optionally creates the credential object, but can also re-use an existing one.

```
BEGIN

  dbms_cloud.create_credential(
    credential_name => 'ODI_FLEX',
    username => 'tenant15',
    password => 'xxxxxxxx'
  );

END;
```

KM Options

This KM has the following options:

- `CREDENTIAL_NAME` — It provides Credential name to connect to Oracle Object Storage. The default value is ODI.

- `CREATE_CREDENTIAL` — It creates new credentials. If set to `False`, ODI reuses the existing credentials.
- `GENERATE_FIELD_LIST` — If this KM option is set to `False`, then the `field_list` clause is skipped and the default settings of `ORACLE_LOADER` access driver is applied.

 **Note:**

You have to always generate the `field_list` clause as it is required by Fixed file format.

- `DELIMITED_FILE_FORMAT` — It specifies delimited File Format and it can be `CSV` (default) or common delimited format known to `ORACLE_LOADER` access driver. You can use this KM option only if the source datastore File Format property is set to `Delimited`.
- `COMPRESSION` — It specifies the compression method of the source file. It can have values `nil` or `auto`. Empty value implies no compression and `AUTO` implies compression type is auto-detected.
- `DATE_FORMAT` — It helps to set specific date format. The default format option `AUTO` searches for the following formats:

```
J
MM-DD-YYYYBC
MM-DD-YYYY
YYYYMMDD HHMISS
YYMMDD HHMISS
YYYY.DDD
YYYY-MM-DD
```

- `REJECT_LIMIT` — The query helps to display an error message after the specified number of rows are rejected. Default value is zero.
- `CONVERSION_ERRORS` — It specifies the processing conversion errors. If any row throws an error because of a conversion error, the related columns are stored as null or the row is rejected.
- `TRIM_SPACES` — It helps to trim the leading and trailing spaces of the fields. If set to `True` it trims the specified spaces.
- `IGNORE_BLANK_LINES` — It set to `True` the blank lines are ignored without throwing any error.
- `IGNORE_MISSING_COLUMNS` — If there are more columns in the `field_list` than the source files, the extra columns will be stored as null.
- `TRUNCATE_COLUMNS` — If the data in a file is too long for a field, then this option will truncate the value of the field rather than rejecting the row.
- `ADD_FORMAT_PROPERTIES` — This option allows adding custom format properties.
Use the following syntax: '`<prop1>`' VALUE '`<value1>`', '`<prop2>`' VALUE '`<value2>`' ...
- `OVERWRITE_FIELD_LIST` — This option gives the possibility to redefine the source file field definitions, where ODI does not have enough information about your input

5

Files

It is important to understand how to work with Files in Oracle Data Integrator. This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a File Model](#)
- [Designing a Mapping](#)

Introduction

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

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 [XML Files](#).

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 SQL KMs listed in [Table 5-1](#) are generic and can be used with any database technology. Technology-specific KMs, using features such as loaders or external tables, are listed in the corresponding technology chapter.

Table 5-1 SQL KMs

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.
IKM File to File (Java)	Integrates data in a target file from a source file using a Java processing. Can take several source files and generates a log and a bad file. See IKM File to File (Java) for more information.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

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 mapping 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.

Requirements for IKM File to File (Java)

The IKM File to File (Java) generates, compiles, and runs a Java program to process the source files. In order to use this KM, a JDK is required.

Supported Datatypes

Datatype Mappings to and from Oracle will be defined for these data types. Supported datatypes for File technology are:

- **Ascii signed zone decimal** — It is a numeric data type commonly used in COBOL files on IBM mainframes. It is a USAGE DISPLAY item where every digit is represented using one byte character, the corresponding ASCII or EBCDIC character is used for each digit. Signed zone decimal can take both positive and negative values. The ZONED data type is valid only in fixed-width text data files. The maximum length of a zoned decimal constant is 15 decimal digits (15 bytes).
- **Ascii unsigned zoned decimal** — Unsigned decimal is very similar to Ascii signed zone decimal and the difference is that it represents only positive numbers. The ASCII zoned decimal formats produce the same printable representation of numbers. There are two nibbles per byte, each indicated by a hexadecimal character. For example, the value 15 is stored in two bytes. The first byte contains the hexadecimal value F1, and the second byte contains the hexadecimal value C5.

 **Note:**

For the ASCII environment the unsigned and signed, positive values are arithmetically and physically equal.

- **Binary signed big endian** — Big endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). For example, in a big endian computer, the two bytes required for the hexadecimal number 4F52 is stored as 4F52 in storage (if 4F is stored at storage address 1000, for example, 52 will be at address 1001). Signed binary data type means that it can hold both positive or negative values i.e., an 8 bit signed binary can hold values from 0 to 127 both positive and negative.
- **Binary signed little endian** — Little endian is an order in which the "little end" (least significant value in the sequence) is stored first. For example, in a little endian computer, the two bytes required for the hexadecimal number 4F52 is stored as 524F (52 at address 1000, 4F at 1001). Signed binary data type means that it can hold both positive or negative values i.e., an 8 bit signed binary can hold values from 0 to 127 both positive and negative.
- **Binary unsigned big endian** — Big endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). For example, in a big endian computer, the two bytes required for the hexadecimal number 4F52 is stored as 4F52 in storage (if 4F is stored at storage address 1000, for example, 52 will be at address 1001). Unsigned binary data type denotes that the binary item can hold only positive values (i.e., an 8 bit unsigned binary can hold values from 0 to 255).
- **Binary unsigned little endian** — Little endian is an order in which the "little end" (least significant value in the sequence) is stored first. For example, in a little endian computer, the two bytes required for the hexadecimal number 4F52 is

stored as 524F (52 at address 1000, 4F at 1001). Unsigned binary data type denotes that the binary item can hold only positive values (i.e., an 8 bit unsigned binary can hold values from 0 to 255).

- **Date** —Based on the international ISO-8601:1988 standard date notation: YYYYMMDD where YYYY represents the year in the usual Gregorian calendar, MM is the month between 01 (January) and 12 (December), and DD is the day of the month with a value between 01 and 31. Dates may be represented in any of the following list of base data types. For some data types, a minimum of 4 bytes is required. The list includes: Numeric, String and EBCDIC.
- **Ebcdic** — A string of characters encoded using the EBCDIC (Extended Binary Coded Decimal Interchange Code) encoding used on larger IBM computers. During a reformat from EBCDIC to ASCII, if a character being converted is not in the EBCDIC character set, the conversion results in a space hexadecimal 20.
- **Ebcdic signed zoned decimal** — A signed zoned decimal is composed of regular EBCDIC numeric characters, one character per byte, for all the digits of the field except the one that holds the sign, either the most-significant (sign leading) or the least-significant (sign trailing) digit, usually the least-significant digit. The digit that holds the sign combines, or "over punches" the sign of the number onto that digit. This saves one byte that the sign would otherwise occupy.
- **Ebcdic unsigned zoned decimal** — Unsigned zoned decimal is very similar to Ebcdic signed zoned decimal and the difference is that it represents only positive numbers. For unsigned (or implied positive) fields each digit is represented by a zoned bit value in the left-most four bits (nibble or half-word) and the binary value for the digit in the right-most four bits (nibble or half-word).

 **Note:**

For the EBCDIC environment, the unsigned and signed, positive values are arithmetically equal but physically different because of the low-order byte or units position (or the sign position).

- **Fixed Ebcdic** — It is a text representation of a floating-point number, space-padded in fixed-length data, encoded as EBCDIC to comply with the character set of the text data file. The PACKED data type is valid only in fixed-width text data files. It is a packed decimal type.
- **Fixed String** — A fixed-length string that is always right-padded with spaces to the specified length when stored. M represents the column length in characters. The range of M is 0 to 255. If M is omitted, the length is 1.CHAR(0) columns can contain 2 values: an empty string or NULL. Such columns cannot be part of an index. The CONNECT storage engine does not support CHAR(0).

 **Note:**

Trailing spaces are removed when CHAR values are retrieved unless the PAD_CHAR_TO_FULL_LENGTH SQL mode is enabled.

- **Numeric** — The numeric data types store positive and negative fixed and floating-point numbers, zero, infinity, and values that are the undefined result of an operation (that is, is "not a number" or NAN). It comprises of Number data type and Floating Point Numbers. The NUMBER data type stores fixed and floating-

point numbers. Numbers of virtually any magnitude can be stored and are guaranteed portable among different systems operating Oracle Database, up to 38 digits of precision.

- **Signed packed decimal** — Packed decimal constants can be used in any type of expression. A packed decimal constant is coded as P'constantValue', where constantValue is a string of 1 to 15 decimal digits, optionally preceded by a plus (+) or minus (-) sign. If no sign is coded, the value is positive. To simplify coding of expressions, packed decimal constants can be coded without the leading P and enclosing apostrophes. The maximum length of a packed decimal constant is 15 decimal digits (8 bytes).
- **String** — A string is a data type used in programming, such as an integer and floating point unit, but is used to represent text rather than numbers. It is comprised of a set of characters that can also contain spaces and numbers. It is often implemented as an array of bytes (or words) that stores a sequence of elements, typically characters, using some character encoding. A string may also denote more general arrays or other sequence (or list) data types and structures. String is a class, a reference data type. Although the compiler has special support for strings, such as converting string literals into string instances, and performing string concatenation, string is not a primitive type, but a Class. By convention, class names begin in uppercase.
- **Unsigned packed decimal** — An unsigned packed decimal number is like a packed decimal number except that the right-most, four-bit nibble contains a packed decimal digit rather than a four-bit sign. Unsigned packed numbers are often encountered when handling certain date and time values returned by the system. Unsigned packed decimal constants are most useful in simple, single term expressions where you have a requirement to output an unsigned packed decimal value. There is no particular advantage to specifying unsigned packed decimal constants in arithmetic or relational expressions since they are always converted to packed decimal in order to complete the operation.

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.

Setting up the Topology

Setting up the topology consists in:

1. [Creating a File Data Server](#)
2. [Creating a File Physical Schema](#)

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.

 **Note:**

Starting with JDK 8, the JDBC-ODBC Bridge is no longer included with the JDK.

The JDBC-ODBC Bridge has always been considered transitional and a non-supported product that was only provided with select JDK bundles and not included with the JRE. Instead, use a JDBC driver provided by the vendor of the database or a commercial JDBC Driver instead of the JDBC-ODBC Bridge.

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 *Administering 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 5-2](#).

Table 5-2 JDBC File Driver Properties

Property	Value	Description
DATA_CONTAINS_LINE_SEPARATOR	TRUE FALSE	If set to true, when reading data, if a record contains a character (or sequence of characters) that is set as a line separator, it is not considered as a line break, but the data is read on till the read 'row size' number of characters.
ENCODING	<encoding_code>	File encoding. The list of supported encoding is available at https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html . The default encoding value is ISO8859_1.

Table 5-2 (Cont.) JDBC File Driver Properties

Property	Value	Description
ERR_FILE_PATH	empty	File location path. This path is taken by the File driver and any errors encountered by driver in parsing the data is put into <property value> + .error. The rows that cause problem are put into <property value> + .bad. So this actually causes creation of two files, in case of any problems.
MULTIBYTES_MODE	0, 1, or 2	0 is the default and indicates no special handling for multibyte. The driver reads file byte by byte 1 indicates that the file contains multibyte strings. The driver reads multibytes file character by character. 2 indicates that the file contains mixture of multibyte characters and binary data. The driver read multibytes file byte by byte for BINARY columns and character by character for other columns.
NO_PAD_DEL_NUMERIC	TRUE FALSE	Restricts left-padding of numbers (integer, float) with spaces to match the physical length of the column. Default value is FALSE.
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.
NO_RTRIM_DEL_STRINGS	TRUE FALSE	Despite its name, this same property acts on both DELimited, and FIXED format text files. When the value is set to FALSE, the trailing spaces at the end of a string value are removed. To avoid right-trimming, the property must be set to TRUE.

 **Note:**

The TRUNC_FIXED_STRINGS and TRUNC_DEL_STRINGS properties affect the treatment of data that is fed into the File driver via an INSERT statement, not the data that File driver reads from the backing file.

JDBC URL example:

```
jdbc:snps:dbfile?ENCODING=ISO8859_1&TRUNC_FIXED_STRINGS=FALSE
```

Creating a File Physical Schema

Create a File physical schema using the standard procedure, as described in *Creating a Physical Schema* in *Administering 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.

Note:

- 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* in *Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using the File database follows the standard procedure. See *Creating an Integration Project* of the *Developing Integration Projects with 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
- IKM File to File (Java)

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

Creating and Reverse-Engineering a File Model

This section contains the following topics:

- [Create a File Model](#)
- [Reverse-engineer a File Model](#)

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 Developing Integration Projects with Oracle Data Integrator*.

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 obtain, from a Microsoft Excel spreadsheet, column definitions of each file datastore within a model and automatically create the file datastores in batch without manual input.

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.

Delimited Files Reverse-Engineering

To perform a delimited file reverse-engineering:

1. In the Models accordion, right click your File Model and select **New Data Store**. The Data Store Editor opens.
2. In the Definition tab, enter the following fields:
 - **Name:** Name of this data store
 - **Resource Name:** Sub-directory (if needed) and name of the file. You can browse for the file using the browse icon next to the field.
3. Go to the Files tab to describe the type of file. Set the fields as follows:
 - **File Format:** Delimited
 - **Heading (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 Data store Editor, go to the Attributes tab.
6. In the editor toolbar, click **Reverse Engineer**.
7. Verify the data type and length for the reverse engineered attributes. Oracle Data Integrator infers the field's data types and lengths from the file contents as following:
 - a. If there are no records in the file, all data store attributes will get the default data type as String after reverse.
 - b. If records are present in the file, then all data store attributes will get the data type based on the first non-header row of the file.
 - c. If header is present in the file and File data store definition has Heading (Number of Lines) defined as not 0, then data store will be reversed with attribute names defined as in file heading.

Attributes are created with pre-generated names (C1, C2 and so on) if file has no header and it is first non-header record.
8. From the File main menu, select **Save**.

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 browse icon next to the field.

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

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 browse icon next to the field.
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 Attributes tab.
6. Create or open a File datastore that has a fixed format.
7. In the Datastore Editor, go to the Attributes 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 attributes described in the Copybook are reverse-engineered and appear in the attributes 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 attribute will appear with no data type.

Customized Reverse-Engineering

In this reverse-engineering method, Oracle Data Integrator reads from a Microsoft Excel spreadsheet containing column definitions of each file datastore within a model and creates the file datastores in batch.

A sample file called `file_repository.xls` is supplied by ODI, typically under `/demo/excel` sub-directory. Follow the specific format in the sample file to input your datastore information.

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

It is recommended that this file shall be closed before the reverse engineering is started.

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.
Note that ODI running on 64-bit JRE will work with 64-bit ODBC only.
2. Add a System DSN (Data Source Name).
3. Select the Microsoft Excel Driver (*.xls, and *.xlsx etc.) as the data source driver.
4. Name the data source `ODI_EXCEL_FILE_REPO` and select the file `/demo/excel/file_repository.xls` as the default workbook. Be sure to select driver version accordingly. Example, "Excel 12.0" for ".xlsx" files.

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:** Select the appropriate JDBC driver for Excel.
 - **JDBC URL:** Enter the URL as required by the selected JDBC driver.
 - **Array Fetch Size:** 0
2. Use default values for the rest of the parameters. From the File main menu, select **Save**.
3. Click **Test Connection** to see if the data server connects to the actual Excel file.
4. 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 logical schema will be created automatically. Note that 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.

 **Note:**

If the EXCEL_FILE_REPOSITORY logical schema does not get created before the time of import, the customization status of the imported RKM will be "Modified by User". Upon the creation of EXCEL_FILE_REPOSITORY, it will be visible as source command schema under the corresponding RKM tasks.

2. Open an existing File model (or create a new one). Define the parameters as you normally will for a File model. Note that the Technology is File, not Microsoft Excel.
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:**

The mandatory Microsoft Excel schema, `EXCEL_FILE_REPOSITORY`, is automatically used by RKM File (FROM EXCEL). It is independent from an actual File model using RKM File (FROM EXCEL).

Designing a Mapping

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

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

Loading Data From Files

Files can be used as a source of a mapping. The LKM choice in the Loading Knowledge Module tab to load a File to the staging area is essential for the mapping 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 Attribute Editor - defines the action to perform and continues to verify the remaining rows. 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.

To handle multi record files as a source of a mapping:

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 5-1](#) 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:
 - a. Create a datastore in the File Model for each type of record.
For [Example 5-1](#) create two datastores.
 - b. 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 5-1](#) you can use `ORDERS` and `ORDER_LINES` as the name of your datastores. Enter `orders.txt` in the Resource Name field for both datastores.
 - c. 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.
 - d. In the Attributes tab, enter the attribute definitions for this record type.
 - e. One or more attributes 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 Attribute Editor, assign a record code for each record type in the **Record Codes** field.

In [Example 5-1](#), enter `ORD` in the Record Codes field of the `CODE_REC` attribute of the `ORDERS` datastore and enter `LIN` in the Record Codes field of the `CODE_REC` attribute 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 attribute contains the value `ORD`. The same applies to the `ORDER_LINES` datastore (only the records with the first attribute containing the value `LIN` will be returned).

Example 5-1 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

Integrating Data in Files

Files can be used as a source and a target of a mapping. The data integration strategies in Files concern loading from the staging area to Files. The IKM choice in the Integration Knowledge Module tab determines the performances and possibilities for integrating.

Oracle Data Integrator provides two Integration Knowledge Modules for integrating File data:

- [IKM SQL to File Append](#)
- [IKM File to File \(Java\)](#)

IKM SQL to File Append

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 mapping.
- 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.

IKM File to File (Java)

The IKM File to File (Java) is the solution for handling File-to-File use cases. This IKM optimizes the integration performance by generating a Java program to process the files. It can process several source files when the datastore's resource name contains a wildcard. This program is able to run the transformations using several threads.

The IKM File to File (Java) provides a KM option for logging and error handling purposes: BAD_FILE.

This IKM supports flat delimited and fixed files where the fields can be optionally enclosed by text delimiters. EBCDIC and XML formats are not supported.

Using the IKM File to File (Java)

To use the IKM File to File (Java), the staging area must be on a File data server. It is the default configuration when creating a new mapping. The staging area is located on the target, which is the File technology.

The IKM File to File (Java) supports mappings and filters. Mappings and filters are always executed on the source or on the staging area, never on the target. When defining the mapping expressions and filters use the Java syntax. Note that the mapping expressions and filter conditions must be written in a single line with no carriage return. The IKM supports the following standard Java datatypes: string, numeric, and date and accepts any Java transformation on these datatypes.

The following are two examples of a mapping expression:

- `FIC.COL1.toLowerCase()`
- `FIC.COL1+FIC.COL2`

In the second example, if COL1 and COL2 are numeric, the IKM computes the sum of both numbers otherwise it concatenates the two strings.

The following are two examples of a filter condition:

- `FIC.COL1.equals("ORDER")`
- `(FIC.COL1==FIC.COL2)&&(FIC.COL3 !=None)`

The following objects and features are not supported:

- Joins
- Datasets
- Changed Data Capture (CDC)
- Flow Control
- Lookups

Processing Several Files

The IKM File to File (Java) is able to process several source files. To specify several source files use wildcards in the datastore's resource name. You can use the PROCESSED_FILE_PREFIX and PROCESSED_FILE_SUFFIX KM options to manage the source files by renaming them once they are processed.

Using the Logging Features

Once the mapping is completed, Oracle Data Integrator generates the following output files according to the KM options:

- Log file: This file contains information about the loading process, including names of the source files, the target file, and the bad file, as well as a summary of the values set for the major KM options, error messages (if any), statistic information about the processed rows.
- Bad file: This file logs each row that could not be processed. If no error occurs, the bad file is empty.

KM Options

This KM has the following options:

- JAVA_HOME indicates the full path to the bin directory of your JDK. If this options is not set, the ODI Java Home will be used.
- APPEND appends the transformed data to the target file if set to Yes. If set to No, the file is overwritten.

- DISCARDMAX indicates the maximum number of records that will be discarded into the bad file. The mapping fails when the number of discarded records exceeds the number specified in this option.

 **Note:**

Rollback is not supported. The records that have been inserted remain.

- MAX_NB_THREADS indicates the number of parallel threads used to process the data.
- BAD_FILE indicates the bad file name. If this option is not set, the bad file name will be automatically generated and the bad file will be written in the target work schema.
- SOURCE_ENCODING indicates the charset encoding for the source files. Default is the machine's default encoding.
- TARGET_ENCODING indicates the charset encoding for the target file. Default is the machine's default encoding.
- REMOVE_TEMPORARY_OBJECTS removes the log and bad files if set to Yes.
- PROCESSED_FILE_PREFIX indicates the prefix that will be added to the source file name after processing.
- PROCESSED_FILE_SUFFIX indicates the suffix that will be added to the source file name after processing.

Example 5-2 Log File

```
Source File: /xxx/abc.dat
Target File: /yyy/data/target_file.dat
Bad File: /yyy/log/target_file.bad

Header Number to skip: 1
Errors allowed: 3
Insert option: APPEND (could be REPLACE)
Thread: 1

ERROR LINE 100: FIELD COL1 IS NOT A DATE
ERROR LINE 120: UNEXPECTED ERROR

32056 Rows susccessfully read
2000 Rows not loaded due to data filter
2 Rows not loaded due to data errors

30054 Rows successfully loaded
```

6

Generic SQL

It is important to understand 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:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting up an Integration Project](#)
- [Creating and Reverse-Engineering a Model](#)
- [Setting up Changed Data Capture](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)

Introduction

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

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

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 6-1](#) 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 6-1 Generic SQL KMs

Knowledge Module	Description
CKM SQL	<p>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.</p> <p>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.</p>
IKM SQL Control Append	<p>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. Supports Flow Control.</p> <p>Consider using this IKM if you plan to load your SQL compliant target table in replace mode, with or without data integrity check.</p> <p>To use this IKM, the staging area must be on the same data server as the target.</p>
IKM SQL Incremental Update	<p>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 identify the 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. Supports Flow Control.</p> <p>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.</p> <p>To use this IKM, the staging area must be on the same data server as the target.</p>
IKM SQL Incremental Update (row by row)	<p>Integrates data in any AINSI-SQL92 compliant target database in incremental update mode. This IKM processes the data row by row, updates existing rows, and inserts non-existent rows. It isolates invalid data in the Error Table, which can be recycled. When using this IKM with a journalized source table, the deletions can be synchronized. Supports Flow Control.</p>
IKM SQL to File Append	<p>Integrates data in a target file from an ANSI SQL-92 compliant staging area in replace mode. Supports Flow Control.</p> <p>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)</p> <p>To use this IKM, the staging area must be different from the target.</p>
IKM SQL to SQL Control Append	<p>Integrates data into a ANSI-SQL92 target database from any ANSI-SQL92 compliant staging area. Supports Flow Control.</p> <p>This IKM is typically used for ETL configurations: source and target tables are on different databases and the mapping's staging area is set to the logical schema of the source tables or a third schema.</p>

Table 6-1 (Cont.) Generic SQL KMs

Knowledge Module	Description
IKM SQL to SQL Incremental Update	<p>Integrates data from any ANSI-SQL92 compliant database into any ANSI-SQL92 compliant database target table in incremental update mode. Supports Flow Control.</p> <p>This IKM is typically used for ETL configurations: source and target tables are on different databases and the mapping's staging area is set to the logical schema of the source tables or a third schema.</p>
IKM SQL Insert	Integrates data into an ANSI-SQL92 target table in append mode. The data is loaded directly in the target table with a single INSERT SQL statement. Built-in KM.
IKM SQL Update	Integrates data into an ANSI-SQL92 target table in incremental update mode. The data is loaded directly into the target table with a single UPDATE SQL statement. Built-in KM.
IKM SQL Merge	Integrates data into an ANSI-SQL92 target table in incremental update mode. The data is loaded directly into the target table with a single MERGE SQL statement. Built-in KM.
LKM File to SQL	<p>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.</p> <p>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.</p>
LKM SQL to File	Loads and integrates data into a target flat file. This LKM ignores the settings in the IKM. Built-in KM.
LKM SQL to SQL	<p>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.</p> <p>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.</p>
LKM SQL to SQL (Built-in)	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. The extract options specified in the source execution unit will be used to generate source query. Built-in KM.

Table 6-1 (Cont.) Generic SQL KMs

Knowledge Module	Description
LKM SQL to SQL (row by row)	<p>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.</p> <p>The following options are used for the logging mechanism:</p> <ul style="list-style-type: none"> • MAX_ERRORS: Specify the maximum number of errors. The LKM process stops when the maximum number of errors specified in this option is reached. <p>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.</p>
LKM SQL to SQL (JYTHON)	<p>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.</p> <p>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.</p> <p>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.</p>
LKM SQL Multi-Connect	Enables the use of multi-connect IKM for target table. Built-in IKM.
RKM SQL (JYTHON)	<p>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.</p> <p>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.</p>
SKM SQL	<p>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.</p> <p>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</p>

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

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.

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.

Setting up the Topology

Setting up the Topology consists in:

1. [Creating a Data Server](#)
2. [Creating a Physical Schema](#)

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 Administering 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.

Creating a Physical Schema

Create a Physical Schema using the standard procedure, as described in *Creating a Physical Schema* in *Administering Oracle Data Integrator*.

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

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 *Developing Integration Projects with 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.

Creating and Reverse-Engineering a Model

This section contains the following topics:

- [Create a Data Model](#)
- [Reverse-engineer a Data Model](#)

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 *Developing Integration Projects with Oracle Data Integrator*.

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

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 *Developing Integration Projects with 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 Developing Integration Projects with Oracle Data Integrator*. This section details only the fields specific to the usage of the RKM SQL (Jython):

This RKM provides a logging option:

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

Setting up Changed Data Capture

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

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 Flow Control and Static Control in the *Developing Integration Projects with Oracle Data Integrator*.

Designing a Mapping

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

The KM choice for a mapping or a check determines the abilities and performances of this mapping 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.

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 a mapping. The LKM choice in the Loading Knowledge Module tab to load data between an ANSI SQL-92 compliant technology and another type of data server is essential for the performance of a mapping.

Loading Data from an ANSI SQL-92 Compliant Technology

The generic KMs that are listed in [Table 6-2](#) 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 6-2 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	LKM SQL to SQL (Built-in)	Built-in KM for SQL-92 to SQL-92 transfers through the agent using JDBC.
ANSI SQL-92 compliant technology	LKM 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	LKM SQL to SQL (row by row)	This LKM uses row by row logging.
ANSI SQL-92 compliant technology	LKM SQL to File	Built-in KM for SQL-92 to flat file transfers.

Loading Data to an ANSI SQL-92 Compliant Technology

The generic KMs that are listed in [Table 6-3](#) 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 6-3 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
ANSI SQL-92 compliant technology	LKM SQL to SQL (Built-in)	Built-in KM for SQL-92 to SQL-92 transfers through the agent using JDBC.
ANSI SQL-92 compliant technology	LKM 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	LKM SQL to SQL (row by row)	This LKM uses row by row logging.

Integrating Data in an ANSI SQL-92 Compliant Technology

An ANSI SQL-92 compliant technology can be used as a target of a mapping. The IKM choice in the Integration Knowledge Module tab determines the performance and possibilities for integrating.

The KMs listed in [Table 6-4](#) 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 6-4 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. Supports Flow Control.
ANSI SQL-92 compliant technology	IKM SQL Incremental Update	Uses Bulk data movement inside data server. Supports Flow Control.
ANSI SQL-92 compliant technology	IKM SQL Incremental Update (row by row)	Uses Bulk data movement inside data server, processes data row by row. Supports Flow Control.
ANSI SQL-92 compliant technology	IKM SQL Insert	Uses SQL INSERT statement for data movement. Built-in KM.
ANSI SQL-92 compliant technology	IKM SQL Update	Uses SQL UPDATE statement for data movement. Built-in KM.
ANSI SQL-92 compliant technology	IKM SQL Merge	Uses SQL MERGE statement for data movement. Built-in KM.
ANSI SQL-92 compliant technology	IKM SQL to File Append	Uses agent for data movement. Supports Flow Control.
ANSI SQL-92 compliant technology	IKM SQL to SQL Incremental Update	Uses agent or JYTHON for data movement. Supports Flow Control.
ANSI SQL-92 compliant technology	IKM SQL to SQL Control Append	Uses agent for control append strategies. Supports Flow Control.

Designing an ETL-Style Mapping

See [Creating a Mapping](#) in *Developing Integration Projects with Oracle Data Integrator* for generic information on how to design mappings. This section describes how to design an ETL-style mapping where the staging area and target are ANSI SQL-92 compliant.

In an ETL-style mapping, 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 6-5](#) for more information on when to use a multi-connection IKM.

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

1. Create a mapping with an ANSI SQL-92 compliant staging area and target using the standard procedure as described in *Creating a Mapping* in *Developing Integration Projects with Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. Change the staging area for the mapping to the logical schema of the source tables or a third schema. See *Configuring Execution Locations* in *Developing Integration Projects with Oracle Data Integrator* for information about how to change the staging area.
3. In the Physical diagram, select an access point. The Property Inspector opens for this object.
4. In the Loading Knowledge Module tab, select an LKM to load from the source(s) to the staging area. See [Table 6-5](#) to determine the LKM you can use.
5. Optionally, modify the KM options.
6. In the Physical diagram, select the Target by clicking its title. The Property Inspector opens for this object.

In the Integration Knowledge Module, select an ETL multi-connection IKM to load the data from the staging area to the target. See [Table 6-5](#) 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 Check Knowledge Module 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 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 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 mapping.

See [Table 6-5](#) 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 mapping:

1. Create a mapping with an ANSI SQL-92 compliant staging area and target using the standard procedure as described in *Creating a Mapping* in *Developing Integration Projects with Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. Change the staging area for the mapping to the logical schema of the source tables or a third schema. See *Configuring Execution Locations* in *Developing Integration Projects with Oracle Data Integrator* for information about how to change the staging area.
3. In the Physical diagram, select an access point. The Property Inspector opens for this object.
4. In the Loading Knowledge Module tab, select an LKM to load from the source(s) to the staging area. See [Table 6-5](#) to determine the LKM you can use.
5. Optionally, modify the KM options.
6. Select the access point for the Staging Area. The Property Inspector opens for this object.
7. In the Loading Knowledge Module tab, select an LKM to load from the staging area to the target. See [Table 6-5](#) to determine the LKM you can use.
8. Optionally, modify the options.
9. Select the Target by clicking its title. The Property Inspector opens for this object.
10. In the Integration Knowledge Module tab, select a standard mono-connection IKM to update the target. See [Table 6-5](#) to determine the IKM you can use.

Table 6-5 KM Guidelines for ETL-Style Mappings 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 Incremental Update	Mono-connection IKM	Allows an incremental update strategy

7

XML Files

It is important to understand how to work with XML files in Oracle Data Integrator. This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a XML File](#)
- [Designing a Mapping](#)
- [Troubleshooting](#)

Introduction

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

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 [Oracle Data Integrator Driver for XML Reference](#) for more information on this driver.

Pre/Post Processing Support for XML Driver

You can now customize the way data is fed to the XML driver. You can set up intermediate processing stages to process the data that is retrieved from an external endpoint using Oracle Data Integrator, or to write the data out to an external endpoint.

For detailed information about configuring and implement the pre and post processing stages for XML driver, see [Pre/Post Processing Support for XML and Complex File Drivers](#).

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 [Generic SQL](#) or the technology chapters for more information on these KMs.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technologic Specific Requirements

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

Supported Data types

Data type Mappings to and from Oracle will be defined for these data types. Supported data types for XML technology are:

- **BIGINT** — The int data type is the primary integer data type in SQL Server. An integer datatype with a precision of 19 decimal digits. The bigint data type is intended for use when integer values might exceed the range that is supported by the int data type. Bigint fits between smallmoney and int in the data type precedence chart. Functions return bigint only if the parameter expression is a bigint data type.
- **BINARY** — The data types BINARY and BINARY VARYING (VARBINARY) are collectively referred to as binary string types and the values of binary string types are referred to as binary strings. A binary string is a sequence of octets, or bytes. A binary value of NULL appears last (largest) in ascending order. Enables storage

of binary data up to 4,096 bytes. Enables your application to store a bit unconstrained by character semantics.

- **BIT** — An integer data type that can take a value of 1, 0, or NULL. It can also be used to store boolean values because TRUE is convertible to 1 and FALSE is convertible to 0. Converting to bit promotes any nonzero value to 1. If there are 8 or fewer bit columns in a table, the columns are stored as 1 byte. If there are from 9 up to 16 bit columns, the columns are stored as 2 bytes, and so on.
- **BLOB** — A BLOB (binary large object) is a varying-length binary string that can be up to 2,147,483,647 characters long. Like other binary types, BLOB strings are not associated with a code page. In addition, BLOB strings do not hold character data. The length is given in bytes for BLOB unless one of the suffixes K, M, or G is given, relating to the multiples of 1024, 1024*1024, 1024*1024*1024 respectively. Length is specified in bytes for BLOB.
- **CHAR** — The CHAR data type stores character data in a fixed-length field. Character data can be stored as fixed-length or variable-length strings. Fixed-length strings are right-extended with spaces on output; variable-length strings are not extended. Any trailing blank spaces are removed on input, and only restored on output. The default length is 1, and the maximum size is 4,096 bytes.
- **CLOB** — A CLOB (character large object) is used to store unicode character-based data, such as large documents in any character set. A CLOB value can be up to 2,147,483,647 (two giga) characters long. The length is given in number characters for both CLOB, unless one of the suffixes K, M, or G is given, relating to the multiples of 1024, 1024*1024, 1024*1024*1024 respectively. Length is specified in characters (unicode) for CLOB.
- **DATE** — Based on the international ISO-8601:1988 standard date notation: YYYYMMDD where YYYY represents the year in the usual Gregorian calendar, MM is the month between 01 (January) and 12 (December), and DD is the day of the month with a value between 01 and 31. Dates may be represented in any of the following list of base data types. For some data types, a minimum of 4 bytes is required. The list includes: Numeric, String and EBCDIC.
- **DECIMAL** — DECIMAL provides an exact numeric in which the precision and scale can be arbitrarily sized. You can specify the precision (the total number of digits, both to the left and the right of the decimal point) and the scale (the number of digits of the fractional component). The amount of storage required is based on the precision.
- **DOUBLE** — A double precision floating-point data type used in CREATE TABLE and ALTER TABLE statements. The double data type is a double-precision 64-bit IEEE 754 floating point. The Double data type provides the largest and smallest possible magnitudes for a number. The default value of Double is 0. For decimal values, this data type is generally the default choice.

**Note:**

This data type should never be used for precise values, such as currency.

- **FLOAT** — The FLOAT datatype is a floating-point number with a binary precision b. The default precision for this datatype is 126 binary, or 38 decimal. A subtype of the NUMBER datatype having precision p. A FLOAT value is represented

internally as NUMBER. The precision *p* can range from 1 to 126 binary digits. A FLOAT value requires from 1 to 22 bytes.

- **INTEGER** — An INTEGER is an ANSI SQL data type which refers to numeric values which have only an integer portion and no floating point or decimal part. It will only store whole numbers, such as 3, 25, 1987 and so on. The INTEGER data type is usually referred to as NUMBER(38). Its precision can range from 1 to 38. SIMPLE_INTEGER, a subtype of BINARY_INTEGER. Its range is -2147483648 to 2147483648. The SIMPLE_INTEGER cannot store a NULL value.
- **LONGVARBINARY** — Variable-length raw-byte data, such as IP addresses. LONG VARBINARY values are not extended to the full width of the column. Stores data up to 32,000,000 bytes. Use the LONG data types only when you need to store data greater than 65,000 bytes, which is the maximum size for VARBINARY and VARCHAR data types. Such data might include unstructured data, online comments or posts, or small log files.
- **LONGVARCHAR** — Variable-length strings of letters, numbers, and symbols. LONG VARCHAR values are not extended to the full width of the column. Stores data up to 32,000,000 bytes. Use the LONG data types only when you need to store data greater than 65,000 bytes, which is the maximum size for VARBINARY and VARCHAR data types. Such data might include unstructured data, online comments or posts, or small log files.
- **NCHAR and NVARCHAR2**— NCHAR and NVARCHAR2 are Unicode datatypes that store Unicode character data. The character set of NCHAR and NVARCHAR2 datatypes can only be either AL16UTF16 or UTF8 and is specified at database creation time as the national character set. AL16UTF16 and UTF8 are both Unicode encoding. The NCHAR datatype stores fixed-length character strings that correspond to the national character set whereas NVARCHAR2 datatype stores variable length character strings. The maximum length of an NCHAR column is 2000 bytes and it can hold up to 2000 characters. The actual data is subject to the maximum byte limit of 2000. The maximum length of an NVARCHAR2 column is 4000 bytes and it can hold up to 4000 characters. The actual data is subject to the maximum byte limit of 4000. The two size constraints must be satisfied simultaneously at run time for both the data types.
- **NCLOB** — NCLOB (National Character Large Object) is a data type that can hold up to 4 GB of character data. It's similar to a CLOB, but characters are from the national character set. In 12c, the storage limit is extended to $(4*1024*1024*1024-1) * \text{CHUNK size}$ given in the LOB STORAGE clause defining the LOB (database block size by default).
- **NUMBER** — The NUMBER data type stores fixed and floating-point numbers. Numbers of virtually any magnitude can be stored and are guaranteed portable among different systems operating Oracle, up to 38 digits of precision. The following numbers can be stored in a NUMBER column - positive numbers in the range 1×10^{-130} to $9.99\dots9 \times 10^{125}$ with up to 38 significant digits, negative numbers from -1×10^{-130} to $9.99\dots99 \times 10^{125}$ with up to 38 significant digits, zero and positive and negative infinity (generated only by importing from an Oracle Version 5 or later version database).
- **NUMERIC** — The numeric data types store positive and negative fixed and floating-point numbers, zero, infinity, and values that are the undefined result of an operation (that is, is "not a number" or NAN). It comprises of Number data type and Floating Point Numbers. The NUMBER data type stores fixed and floating-point numbers. Numbers of virtually any magnitude can be stored and are guaranteed portable among different systems operating Oracle Database, up to 38 digits of precision.

- **OBJECT** — Object types are abstractions of the real-world entities—for example, purchase orders—that application programs deal with. An object type is a schema object with three kinds of components - a Name, which serves to identify the object type uniquely within that schema, Attributes, which model the structure and state of the real-world entity. Attributes are built-in types or other user-defined types, Methods, which are functions or procedures written in PL/SQL or Java and stored in the database, or written in a language such as C and stored externally. Methods implement operations the application can perform on the real-world entity. An object type is a template. A structured data unit that matches the template is called an object.
- **REAL** — The REAL data type is a floating-point number with a binary precision of 63, or 18 decimal. A REAL is a signed approximate numeric value with a mantissa decimal precision 7. Its absolute value is either zero or between 10^{-38} and 10^{38} . Example - 5600E+12. The REAL data type provides 4 bytes of storage for numbers using IEEE floating-point notation.
- **SMALLINT** — A SMALLINT, small integer type, is an exact numeric value with precision 5 and scale 0, typically 2 bytes or 16 bits. If signed, the range can be -32,768 to +32,767 (SQL_C_SSHORT or SQL_C_SHORT) or, if unsigned, 0 to 65,535 (SQL_C_USHORT). $-32,768 \leq n \leq 32,767$, where n is the value of a SMALLINT. SMALLINT provides 2 bytes of storage.
- **TEXT** — The TEXT data type stores any kind of text data. It can contain both single-byte and multibyte characters that the locale supports. The term simple large object refers to an instance of a TEXT or BYTE data type. A TEXT column has a theoretical limit of 231 bytes (two gigabytes) and a practical limit that your available disk storage determines. No more than 195 columns of the same table can be declared as TEXT data types. The same restriction also applies to BYTE data types. You can store, retrieve, update, or delete the value in a TEXT column.

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

Setting up the Topology

Setting up the topology consists in:

1. [Creating an XML Data Server](#)
2. [Creating a Physical Schema for XML](#)

Creating an XML Data Server

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

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 *Administering 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 7-1](#) lists the key properties of the Oracle Data Integrator Driver for XML. These properties can be specified in the JDBC URL.

See [Oracle Data Integrator Driver for XML Reference](#) for a detailed description of these properties and for a comprehensive list of all properties.

Table 7-1 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.

Table 7-1 (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
```

Creating a Physical Schema for XML

Create an XML physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering 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 in Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a Project using the XML database follows the standard procedure. See *Creating an Integration Project of the Developing Integration Projects with 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

Creating and Reverse-Engineering a XML File

This section contains the following topics:

- [Create an XML Model](#)
- [Reverse-Engineering an XML Model](#)

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 Developing Integration Projects with Oracle Data Integrator*. Select the XML technology and the XML logical schema created when configuring the topology.

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 Developing Integration Projects with 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 *Oracle Data Integrator Driver for XML Reference* for more information.

Designing a Mapping

You can use XML as a source or a target of a mapping.

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

Notes about XML Mappings

Read carefully these notes before working with XML in mappings.

Targeting an XML Structure

When using a datastore of an XML model as a target of a mapping, 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 7-1](#), 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.

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

In [Example 7-1](#) 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 column 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.

Example 7-1 XML Structure

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 mappings and procedures manipulating the XML schema.

See [Oracle Data Integrator Driver for XML Reference](#) for more information on these commands.

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 [Schema Storage](#) for more information on these commands.

Loading Data from and to XML

An XML file can be used as a mapping's source or target. The LKM choice in the Loading Knowledge Module tab that is used to load data between XML files and other types of data servers is essential for the performance of the mapping.

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 7-2](#) lists some examples of KMs that you can use to load from an XML source to a staging area:

Table 7-2 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)
All	LKM SQL to SQL	Generic KM to load data between an ANSI SQL-92 source and an ANSI SQL-92 staging area.

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 mapping 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 7-3](#) lists some examples of KMs that you can use to load from a source to an XML staging area.

Table 7-3 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.

Integrating Data in XML

XML can be used as a target of a mapping. The data integration strategies in XML concern loading from the staging area to XML. The IKM choice in the Integration Knowledge Module 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 7-4](#) 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 7-4 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

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](#)

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 sent to the database to find the source of the error. The code is displayed in the description tab of the task in error.

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.

8

Complex Files

It is important to understand how to work with Complex Files in Oracle Data Integrator. This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Building a Native Schema Description File Using the Native Format Builder](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a Complex File Model](#)
- [Designing a Mapping](#)

Introduction

Oracle Data Integrator supports several files types. This chapter describes how to work with the Complex (or native) File format. See [Files](#) for information about simple fixed or delimited files containing ASCII or EBCDIC data.

For complex files it is possible to build a Native Schema description file that describes the file structure. Using this Native Schema (nXSD) description and the *Oracle Data Integrator Driver for Complex Files*, Oracle Data Integrator is able to reverse-engineer, read and write information from complex files.

See [Building a Native Schema Description File Using the Native Format Builder](#) for information on how to build a native schema description file using the Native Format Builder Wizard, and [Oracle Data Integrator Driver for Complex Files Reference](#) for reference information on the Complex File driver.

Concepts

The *Oracle Data Integrator Driver for Complex Files (Complex File driver)* converts native format to a relational structure and exposes this relational structure as a data model in Oracle Data Integrator.

The Complex File driver translates internally the native file into an XML structure, as defined in the Native Schema (nXSD) description and from this XML file it generates a relational schema that is consumed by Oracle Data Integrator. The overall mechanism is shown in [Figure 8-1](#).

Figure 8-1 Complex File Driver Process



Most concepts and processes that are used for Complex Files are equivalent to those used for XML files. The main difference is the step that transparently translates the Native File into an XML structure that is used internally by the driver but never persisted.

The Complex File technology concepts map the Oracle Data Integrator concepts as follows: A Complex File corresponds to an Oracle Data Integrator data server. Within this data server, a single schema maps the content of the complex file.

The Oracle Data Integrator Driver for Complex File (Complex File driver) loads the complex structure of the native 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 Complex File driver is also able to unload the relational schema back into the complex file. The relational schema is reverse-engineered as a data model in ODI, with tables, columns, and constraints. This model is used like a standard relational data model in ODI. If the modified data within the relational schema needs to be written back to the complex file, the driver provides the capability to *synchronize* the relational schema into the file.

Note that for simple flat files formats (fixed and delimited), it is recommended to use the File technology, and for XML files, the XML technology. See [Files](#) and [XML Files](#) for more information.

Pre/Post Processing Support for Complex File Driver

You can now customize the way data is fed to the Complex File driver. You can set up intermediate processing stages to process the data that is retrieved from an external endpoint using Oracle Data Integrator, or to write the data out to an external endpoint.

For detailed information about configuring and implement the pre and post processing stages for Complex File driver, see [Pre/Post Processing Support for XML and Complex File Drivers](#).

Knowledge Modules

You can use a Complex File data server as any SQL data server. Complex File data servers support both the technology-specific KMs sourcing or targeting SQL data servers, as well as the generic KMs. See [Generic SQL](#) or the technology chapters for more information on these KMs.

You can also use the IKM XML Control Append when writing to a Complex File data server. This Knowledge Module implements specific option to synchronize the data from the relational schema to the file, which is supported by the Complex File driver.

Installation and Configuration

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

- [System Requirements](#)
- [Technology Specific Requirements](#)
- [Connectivity Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technology Specific Requirements

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

Connectivity Requirements

This section lists the requirements for connecting to complex files.

Oracle Data Integrator Driver for Complex Files

Complex files are accessed through the Oracle Data Integrator Driver for Complex File. 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 Native Schema (nXSD) file associated with your native file
- The location of the native complex file

Building a Native Schema Description File Using the Native Format Builder

You can build a Native Schema (nXSD) description file using the Native Format Builder Wizard. You can start the Native Format Builder Wizard from the Data Server Editor when creating the Complex File data server.

To build a native schema description file using the native format builder:

1. In the Topology Navigator expand the **Technologies** node in the Physical Architecture accordion.
2. Select the **Complex File** technology.
3. Right-click and select **New Data Server**.
4. In the JDBC tab, click the **Edit nXSD** button. The Native Format Builder Wizard appears.
5. Follow the on-screen instructions and complete the Native Format Builder Wizard to create a Native Schema description file.

See Native Format Builder Wizard in the *User's Guide for Technology Adapters*, for more information on the Native Schema format.

Setting up the Topology

Setting up the topology consists in:

1. [Creating a Complex File Data Server](#)
2. [Creating a Complex File Physical Schema](#)

Creating a Complex File Data Server

A Complex File data server corresponds to one native file that is accessible to Oracle Data Integrator.

Note:

The use of the In-Memory Engine is not a best practice, and may cause issues, when using the Complex File Technology.

Cause of the error: In-memory HSQL is not recommended for production use. It has memory leaks, beyond the scope of ODI that will eventually bring the JVM down. It is only meant for development use. It is recommended to switch to external DB such as Oracle, MySQL, or MS SQLServer.

Creation of the Data Server

Create a data server for the Complex File technology using the standard procedure, as described in *Creating a Data Server of Administering Oracle Data Integrator*. This section details only the fields required or specific for defining a Complex 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 Complex File data servers.
2. In the JDBC tab, enter the following values:
 - **JDBC Driver:** `oracle.odi.jdbc.driver.file.complex.ComplexFileDriver`
 - **JDBC URL:** `jdbc:snps:complexfile`
 - **Edit nXSD:** Launch the Native Format Builder Wizard if you want to create a Native Schema description file.

For more information on Native Format Builder Wizard, see [Building a Native Schema Description File Using the Native Format Builder](#).

- **Properties:** Configure the properties, such as native file location, native schema, root element, and schema name, for the Oracle Data Integrator Driver for Complex Files.

[Table 8-1](#) lists the key properties of the Oracle Data Integrator Driver for Complex Files. These properties can be specified in JDBC URL.

See [Oracle Data Integrator Driver for Complex Files Reference](#) for a detailed description of these properties and for a comprehensive list of all properties.

Table 8-1 Complex File Driver Properties

Property	Value	Notes
f	<native file name>	Native file location. 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. This parameter is mandatory.
d	<native schema>	Native Schema (nXSD) file location. 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 property can be used for reverse-engineering for example a specific section of the Native Schema. This parameter is mandatory.
s	<schema name>	Name of the relational schema where the complex file will be loaded. This parameter is optional. This schema will be selected when creating the physical schema under the Complex File data server.

Creating a Complex File Physical Schema

Create a Complex File physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering 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 in Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using the Complex File technology follows the standard procedure. See *Creating an Integration Project of Developing Integration Projects with Oracle Data Integrator*.

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

- LKM SQL to SQL
- IKM XML Control Append

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

Creating and Reverse-Engineering a Complex File Model

This section contains the following topics:

- [Create a Complex File Model](#)
- [Reverse-engineer a Complex File Model](#)

Create a Complex File Model

A Complex File model groups a set of datastores. Each datastore typically represents an element in the intermediate XML file generated from the native file using the native schema.

Create a Complex File model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Reverse-engineer a Complex File Model

The Complex File technology supports standard reverse-engineering, which uses only the abilities of the Complex File driver.

Standard Reverse-Engineering

To perform a Standard Reverse- Engineering with a Complex File model use the usual procedure, as described in *Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator*.

This reverse-engineering uses the same process as the reverse-engineering of XML Files. The native schema (nXSD) provided in the data server URL is used as the XSD file to describe the XML structure. See [Reverse-Engineering an XML Model](#) and [XML to SQL Mapping](#) for more information.

Designing a Mapping

You can use a complex file as a source or a target of a mapping.

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

Complex File data models are handled in mappings similarly to XML structures. For example, the Synchronization model is the same for complex files and XML files and the same knowledge modules can be used for both technologies.

See [Designing a Mapping](#) in *XML Files* for more information.

9

Microsoft SQL Server

It is important to understand how to work with Microsoft SQL Server in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a Microsoft SQL Server Model](#)
- [Setting up Changed Data Capture](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 9-1](#) 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 [Generic SQL](#) for more information.

Table 9-1 MSSQL KMs

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 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.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

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.

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.

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.

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>

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.

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](#)

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.

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 Administering 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[:...]]`

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 in Administering 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 in Administering Oracle Data Integrator* and associate it in a given context.

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 Developing Integration Projects with 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

Creating and Reverse-Engineering a Microsoft SQL Server Model

This section contains the following topics:

- [Create a Microsoft SQL Server Model](#)
- [Reverse-engineer a Microsoft SQL Server Model](#)

Create a Microsoft SQL Server Model

Create a Microsoft SQL Server Model using the standard procedure, as described in *Creating a Model of the Developing Integration Projects with Oracle Data Integrator*.

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 Developing Integration Projects with 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 Developing Integration Projects with Oracle Data Integrator*. This section details only the fields specific to the Microsoft SQL Server technology:

1. In the Reverse Engineer 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 attribute 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.

Setting up Changed Data Capture

The ODI Microsoft SQL Server Knowledge Modules support the Changed Data Capture feature. See *Working with Changed Data Capture of Developing Integration Projects with 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 9-2](#) for journalizing Microsoft SQL Server tables.

Table 9-2 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 GoldenGate. See [Oracle GoldenGate](#) for more information.

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 *Flow Control and Static Control in Developing Integration Projects with Oracle Data Integrator* for details.

See [Generic SQL](#) for more information.

Designing a Mapping

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

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

Loading Data from and to Microsoft SQL Server

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

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 9-3](#).

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 9-3 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.

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 9-4](#).

In addition to these KMs, you can also use the [Generic SQL](#) KMs or the KMs specific to the other technology involved.

Table 9-4 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.

Table 9-4 (Cont.) KMs for loading data to 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.
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.

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 9-5](#).

In addition to these KMs, you can also use the [Generic SQL](#) KMs.

The IKM choice in the Integration Knowledge Module tab determines the performances and possibilities for integrating.

Table 9-5 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.

10

Microsoft Excel

It is important to understand how to work with Microsoft Excel in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a Microsoft Excel Model](#)
- [Designing a Mapping](#)

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.

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.

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 [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)

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](#)
- [Specific Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Specific Requirements

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

 **Note:**

ODI does not come with a specific driver to access Microsoft Excel and you have to acquire one from third-party vendors.

To install drivers from third-party vendors, refer to:

- For ODI Studio — [Adding Libraries to ODI Studio](#)

- For Standalone Agent or Collocated Agent — [Adding Libraries to a Standalone or Standalone Collocated Agent](#)
- JEE Agent — [Adding Libraries to a Java EE Agent](#)

Setting up the Topology

Setting up the Topology consists in:

1. [Creating a Microsoft Excel Data Server](#)
2. [Creating a Microsoft Excel Physical Schema](#)

Creating a Microsoft Excel Data Server

Create a data server for the Microsoft Excel technology using the standard procedure, as described in *Creating a Data Server of Administering Oracle Data Integrator*.

In the **JDBC** tab, for **JDBC Driver** and **JDBC URL** parameters, enter the details provided by your driver provider.

Creating a Microsoft Excel Physical Schema

Create a Microsoft Excel Physical Schema using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*. Note that Oracle Data Integrator needs only one physical schema for each Microsoft Excel data server.

Create for this physical schema a logical schema using the standard procedure, as described in *Creating a Logical Schema in Administering 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 a mapping.

Setting Up an Integration Project

Setting up a Project using the Microsoft Excel follows the standard procedure. See *Creating an Integration Project of Developing Integration Projects with 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 [Generic SQL](#) for more information about these KMs.

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](#)

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 Developing Integration Projects with Oracle Data Integrator*

Reverse-engineer a Microsoft Excel Model

Microsoft Excel supports only the Standard reverse-engineering and its capabilities entirely depend on the driver being used.

To perform a Standard Reverse-Engineering on Microsoft Excel use the usual procedure, as described in *Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator*.

Designing a Mapping

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

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

Loading Data From and to Microsoft Excel

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

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 10-1 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.

Table 10-1 (Cont.) KMs to Load from Microsoft Excel

Target or Staging Area	KM	Notes
SQL	LKM SQL to SQL	Loads data from any ISO-92 database to any ISO-92 compliant target database.

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 [Integrating Data in Microsoft Excel](#) for more information on how to integrate data into Microsoft Excel.

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.

11

Microsoft Access

It is important to understand how to work with Microsoft Access in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Concepts](#)
- [Knowledge Modules](#)
- [Specific Requirements](#)

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 mappings.

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.

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 [Generic SQL](#) for more information.

 **Note:**

When reverse engineering MS Access, primary keys are not retrieved. Primary key constraints have to be added manually to the datastores for IKM Access Incremental Update to work correctly.

 **Note:**

Microsoft Access supports only the Standard reverse-engineering and its capabilities entirely depend on the driver being used.

Specific Requirements

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

 **Note:**

ODI does not come with a driver to use Microsoft Access and you have to acquire one from third-party vendors.

To install drivers from third-party vendors, refer to:

- For ODI Studio — [Adding Libraries to ODI Studio](#)
- For Standalone Agent or Collocated Agent — [Adding Libraries to a Standalone or Standalone Collocated Agent](#)
- JEE Agent — [Adding Libraries to a Java EE Agent](#)

To know more about the supported data types, refer to [Microsoft Access documentation](#).

12

Netezza

It is important to understand how to work with Netezza in Oracle Data Integrator. This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a Netezza Model](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 12-1](#) 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 [Generic SQL](#) for more information.

Table 12-1 Netezza KMs

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.
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.

Table 12-1 (Cont.) Netezza KMs

Knowledge Module	Description
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.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

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 mapping.
- 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.

To know more about the supported data types, refer to [Netezza documentation](#).

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.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating a Netezza Data Server](#)
2. [Creating a Netezza Physical Schema](#)

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.

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 Administering 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.

Creating a Netezza Physical Schema

Create a Netezza physical schema using the standard procedure, as described in Creating a Physical Schema in *Administering 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 in *Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using the Netezza database follows the standard procedure. See Creating an Integration Project of *Developing Integration Projects with Oracle Data Integrator*.

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

- CKM NetezzaLKM Netezza Control AppendLKM Netezza Incremental UpdateLKM Netezza To File (EXTERNAL TABLE)LKM File to Netezza (EXTERNAL TABLE)LKM File to Netezza (NZLOAD)RKM Netezza

Creating and Reverse-Engineering a Netezza Model

This section contains the following topics:

- [Create a Netezza Model](#)
- [Reverse-engineer a Netezza Model](#)

Create a Netezza Model

Create a Netezza Model using the standard procedure, as described in Creating a Model of *Developing Integration Projects with Oracle Data Integrator*.

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 Developing Integration Projects with 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 Developing Integration Projects with Oracle Data Integrator*:

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

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

Setting up Data Quality

Oracle Data Integrator provides the CKM Netezza for checking data integrity against constraints defined on a Netezza table. See *Flow Control and Static Control in Developing Integration Projects with Oracle Data Integrator* for details.

Designing a Mapping

You can use Netezza as a source, staging area, or a target of a mapping.

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

Loading Data from and to Netezza

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

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 [Integrating Data in Netezza](#) for more information.

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 12-2](#). In addition to these KMs, you can also use the [Generic SQL](#) KMs or the KMs specific to the other technology involved.

Table 12-2 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.

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 12-3](#). In addition to these KMs, you can also use the [Generic SQL](#) KMs.

The IKM choice in the Integration Knowledge Module tab determines the performances and possibilities for integrating.

Table 12-3 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	<p>Integrates data in a Netezza target table in incremental update mode.</p> <p>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.</p> <p>Please follow Netezza's recommendations on choosing a such a column. Valid options are:</p> <ul style="list-style-type: none"> [PK]: Primary Key of the target table. [UK]: Update key of the mapping [RANDOM]: Random distribution <list of column>: a comma separated list of columns <p>If no value is set (empty), no index will be created.</p> <p>This KM also uses an ANALYZE_TARGET option to generate statistics on the target after integration.</p>

Table 12-3 (Cont.) KMs for integrating data to Netezza

KM	Notes
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.

13

Teradata

It is important to understand how to work with Teradata in Oracle Data Integrator. This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a Teradata Model](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)
- [KM Optimizations for Teradata](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 13-1](#) 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 [Generic SQL](#) for more information.

Table 13-1 Teradata KMs

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.

Table 13-1 (Cont.) Teradata KMs

Knowledge Module	Description
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 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 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 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 mapping'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 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 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.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technology Specific Requirements

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

- FastLoad
- MultiLoad
- T pump
- 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 where 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.

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>

Setting up the Topology

Setting up the Topology consists of:

1. [Creating a Teradata Data Server](#)
2. [Creating a Teradata Physical Schema](#)

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.

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 Administering 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

Creating a Teradata Physical Schema

Create a Teradata physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using the Teradata database follows the standard procedure. See *Creating an Integration Project of Developing Integration Projects with 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

Creating and Reverse-Engineering a Teradata Model

This section contains the following topics:

- [Create a Teradata Model](#)
- [Reverse-engineer a Teradata Model](#)

Create a Teradata Model

Create a Teradata Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

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 Developing Integration Projects with 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 Developing Integration Projects with Oracle Data Integrator*:

1. In the Reverse Engineer 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 constraints.
4. Set REVERSE_COLUMN_CHARACTER_SET to `true` if you want to reverse-engineer VARCHAR and CHAR for a Unicode database as CHAR()CHARACTER SET UNICODE or VARCHAR()CHARACTER SET UNICODE respectively, regardless of the use of CHARACTER SET UNICODE clause at table creation.

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).

Setting up Data Quality

Oracle Data Integrator provides the CKM Teradata for checking data integrity against constraints defined on a Teradata table. See Flow Control and Static Control in *Developing Integration Projects with Oracle Data Integrator* for details.

Oracle Data Integrator provides the Knowledge Module listed in [Table 13-2](#) to perform a check on Teradata.

Table 13-2 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: <ul style="list-style-type: none"> • Primary Indexes • Statistics

Designing a Mapping

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

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

Loading Data from and to Teradata

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

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 [Integrating Data in Teradata](#) for more information.

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 13-3](#). In addition to these KMs, you can also use the [Generic SQL](#) KMs or the KMs specific to the other technology involved.

Table 13-3 KMs for loading data to Teradata

Source or Staging Area Technology	KM	Notes
File	LKM File to Teradata (TTU)	<p>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.</p> <p>Consider using this LKM if your source is a large flat file and your staging area is a Teradata database.</p> <p>This KM support the following Teradata optimizations:</p> <ul style="list-style-type: none"> • Statistics • Optimized Temporary Tables Management

Table 13-3 (Cont.) KMs for loading data to Teradata

Source or Staging Area Technology	KM	Notes
SQL	LKM SQL to Teradata (TTU)	<p>Loads data from a SQL compliant source database to a Teradata staging area database using a native Teradata bulk utility.</p> <p>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.</p> <p>Consider using this IKM when:</p> <ul style="list-style-type: none"> • 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. <p>This KM support the following Teradata optimizations:</p> <ul style="list-style-type: none"> • Support for Teradata Utilities • Support for Named Pipes • Optimized Temporary Tables Management

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 13-4](#). In addition to these KMs, you can also use the [Generic SQL](#) KMs.

The IKM choice in the Integration Knowledge Module tab determines the performances and possibilities for integrating.

Table 13-4 KMs for integrating data to Teradata

KM	Notes
IKM Teradata Control Append	<p data-bbox="829 331 1446 449">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.</p> <p data-bbox="829 457 1446 541">Consider using this IKM if you plan to load your Teradata target table in replace mode, with or without data integrity check.</p> <p data-bbox="829 550 1446 611">To use this IKM, the staging area must be on the same data server as the target Teradata table.</p> <p data-bbox="829 619 1446 646">This KM support the following Teradata optimizations:</p> <ul data-bbox="829 655 1446 716" style="list-style-type: none"> <li data-bbox="829 655 1446 682">• Primary Indexes and Statistics <li data-bbox="829 690 1446 716">• Optimized Temporary Tables Management
IKM Teradata Incremental Update	<p data-bbox="829 724 1446 898">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.</p> <p data-bbox="829 907 1446 991">Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data.</p> <p data-bbox="829 999 1446 1083">Consider using this IKM if you plan to load your Teradata target table to insert missing records and to update existing ones.</p> <p data-bbox="829 1092 1446 1152">To use this IKM, the staging area must be on the same data server as the target.</p> <p data-bbox="829 1161 1446 1188">This KM support the following Teradata optimizations:</p> <ul data-bbox="829 1197 1446 1255" style="list-style-type: none"> <li data-bbox="829 1197 1446 1224">• Primary Indexes and Statistics <li data-bbox="829 1232 1446 1255">• Optimized Temporary Tables Management
IKM Teradata Multi Statement	<p data-bbox="829 1264 1446 1318">Integrates data in Teradata database target table using multi statement requests, managed in one SQL transaction</p>

Table 13-4 (Cont.) KMs for integrating data to Teradata

KM	Notes
IKM Teradata Slowly Changing Dimension	<p>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.</p> <p>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.</p> <p>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.</p> <p>This KM support the following Teradata optimizations:</p> <ul style="list-style-type: none"> • Primary Indexes and Statistics • Optimized Temporary Tables Management <p>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.</p>
IKM Teradata to File (TTU)	<p>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.</p> <p>Consider using this IKM if you plan to transform and export data to a target file from your Teradata server.</p> <p>To use this IKM, the staging area must be different from the target. It should be set to a Teradata location.</p> <p>This KM support the following Teradata optimizations:</p> <ul style="list-style-type: none"> • Support for Teradata Utilities
IKM File to Teradata (TTU)	<p>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.</p> <p>Depending on the utility you choose, you'll have the ability to integrate the data in either replace or incremental update mode.</p> <p>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.</p> <p>To use this IKM, you have to set the staging area to the source file's schema.</p> <p>This KM support the following Teradata optimizations:</p> <ul style="list-style-type: none"> • Primary Indexes and Statistics • Support for Teradata Utilities • Optimized Temporary Tables Management.

Table 13-4 (Cont.) KMs for integrating data to Teradata

KM	Notes
IKM SQL to Teradata (TTU)	<p data-bbox="833 331 1458 422">Integrates data from a SQL compliant database to a Teradata database target table using Teradata Utilities TPUMP, FASTLOAD OR MULTILOAD.</p> <p data-bbox="833 428 1458 688">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.</p> <p data-bbox="833 695 1458 785">Depending on the utility you choose, you'll have the ability to integrate the data in replace or incremental update mode.</p> <p data-bbox="833 791 1156 819">Consider using this IKM when:</p> <ul data-bbox="833 825 1458 1003" style="list-style-type: none"> <li data-bbox="833 825 1458 884">• You plan to load your target table with few transformations on the source <li data-bbox="833 890 1458 949">• All your source tables are on the same data server (used as the staging area) <li data-bbox="833 955 1458 1003">• You don't want to stage your data between the source and the target <p data-bbox="833 1010 1458 1068">To use this IKM, you have to set the staging area to the source data server's schema.</p> <p data-bbox="833 1075 1403 1102">This KM support the following Teradata optimizations:</p> <ul data-bbox="833 1108 1338 1241" style="list-style-type: none"> <li data-bbox="833 1108 1203 1136">• Primary Indexes and Statistics <li data-bbox="833 1142 1187 1169">• Support for Teradata Utilities <li data-bbox="833 1176 1149 1203">• Support for Named Pipes <li data-bbox="833 1209 1338 1241">• Optimized Temporary Tables Management
IKM SQL to Teradata Control Append	<p data-bbox="833 1247 1458 1337">Integrates data from an ANSI-92 compliant source database into Teradata target table in truncate / insert (append) mode.</p> <p data-bbox="833 1344 1458 1488">This IKM is typically used for ETL configurations: source and target tables are on different databases and the mapping's staging area is set to the logical schema of the source tables or a third schema. See Designing an ETL-Style Mapping for more information.</p>

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 typically enable the parallel execution of simple mappings. The Teradata performance is improved by synchronized scans and by avoiding transient journal.

Set the KM options as follows:

- Mappings using this KM must be used within a package:
 - In the first mapping of the package loading a table via the multi-statement set the `INIT_MULTI_STATEMENT` option to `YES`.
 - The subsequent mappings loading a table via the multi-statement must use this KM and have the `INIT_MULTI_STATEMENT` option set to `NO`.
 - The last mapping 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 mapping.
- 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 mapping (`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.

Designing an ETL-Style Mapping

See *Creating a Mapping in [Developing Integration Projects with Oracle Data Integrator](#)* for generic information on how to design mappings. This section describes how to design an ETL-style mapping where the staging area is on a Teradata database or any ANSI-92 compliant database and the target on Teradata.

In an ETL-style mapping, 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 [Generic SQL](#) for more information.

See [Table 13-5](#) for more information on when to use a multi-connection IKM.

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

1. Create a mapping with an ANSI-92 compliant staging area and the target on Teradata using the standard procedure as described in *Creating a Mapping in Developing Integration Projects with Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. Change the staging area for the mapping to the logical schema of the source tables or a third schema. See *Configuring Execution Locations in Developing Integration Projects with Oracle Data Integrator* for information about how to change the staging area.
3. In the Physical diagram, select an access point. The Property Inspector opens for this object.
4. In the Loading Knowledge Module tab, select an LKM to load from the source(s) to the staging area. See [Table 13-5](#) to determine the LKM you can use.
5. Optionally, modify the KM options.
6. In the Physical diagram, select the Target by clicking its title. The Property Inspector opens for this object.
7. In the Integration Knowledge Module tab, select an ETL multi-connection IKM to load the data from the staging area to the target. See [Table 13-5](#) 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 mapping.

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 mapping. The target is Teradata.

See [Table 13-5](#) 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 mapping:

1. Create a mapping with an ANSI-92 compliant staging area and the target on Teradata using the standard procedure as described in *Creating a Mapping in Developing Integration Projects with Oracle Data Integrator*. This section describes only the ETL-style specific steps.
2. Change the staging area for the mapping to the logical schema of the source tables or a third schema. See *Configuring Execution Locations in the Developing Integration Projects with Oracle Data Integrator* for information about how to change the staging area.

3. In the Physical diagram, select an access point. The Property Inspector opens for this object.
4. In the Loading Knowledge Module tab, select an LKM to load from the source(s) to the staging area. See [Table 13-5](#) to determine the LKM you can use.
5. Optionally, modify the KM options.
6. Select the access point for the Staging Area. The Property Inspector opens for this object.
7. In the Loading Knowledge Module tab, select an LKM to load from the staging area to the target. See [Table 13-5](#) to determine the LKM you can use.
8. Optionally, modify the options.
9. Select the Target by clicking its title. The Property Inspector opens for this object.

In the Integration Knowledge Module tab, select a standard mono-connection IKM to update the target. See [Table 13-5](#) to determine the IKM you can use.

Table 13-5 KM Guidelines for ETL-Style Mappings 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.

Table 13-5 (Cont.) KM Guidelines for ETL-Style Mappings with Teradata Data

Source	Staging Area	Target	Exporting LKM	IKM	KM Strategy	Comment
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
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

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](#)

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 mapping. 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.

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 mapping.

- **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 mapping.

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

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.

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.

14

Hypersonic SQL

It is important to understand how to work with Hypersonic SQL in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a Hypersonic SQL Model](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 14-1](#) 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 for more information.

Table 14-1 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.

Table 14-1 (Cont.) Hypersonic SQL Knowledge Modules

Knowledge Module	Description
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.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technology Specific Requirements

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

To know more about the supported data types, refer to [Hypersonic SQL documentation](#).

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.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating a Hypersonic SQL Data Server](#)
2. [Creating a Hypersonic SQL Physical Schema](#)

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 *Administering 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:hsqldb://<host>:<port>`

The URL parameters are:

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

Creating a Hypersonic SQL Physical Schema

Create a physical schema using the standard procedure, as described in *Creating a Physical Schema* in *Administering Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in *Creating a Logical Schema* in *Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using the Hypersonic SQL database follows the standard procedure. See *Creating an Integration Project* of *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 for more information about these KMs.

Creating and Reverse-Engineering a Hypersonic SQL Model

This section contains the following topics:

- [Create a Hypersonic SQL Model](#)
- [Reverse-engineer a Hypersonic SQL Model](#)

Create a Hypersonic SQL Model

Create a Hypersonic SQL Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

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*.

Setting up Changed Data Capture

The ODI Hypersonic SQL Knowledge Modules support the Changed Data Capture feature. See *Working with Changed Data Capture of Developing Integration Projects with 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 14-2](#) for journalizing Hypersonic SQL tables.

Table 14-2 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.

Setting up Data Quality

Oracle Data Integrator provides the CKM HSQL for checking data integrity against constraints defined on a Hypersonic SQL table. See Flow Control and Static Control in *Developing Integration Projects with Oracle Data Integrator* for details.

Oracle Data Integrator provides the Knowledge Module listed in [Table 14-3](#) to perform a check on Hypersonic SQL.

Table 14-3 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.

Designing a Mapping

You can use Hypersonic SQL as a source, staging area or a target of a mapping.

The KM choice for a mapping or a check determines the abilities and performance of this mapping 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 KMs or the KMs specific to the other technologies used as source, target, or staging area.

15

IBM Informix

It is important to understand how to work with IBM Informix in Oracle Data Integrator. This chapter includes the following sections:

- [Introduction](#)
- [Concepts](#)
- [Knowledge Modules](#)
- [Specific Requirements](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 15-1](#) 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 for more information.

Table 15-1 IBM Informix Knowledge Modules

Knowledge Module	Description
IKM Informix Incremental Update	<p>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.</p> <p>Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data.</p> <p>Consider using this IKM if you plan to load your IBM Informix target table to insert missing records and to update existing ones.</p> <p>To use this IKM, the staging area must be on the same data server as the target.</p>
JKM Informix Consistent	<p>Creates the journalizing infrastructure for consistent journalizing on IBM Informix tables using triggers.</p> <p>Enables Consistent Set Changed Data Capture on IBM Informix.</p> <p>The source database must have transaction logging enabled to use this KM.</p>
JKM Informix Simple	<p>Creates the journalizing infrastructure for simple journalizing on IBM Informix tables using triggers.</p> <p>Enables Simple Changed Data Capture on IBM Informix.</p>
LKM Informix to Informix (SAME SERVER)	<p>Loads data from a source Informix database to a target Informix staging area located inside the same server.</p> <p>This LKM creates a view in the source database and a synonym in the staging area database. This method is often more efficient than the standard "LKM SQL to SQL" when dealing with large volumes of data.</p> <p>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.</p> <p>Both databases must have the same logging mode enabled to use this KM.</p>
RKM Informix	<p>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.</p> <p>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.</p>
RKM Informix SE	<p>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.</p> <p>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.</p>
SKM Informix	<p>Generates data access Web services for IBM Informix databases. See SKM SQL in for more details.</p>

Specific Requirements

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

To know more about the supported data types, refer to [IBM Informix documentation](#).

16

IBM DB2 for iSeries

It is important to understand how to work with IBM DB2 for iSeries in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering an IBM DB2/400 Model](#)
- [Setting up Changed Data Capture](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)
- [Specific Considerations with DB2 for iSeries](#)
- [Troubleshooting](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 16-1](#) 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 [Generic SQL](#) for more information.

Table 16-1 DB2 for iSeries KMs

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 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.
RKM DB2 400	Retrieves metadata for IBM DB2 for iSeries: physical files, tables, views, foreign keys, unique keys.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

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 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.

To know more about the supported data types, refer to [IBM DB2 for iSeries documentation](#).

Setting up the Topology

Setting up the Topology consists of:

1. [Creating a DB2/400 Data Server](#)
2. [Creating a DB2/400 Physical Schema](#)

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.

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 Administering 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
 - Host (Data Server): Name or IP address of the host
 - User/Password: DB2 user with its password
2. In the JDBC tab:
 - JDBC Driver: `weblogic.jdbc.db2.DB2Driver`
 - JDBC URL: `jdbc:as400://<host>[;libraries=<library>] [;<property>=<value>...]`

The URL parameters are:

- `<host>`: server network name or IP address
- `<library>`: default library or collection to access
- `<property>=<value>`: connection properties. Refer to the driver's documentation for a list of available properties.

Creating a DB2/400 Physical Schema

Create an IBM DB2/400 physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering 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 in Administering Oracle Data Integrator* and associate it in a given context.

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 Developing Integration Projects with 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 Slowly Changing Dimension
- JKM DB2 400 Consistent
- JKM DB2 400 Simple
- RKM DB2 400
- CKM SQL

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](#)

Create an IBM DB2/400 Model

Create an IBM DB2/400 Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

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 Developing Integration Projects with 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 Developing*

Integration Projects with 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>.

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 [Setting up Trigger-Based CDC](#) for more information.
- **Log-based CDC by reading the native iSeries transaction journals.** This method does not support Consistent Set CDC and requires a platform-specific configuration. See [Setting up Trigger-Based CDC](#) for more information.

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 16-2](#) for journalizing IBM DB2 for iSeries tables using triggers.

See Working with Changed Data Capture of *Developing Integration Projects with Oracle Data Integrator* for details on how to set up journalizing and how to use captured changes.

Table 16-2 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.

Setting up Data Quality

Oracle Data Integrator provides the generic CKM SQL for checking data integrity against constraints defined in DB2/400. See Flow Control and Static Control in *Developing Integration Projects with Oracle Data Integrator* for details.

See [Generic SQL](#) for more information.

Designing a Mapping

You can use IBM DB2 for iSeries as a source, staging area or a target of a mapping.

The KM choice for a mapping or a check determines the abilities and performance of this mapping 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.

Loading Data from and to IBM DB2 for iSeries

IBM DB2 for iSeries can be used as a source, target or staging area of a mapping. The LKM choice in the Mapping Flow tab to load data between IBM DB2 for iSeries and another type of data server is essential for the performance of a mapping.

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 16-3](#).

In addition to these KMs, you can also use the [Generic SQL](#) KMs.

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

Table 16-3 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 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 attributes 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.

Specific Considerations with DB2 for iSeries

This section provides specific considerations when using Oracle Data Integrator in an iSeries environment.

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.

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.

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.

To configure a data server with the driver wrapper, 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...]]`

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](#)

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.

Common Problems and Solutions

This section describes common problems and solutions.

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://192.0.2.1;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 mapping).
- 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 &l 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.

17

IBM DB2 UDB

It is important to understand how to work with IBM DB2 UDB in Oracle Data Integrator. This chapter includes the following sections:

- [Introduction](#)
- [Concepts](#)
- [Knowledge Modules](#)
- [Specific Requirements](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 17-1](#) 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 [Generic SQL](#) for more information

Table 17-1 DB2 UDB KMs

Knowledge Module	Description
IKM DB2 UDB Incremental Update	<p>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.</p> <p>Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data.</p> <p>Consider using this IKM if you plan to load your IBM DB2 UDB target table to insert missing records and to update existing ones.</p> <p>To use this IKM, the staging area must be on the same data server as the target.</p>
IKM DB2 UDB Slowly Changing Dimension	<p>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.</p> <p>Because inserts and updates are done in bulk set-based processing, this IKM is optimized for large volumes of data.</p> <p>Consider using this IKM if you plan to load your IBM DB2 UDB target table as a Type II Slowly Changing Dimension.</p> <p>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.</p>
JKM DB2 UDB Consistent	<p>Creates the journalizing infrastructure for consistent journalizing on IBM DB2 UDB tables using triggers.</p> <p>Enables Consistent Changed Data Capture on IBM DB2 UDB.</p>
JKM DB2 UDB Simple	<p>Creates the journalizing infrastructure for simple journalizing on IBM DB2 UDB tables using triggers.</p> <p>Enables Simple Changed Data Capture on IBM DB2 UDB.</p>
LKM DB2 UDB to DB2 UDB (EXPORT_IMPORT)	<p>Loads data from an IBM DB2 UDB source database to an IBM DB2 UDB staging area database using the native EXPORT / IMPORT commands.</p> <p>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 is often more efficient than the standard LKM SQL to SQL when dealing with large volumes of data.</p> <p>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.</p>

Table 17-1 (Cont.) DB2 UDB KMs

Knowledge Module	Description
LKM File to DB2 UDB (LOAD)	<p>Loads data from a File to a DB2 UDB staging area database using the native CLP LOAD Command.</p> <p>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.</p> <p>Consider using this LKM if your source is a large flat file and your staging area is an IBM DB2 UDB database.</p>
LKM SQL to DB2 UDB	<p>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 Generic SQL except that you can specify some additional specific IBM DB2 UDB parameters.</p>
LKM SQL to DB2 UDB (LOAD)	<p>Loads data from any ANSI SQL-92 standard compliant source database to an IBM DB2 UDB staging area using the CLP LOAD command.</p> <p>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.</p> <p>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.</p>
SKM IBM UDB	<p>Generates data access Web services for IBM DB2 UDB databases. See SKM SQL in Generic SQL for more information.</p>

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.

To know more about the supported data types, refer to [IBM DB2 UDB documentation](#).

18

Salesforce.com

It is important to understand how to work with Salesforce.com in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a Salesforce.com Model](#)
- [Designing a Mapping](#)

Introduction

Oracle Data Integrator (ODI) seamlessly integrates with Salesforce.com. Oracle Data Integrator features are designed to work best with Salesforce.com, including reverse-engineering and mappings.

Concepts

The Salesforce.com database concepts map the Oracle Data Integrator concepts as follows: A Salesforce.com 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) to connect to a Salesforce.com data server. See [Connectivity Requirements](#) for more details.

Knowledge Modules

Oracle Data Integrator provides no Knowledge Module (KM) specific to the Salesforce.com technology. You can use the generic SQL KMs to perform the data integration and transformation operations of Salesforce.com data. See [Generic SQL](#) for more information.

Installation and Configuration

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

- [System Requirements and Certifications](#)
- [Technology Specific Requirements](#)
- [Connectivity Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>

Technology Specific Requirements

There are no technology-specific requirements for using Salesforce.com in Oracle Data Integrator.

To know more about the supported data types, refer to [Salesforce.com documentation](#).

Connectivity Requirements

This section lists the requirements for connecting to a Salesforce.com database.

JDBC Driver

Oracle Data Integrator uses the Salesforce.com JDBC Driver to connect to a Salesforce.com database.

Setting up the Topology

Setting up the topology consists of:

- [Creating a Salesforce.com Data Server](#)
- [Creating a Physical Schema for Salesforce.com Data Server](#)

Creating a Salesforce.com Data Server

Create a data server for the Salesforce.com technology using the standard procedure, as described in *Creating a Data Server* of *Administering Oracle Data Integrator*. This section details only the fields required or specific for defining a Salesforce.com data server:

1. In the Definition tab:
 - **Name:** Name of the data server that will appear in Oracle Data Integrator
 - **Instance/dblink (Data Server):** Not required for Salesforce.com. Leave this field blank.
 - **User/Password:** User name and password for connecting to the data server
2. In the JDBC tab:
 - **JDBC Driver:** `weblogic.jdbc.sforce.SForceDriver`

- **JDBC URL:** The URL used for connecting to the data server. For example, `jdbc:weblogic:sforce://login.salesforce.com`.
3. In the Properties section:
- **ConfigOptions:** The configuration options that you want to use. For example, `(AuditColumns=all;MapSystemColumnNames=0;)`.
 - **DatabaseName:** The instance of the database. This needs to be changed as per the JDBC URL used.

 **Note:**

For more information on the connection properties supported by the Salesforce.com driver, see http://media.datadirect.com/download/docs/jdbc/alljdbc/help.html#page/jdbccconnect%2FConnection_Properties_11.html%23wwID0EZT5Y.

Creating a Physical Schema for Salesforce.com Data Server

An Oracle Data Integrator physical schema corresponds to a pair of schemas:

- A Data Schema into which Oracle Data Integrator will look for the source and target data structures for the mapping.
- A Work Schema into which Oracle Data Integrator can create and manipulate temporary work data structures associated with the sources and targets contained in the data schema.

Create a physical schema for the Salesforce.com data server using the standard procedure, as described in Creating a Physical Schema in *Administering Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in Creating a Logical Schema in *Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using Salesforce.com follows the standard procedure. See Creating an Integration Project of *Developing Integration Projects with Oracle Data Integrator*.

Import the following generic SQL KMs into your project for getting started with Salesforce.com:

- IKM SQL to SQL Control Append
- IKM SQL to SQL Incremental Update

See [Generic SQL](#) for more information about these KMs.

 **Note:**

The following KMs are available in the system by default:

- LKM SQL to Oracle (Built-In)
- LKM SQL to SQL (Built-In)
- LKM SQL Multi-Connect
- IKM Oracle Insert
- IKM Oracle Update

Creating and Reverse-Engineering a Salesforce.com Model

This section contains the following topics:

- [Create a Salesforce.com Model](#)
- [Reverse-engineer a Salesforce.com Model](#)

Create a Salesforce.com Model

Create a Salesforce.com model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Reverse-engineer a Salesforce.com Model

Salesforce.com supports Standard reverse-engineering - which uses only the abilities of the JDBC driver.

To perform a Standard reverse-engineering on Salesforce.com, use the usual procedure, as described in *Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator*.

Designing a Mapping

You can use Salesforce.com as a source or a target of a mapping, but not as the staging area.

The KM choice for a mapping or a check determines the abilities and performances of this mapping or check. The recommendations below help in the selection of the KM for different situations concerning a Salesforce.com server.

Loading Data from and to Salesforce.com

Salesforce.com can be used as a source or a target of a mapping. The LKM choice in the Mapping Flow tab to load data between Salesforce.com and another type of data server is essential for the performance of a mapping.

Loading Data from Salesforce.com

Oracle Data Integrator does not provide specific knowledge modules for Salesforce.com. 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 Salesforce.com to any staging area.

Table 18-1 KMs to Load from Salesforce.com

Target or Staging Area	KM	Notes
Oracle	LKM SQL to Oracle	Loads data from any ANSI SQL-92 source database to an Oracle staging area.
SQL	LKM SQL to SQL	Loads data from an ANSI SQL-92 compliant database for 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.

Loading Data to Salesforce.com

Because Salesforce.com cannot be used as staging area, you cannot use a LKM to load data into Salesforce.com. See [Integrating Data in Salesforce.com](#) for more information on how to integrate data into Salesforce.com.

Integrating Data in Salesforce.com

Oracle Data Integrator does not provide specific knowledge modules for Salesforce.com. Use the Generic SQL KMs or the KMs specific to the technology used as the staging area. For integrating with Salesforce.com, only the IKMs that do not require a LKM and that do not require the staging area to be set on target can be used. The following table lists the generic SQL KMs that can be used for integrating data from a staging area to Salesforce.com.

Table 18-2 KMs for Integrating Data to Salesforce.com

KM	Notes
IKM SQL to SQL Control Append	Integrates data into an ANSI-SQL92 target database from any ANSI-SQL92 compliant staging area. This IKM is typically used for ETL configurations: source and target tables are in different databases and the mapping's staging area is set to the logical schema of the source tables or a third schema.

Table 18-2 (Cont.) KMs for Integrating Data to Salesforce.com

KM	Notes
IKM SQL to SQL Incremental Update	<p>Integrates data from any ANSI-SQL92 compliant database into any AINSISQL92 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 mapping's staging area is set to the logical schema of the source tables or a third schema.</p> <p>To use this IKM, the <code>FLOW_TABLE_LOCATION</code> option should be set to <code>STAGING</code>.</p>

19

Sybase IQ

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

NOT_SUPPORTED:

This chapter applies only to Data Integration Platform Cloud.

This chapter includes the following sections:

- [Introduction](#)
- [Concepts](#)
- [Knowledge Modules](#)
- [Specific Requirements](#)

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.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in the below table 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 5, [Generic SQL](#) for more information.

Note:

Generic SQL KMs for Update, Incremental Update and Merge cannot be used to incrementally load or update Sybase IQ target table. Use IKM Sybase IQ Incremental Update.

Table 19-1 Sybase IQ Knowledge Modules

Knowledge Module	Description
CKM Sybase IQ	<p>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.</p> <p>Consider using this KM, if you plan to check data integrity on a Sybase IQ database.</p>
IKM Sybase IQ Incremental Update	<p>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.</p> <p>Inserts and updates are done in bulk set-based processing to maximize performance. Therefore, this IKM is optimized for large volumes of data.</p> <p>Consider using this IKM if you plan to load your Sybase IQ target table to insert missing records and to update existing ones.</p> <p>To use this IKM, the staging area must be on the same data server as the target.</p>
IKM Sybase IQ Slowly Changing Dimension	<p>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.</p> <p>Because inserts and updates are done in bulk set-based processing, this IKM is optimized for large volumes of data.</p> <p>Consider using this IKM if you plan to load your Sybase IQ target table as a Type II Slowly Changing Dimension.</p> <p>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.</p>
LKM File to Sybase IQ (LOAD TABLE)	<p>Loads data from a File to a Sybase IQ staging area database using the <code>LOAD TABLE SQL</code> command.</p> <p>Because this method uses the native <code>LOAD TABLE</code> 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.</p> <p>Consider using this LKM if your source is a large flat file and your staging area is a Sybase IQ database.</p>

Table 19-1 (Cont.) Sybase IQ Knowledge Modules

Knowledge Module	Description
LKM SQL to Sybase IQ (LOAD TABLE)	<p>Loads data from any ANSI SQL-92 standard compliant source database to a Sybase IQ staging area database using the native <code>LOAD TABLE SQL</code> command.</p> <p>This LKM unloads the source data in a temporary file and calls the Sybase IQ <code>LOAD TABLE SQL</code> command to populate the staging table. Because this method uses the native <code>LOAD TABLE</code>, it is often more efficient than the LKM SQL to SQL method when dealing with large volumes of data.</p> <p>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.</p>

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.

To know more about the supported data types, refer to [Sybase IQ documentation](#).

Part II

Business Intelligence

It is important to understand how to work with Business Intelligence in Oracle Data Integrator.

Part II contains the following chapters:

- [Oracle Business Intelligence Enterprise Edition](#)
- [Oracle Business Intelligence Cloud Service](#)
- [Oracle Hyperion Planning](#)
- [Oracle Hyperion Essbase](#)

20

Oracle Business Intelligence Enterprise Edition

It is important to understand how to work with Oracle Business Intelligence Enterprise Edition in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering an Oracle BI Model](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)

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 mappings.

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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 20-1](#) for handling Oracle BI data. These KMs use Oracle BI specific features.

Table 20-1 Oracle BI KMs

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.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technology Specific Requirements

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

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.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating an Oracle BI Data Server](#)
2. [Creating an Oracle BI Physical Schema](#)

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.

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 Administering 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.

Creating an Oracle BI Physical Schema

Create a Oracle BI physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering 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 in Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using an Oracle BI Server follows the standard procedure. See *Creating an Integration Project of Developing Integration Projects with 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 SQLIKM Oracle BI to SQL Append

Import also the knowledge modules (IKM, CKM) required for the other technologies involved in your project.

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](#)

Create an Oracle BI Model

Create an Oracle BI Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

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 Developing Integration Projects with Oracle Data Integrator*. This section details only the fields specific to the Oracle BI technology:

1. In the Reverse Engineer tab of the Oracle BI Model, select the KM: RKM Oracle BI (Jython).<project name>.

This KM implements the USE_LOG logging option to trace the reverse-engineering process.

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.

Designing a Mapping

You can use Oracle BI as a source of a mapping.

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

Loading Data from and to Oracle BI

The LKM choice in the Loading Knowledge Module tab to load data between Oracle BI and another type of data server is essential for the performance of a mapping.

Loading Data from Oracle BI

Use the knowledge modules listed in [Table 20-2](#) to load data from an Oracle BI server to a target or staging area database.

Table 20-2 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.

Table 20-2 (Cont.) KMs for loading data From Oracle BI

Staging Area/Target Technology	KM	Notes
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 mapping 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.

Loading Data to Oracle BI

Oracle BI cannot be used as a staging area. No LKM targets Oracle BI.

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.

21

Oracle Business Intelligence Cloud Service

It is important to understand how to work with Oracle Business Intelligence Cloud Service (BICS) in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Setting up the Topology](#)
- [Reverse Engineering a BICS Model](#)
- [Designing a Mapping](#)

Introduction

Oracle Business Intelligence Cloud Service (BICS) uses entities called Datasets and Tables for storing data that then get used in an analytics solution.

Table 21-1 Datasets versus Tables

Datasets	Tables
Does not have index	Can have index
Creation and insertion of data can be a single operation	Creation and insertion of data are two distinct steps with different payloads
Loading data supports batching explicitly	Loading data involves more fine-grained controls For example, maximum number of errors to be allowed, while loading Also, more controls exist over column definition

 **Note:**

Both the Datasets and the Tables have the parameters 'firstBatch' and 'lastBatch.' They are backed by a DBCS schema. The DBCS schema information is not published. Data is loaded into Datasets/Tables as application/octet-stream format part of a multi-part message. The stream can be Text stream with delimiters or Java object array stream. ODI will load data only into the BICS Dataset or the Table. ODI will not read data from the BICS Dataset or the Table.

For both the Dataset and the Table, you must define a BICS target. As each of these entities are bound to a different URL endpoint, an ODI Datastore container will be bound to either Dataset or Table, but never to both. This implies that an ODI Model (and by inference the associated Logical and Physical Schema) can be only bound to either the Datasets endpoint or the Tables endpoint.

Since loading data into a BICS target involves Mappings, you must model a BICS Dataset or Table as a Datastore in ODI. BICS Logical Schema cannot be used for staging, and BICS Datastores cannot be used as source in a Mapping.

Setting up the Topology

Setting up the topology consists of:

- [Creating an Oracle BICS Data Server](#)
- [Creating an Oracle BICS Physical Schema](#)

Creating an Oracle BICS Data Server

BICS Dataserver defines the endpoint URL and the dataloader suffix. The data source suffix part of the URL depends on whether we are exploring Datasets or Tables and will be exposed in the Physical Schema page. This will allow a single BICS Dataserver to work with both the Dataset and the Table.

The Data Server page contains fields for the Dataserver name, base URI, username, password, and the Identity domain.

The following is a full BICS URI:

```
https://service-identity_domain.analytics.data_center.oraclecloud.com/resource-path
```

The base URI is the BICS service instance's first paths segment.

The following is a base URI:

```
https://service-identity_domain.analytics.data_center.oraclecloud.com
```

The Data loader path field is a constant, auto-filled. This enables you to see the path segment.

The following is a data loader path segment:

```
/dataload/v1
```

Creating an Oracle BICS Physical Schema

Once the BICS Dataserver is configured, you can configure its Physical Schema. BICS Physical Schema will prompt for choosing either Dataset or Table. This in turn will control the Resource URI.

**Note:**

Resource URI can be chosen from the list or typed in, but once chosen/typed in and then saved, it cannot be edited again.

Choice of whether the Physical Schema is to be bound to BICS Tables or Datasets triggers the association of REST Operations. The Operations are unique and pre-defined for Datasets and Tables.

Reverse Engineering a BICS Model

Once the BICS Logical Schema is set, you can create a Model based on this Logical Schema, and then reverse engineer. You must select the RKM Oracle BI Cloud Service to reverse engineer the BICS tables or datasets metadata.

 **Note:**

After reverse engineering, make sure to manually fix the column datatypes, after seeing the BICS Table/Dataset.

Table 21-2 KM Options

Option	Type	Default	Description
GET_TABLE_INDEXES	Table	True	Whether or not to retrieve table indexes.
DEFAULT_DIRECTORY	Table	java.lang.System.getPrope rty("java.io.tmpdir")	Directory for generated temporary (return) files by REST calls. All temporary data files generated by REST calls will be deleted at the end of RKM execution.

Designing a Mapping

Similar to the IKMs for Hyperion, BICS IKM is also multi-connect. It uses batching capabilities of the BICS Dataset/Table.

 **Note:**

The IKM SQL to Oracle BI Cloud Service does not support loading the Oracle SDO_GEOMETRY data type column to the BICS target table. Oracle BI Cloud Service cannot be used as the staging area, and does not support incremental update or flow/static check. Therefore, the following KMs will not work with the Oracle BI Cloud Service technology:

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

BICS Datastore as target for Mapping

The IKM SQL to Oracle BI Cloud Service exposes Dataset/Table loading options as KM options.

Table 21-3 Supported KM Options

Option	Type	Default	Description
TRUNCATE_TARGET_TABLE	Boolean	False	Deletes data before starting to load data. This is only applicable for BICS Table.
DROP_TARGET	Boolean	False	Drops the target Table/Dataset before starting to load data.

Table 21-3 (Cont.) Supported KM Options

Option	Type	Default	Description
CREATE_TARGET	Boolean	False	If the target Table/Dataset does not exist, creates it first.


 **Note**: The REST API Dataset loader semantic table is created

Table 21-3 (Cont.) Supported KM Options

Option	Type	Default	Description
--------	------	---------	-------------


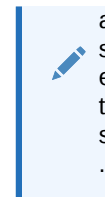

 e
 a
 t
 e
 -
 i
 f
 -
 d
 o
 e
 s
 -
 n
 o
 t
 -
 e
 x
 i
 s
 t
 ,
 .
 S
 o
 t
 h
 i
 s
 s
 e
 t
 t
 i
 n
 g
 i
 s
 o
 p
 t
 i
 o
 n
 a
 l
 f
 o
 r
 D
 a
 t

Table 21-3 (Cont.) Supported KM Options

Option	Type	Default	Description
DATA_WRITE_MODE	Choice	Insert all	Choice between Insert all, Insert missing, Upsert, Update only. This is applicable only if the target is BICS Table. Choosing Upsert/Update only will fail, if the BICS Table does not have unique indexes.
NUM_RETRIES	Text	0	Each dataload batch operation could error out. This is a numeric option that will allow retry. Default is not to retry at all.
RETRY_DELAY	Text	5	Time delay in seconds between each retry attempt.
REMOVE_DUPLICATES	Boolean	False	Applicable only for BICS Table to indicate whether or not to remove duplicate data from within the batch that is being sent. Does not touch data already in the BICS Table.
BATCH_SIZE	Text	1000	Number of rows to be send at one time (in one POST request).
VALIDATE_COLUMNS	Boolean	False	Whether to validate the BICS target's column names before trying to load data.
MAX_ERR_PER_BATCH	Text	0	Maximum number of errors per batch that Oracle BICS will allow. Applicable only for Tables.
TRACE_FILE	Text	Empty	Location of file to which trace of all the REST calls made by the IKM are logged. If left empty, no trace will be created.



 **Note:**

Datasets

- BICS Datasets do not have indexes.
- No unique index errors will be raised on loading data.
- Only possible errors are when data does not match the datatype of the target column.

Tables

- BICS Tables support unique indexes.
- The insert/update modes depend on unique indexes being present and being part of the data load operation.
- 'Remove duplicates' also requires unique indexes.
- Insert missing/Update only/Upsert all require unique index be part of the data load
- 'Insert all' does not need unique index as long as the columns involved in the data load are nullable.

Oracle Hyperion Planning

It is important to understand how to work with Oracle Hyperion Planning in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up Hyperion Planning Adapter](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering a Planning Model](#)
- [Designing a Mapping](#)
- [Datastore Tables and Data Load Columns](#)

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.

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 [Setting up the Topology](#).
- Reverse-engineering a Planning application using the adapter's Reverse-engineering Knowledge Module (RKM)
See [Creating and Reverse-Engineering a Planning Model](#).
- Loading metadata and data into the Planning application using the adapter's Integration Knowledge Module (IKM)
See [Designing a Mapping](#).

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 22-1](#) 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.

Table 22-1 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 mappings. 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.

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](#)
- [Setting up Hyperion Planning Adapter](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technology Specific Requirements

There are no technology-specific requirements for using the Oracle Data Integrator Adapter for Planning.

Connectivity Requirements

There are no connectivity-specific requirements for using the Oracle Data Integrator Adapter for Planning.

Setting up Hyperion Planning Adapter

The following sections explain how to set up Hyperion Planning Adapter for ODI Studio and ODI standalone agent.

Setting up Adapter for ODI Studio

Exit from ODI Studio before setting up Hyperion Planning Adapter.

1. In Oracle Hyperion Planning directory, locate HspJS.jar
2. If HspJS.jar is not directly accessible by ODI, copy it to a location that allows ODI access.
3. Modify <ODI_HOME>/odi/studio/bin/odi.conf file to include HspJS.jar.

For Example:

```
AddJavaLibFile /server/lib/HspJS.jar
```

Setting up Adapter for ODI Standalone Agent

Stop ODI Agent before setting up Hyperion Planning Adapter.

1. In Oracle Hyperion Planning directory, locate HspJS.jar
2. Copy it into <DOMAIN_HOME>/lib directory.

For more information, see *Configuring the Domain for the Standalone Collocated Agent* in *Installing and Configuring Oracle Data Integrator*.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating an Hyperion Planning Data Server](#)
2. [Creating an Hyperion Planning Physical Schema](#)

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 *Administering 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.

Creating an Hyperion Planning Physical Schema

Create a Hyperion Planning physical schema using the standard procedure, as described in *Creating a Physical Schema of Administering 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 Administering Oracle Data Integrator* and associate it in a given context.

Creating and Reverse-Engineering a Planning Model

This section contains the following topics:

- [Create a Planning Model](#)
- [Reverse-engineer a Planning Model](#)

Create a Planning Model

Create a Planning Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

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 Developing Integration Projects with 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

Designing a Mapping

After reverse-engineering a Planning application as a model, you can use the datastores in this model as targets of mappings for loading data and metadata into the application.

The KM choice for a mapping determines the abilities and performance of this mapping. 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](#)

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 mapping for each dimension that you load. You can chain mappings to load metadata into several dimensions at once.

Note:

Please note the following:

- 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](#).
- If the REFRESH_DATABASE option in IKM SQL to Hyperion is set to Yes and refresh Essbase database operation fails with an error, the execution is still shown as successful in the Operator tab. However, the errors are reported in the log file. In situations when the Essbase database is not refreshed and the execution is successful in the Operator tab, check the log file for errors.

To load metadata into a Planning application:

1. Create a mapping. Make sure that you select the IKM SQL to Hyperion Planning on the Flow tab.
2. Specify the load options as described in [Load Options](#).
3. Run the mapping to load the metadata into the application
4. Validate the dimension:
 - a. Log on to Planning Web.
 - b. Select **Administration > Dimensions**.

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 a mapping for loading data.

 **Note:**

You can use the same mapping for loading metadata and data. [Load Options](#) lists the options of the IKM SQL to Hyperion Planning

3. Check the Operator log to see if the mapping 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.

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 logged to 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

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.

Accounts

[Table 22-2](#) describes the columns of the Accounts table. See [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 22-2 Accounts

Column	Description
Account	<p>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.</p> <p>The value for this field must meet these requirements:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string.</p>

Table 22-2 (Cont.) Accounts

Column	Description
Parent	<p>Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension.</p> <p>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.</p> <p>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.</p> <p>The record is not loaded if one of the following situations occurs:</p> <ul style="list-style-type: none"> • 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	<p>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.</p> <p>The value for this column must meet the following requirements for a successful load:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string; default value: a null string.</p>
Additional Alias	<p>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.</p>
Data Storage	<p>Takes the storage attribute for the member being loaded.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • Store • Dynamic Calc • Dynamic Calc and Store • Shared • Never Share (default) • Label Only <p>This value is passed as a string.</p>

Table 22-2 (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	<p>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.)</p> <p>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:</p> <ul style="list-style-type: none"> • Revenue-Flow • Expense-Flow • Asset-Balance • Liability-Balance • Equity-Balance <p>Note: When Time Balance is Flow, records with any valid Skip Values are loaded, but Skip Value is disabled for all account types.</p>
Skip Value	<p>Skip Value Takes 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:</p> <ul style="list-style-type: none"> • 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 <p>Note: When Time Balance is Flow, records with any valid Skip Values are loaded, but Skip Value is disabled for all Account types.</p>

Table 22-2 (Cont.) Accounts

Column	Description
Data Type	<p>Takes the data storage value. Valid values:</p> <ul style="list-style-type: none"> • 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 <i>mm/dd/yyyy</i> or <i>dd/mm/yyyy</i> • Text-Stores and displays the member's data value as text. • Unspecified-Stores and displays the member's data value as "unspecified." <p>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.</p>
Exchange Rate Type	<p>Takes the exchange rate. This column is dependent on the value specified for the Data Type column. Valid values:</p> <ul style="list-style-type: none"> • Average, Ending, and Historical when Data Type is equal to Currency • None when Data Type is equal to Non-currency or Percentage <p>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:</p> <ul style="list-style-type: none"> • Revenue-Average • Expense-Average • Asset-Ending • Liability-Ending • Equity-Ending • Saved Assumption-None
Use 445	<p>Indicates the distribution selected in the Planning application. If the application has no distribution, this column is not displayed.</p> <p>Valid values are 0 and 1 (or any number other than 0); default value: 1.</p>

Table 22-2 (Cont.) Accounts

Column	Description
Variance Reporting	<p>Takes a value for account members with an account type of Saved Assumption or if the record is rejected. Valid values:</p> <ul style="list-style-type: none"> • 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. <p>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.</p> <p>For Account types, the value is set to the following:</p> <ul style="list-style-type: none"> • Revenue-Non-Expense • Expense-Expense • Asset-Non-Expense • Liability-Non-Expense • Equity-Non-Expense
Source Plan Type	<p>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.</p> <p>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.</p> <p>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."</p> <p>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).</p> <p>Note: If a Source Plan Type is specified in the adapter but is not valid for the parent, the record is rejected.</p>
Plan Type (<i>Plan1</i>)	<p>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.</p>

Table 22-2 (Cont.) Accounts

Column	Description
Aggregation (<i>Plan1</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan1. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • . • * • / • % • ~ • Never
Plan Type (<i>Plan 2</i>)	<p>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.</p>
Aggregation (<i>Plan2</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan2. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • . • * • / • % • ~ • Never
Plan Type (<i>Plan3</i>)	<p>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.</p>
Aggregation (<i>Plan3</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan3. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • . • * • / • % • ~ • Never

Table 22-2 (Cont.) Accounts

Column	Description
Plan Type (<i>Wrkforce</i>)	For Workforce Planning: The Plan Type (<i>Wrkforce</i>) 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 (<i>Wrkforce</i>)	For Workforce Planning: The Aggregation (<i>Wrkforce</i>) 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: <ul style="list-style-type: none"> • + (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: <ul style="list-style-type: none"> • + (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 22-2 (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	<p>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 associated with a dimension but only one Smart List associated with a dimension member.</p> <p>These predefined Smart Lists are available in a Workforce Planning application:</p> <ul style="list-style-type: none"> • None • Status • FT_PT • HealthPlan • TaxRegion • Month • Performance • Position • EmployeeType
Description	<p>Takes a description for the member that is being loaded. By default, the Description column is empty.</p> <p>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.</p>
Operation	<p>Takes any of these values:</p> <ul style="list-style-type: none"> • 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. <p>Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.</p>

Employee

[Table 22-3](#) describes the columns of the Employee table. See [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 22-3 Employee

Column	Description
Employee	<p>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.</p> <p>The value for this field must meet these requirements:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string.</p>
Parent	<p>Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension.</p> <p>When you load data for a member and specify a different parent member than the parent member in the application, the member is updated with the parent value that you specify.</p> <p>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.</p> <p>The record is not loaded if one of the following situations occurs:</p> <ul style="list-style-type: none"> • The specified parent is a descendant of the member that you are loading. • The specified parent does not exist in the Planning application.

Table 22-3 (Cont.) Employee

Column	Description
Default Alias	<p>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.</p> <p>The value for this column must meet the following requirements for a successful load:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string; default value: a null string.</p>
Additional Alias	<p>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.</p>
Data Storage	<p>Takes the storage attribute for the member being loaded.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • Store • Dynamic Calc • Dynamic Calc and Store • Shared • Never Share (default) • Label Only <p>This value is passed as a string.</p>
Valid for Consolidation	<p>The column is ignored.</p>
Two Pass Calculation	<p>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.</p>

Table 22-3 (Cont.) Employee

Column	Description
Data Type	<p>Takes the data storage value. Valid values:</p> <ul style="list-style-type: none"> • 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 <i>mm/dd/yyyy</i> or <i>dd/mm/yyyy</i> • Text-Stores and displays the member's data value as text. • Unspecified-Stores and displays the member's data value as "unspecified." <p>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.</p>
Custom Attribute	<p>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.</p>
Aggregation (<i>Plan1</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan1. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • . • * • / • % • ~ • Never

Table 22-3 (Cont.) Employee

Column	Description
Aggregation (<i>Plan2</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan2. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Aggregation (<i>Plan3</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan3. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Aggregation (<i>Wrkforce</i>)	<p>For Workforce Planning: The Aggregation (<i>Wrkforce</i>) 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never

Table 22-3 (Cont.) Employee

Column	Description
Aggregation (Capex)	<p>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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Member Formula	<p>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.</p>
UDA	<p>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.</p>
Smart Lists	<p>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 associated with a dimension but only one Smart List associated with a dimension member.</p> <p>These predefined Smart Lists are available in a Workforce Planning application:</p> <ul style="list-style-type: none"> • None • Status • FT_PT • HealthPlan • TaxRegion • Month • Performance • Position • EmployeeType
Description	<p>Takes a description for the member that is being loaded; empty by default.</p> <p>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.</p>

Table 22-3 (Cont.) Employee

Column	Description
Operation	<p>Takes any of these values:</p> <ul style="list-style-type: none"> • 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. <p>Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.</p>

Entities

[Table 22-4](#) describes the columns of the Entities table. See [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 22-4 Entities

Column	Description
Entity	<p>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.</p> <p>The value for this column must meet the following requirements for a successful load:</p> <p>The value for this field must meet these requirements:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string.</p>

Table 22-4 (Cont.) Entities

Column	Description
Parent	<p>Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension.</p> <p>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.</p> <p>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.</p> <p>The record is not loaded if one of the following situations occurs:</p> <ul style="list-style-type: none"> • 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	<p>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.</p> <p>The value for this column must meet the following requirements for a successful load:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string; default value: a null string.</p>
Additional Alias	<p>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.</p>

Table 22-4 (Cont.) Entities

Column	Description
Data Storage	<p>Takes the storage attribute for the member being loaded.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • Store • Dynamic Calc • Dynamic Calc and Store • Shared • Never Share (default) • Label Only <p>This value is passed as a string.</p>
Two Pass Calculation	<p>Boolean value to indicate if the member being loaded has the Two-Pass Calculation attribute associated in the Planning application. 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.</p>
Data Type	<p>Takes the data storage value. Valid values:</p> <ul style="list-style-type: none"> • 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 <i>mm/dd/yyyy</i> or <i>dd/mm/yyyy</i> • Text-Stores and displays the member's data value as text. • Unspecified-Stores and displays the member's data value as "unspecified." <p>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.</p>
Base Currency	<p>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.</p>
Plan Type (<i>Plan1</i>)	<p>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.</p>

Table 22-4 (Cont.) Entities

Column	Description
Aggregation (<i>Plan1</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan1. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Plan Type (<i>Plan2</i>)	<p>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.</p>
Aggregation (<i>Plan2</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan2. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Plan Type (<i>Plan 3</i>)	<p>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.</p>
Aggregation (<i>Plan3</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan3. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never

Table 22-4 (Cont.) Entities

Column	Description
Aggregation (<i>Wrkforce</i>)	<p>For Workforce Planning: The Aggregation (<i>Wrkforce</i>) 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Aggregation (<i>Capex</i>)	<p>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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Custom Attribute	<p>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.</p>
Member Formula	<p>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.</p>
UDA	<p>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.</p>

Table 22-4 (Cont.) Entities

Column	Description
Smart Lists	<p>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 associated with a dimension but only one Smart List associated with a dimension member.</p> <p>These predefined Smart Lists are available in a Workforce Planning application:</p> <ul style="list-style-type: none"> • None • Status • FT_PT • HealthPlan • TaxRegion • Month • Performance • Position • EmployeeType
Description	<p>Takes a description for the member that is being loaded; empty by default.</p> <p>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.</p>
Operation	<p>Takes any of these values:</p> <ul style="list-style-type: none"> • 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. <p>Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.</p>

User-Defined Dimensions

Table 22-5 describes the columns of the User-Defined Dimensions table.

Table 22-5 User-Defined Dimensions

Column	Description
Entity	<p>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.</p> <p>The value for this column must meet the following requirements for a successful load:</p> <p>The value for this field must meet these requirements:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string.</p>
Parent	<p>Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension.</p> <p>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.</p> <p>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.</p> <p>The record is not loaded if one of the following situations occurs:</p> <ul style="list-style-type: none"> • The specified parent is a descendant of the member that you are loading. • The specified parent does not exist in the Planning application.

Table 22-5 (Cont.) User-Defined Dimensions

Column	Description
Default Alias	<p>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.</p> <p>The value for this column must meet the following requirements for a successful load:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string; default value: a null string.</p>
Additional Alias	<p>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.</p>
Data Storage	<p>Takes the storage attribute for the member being loaded.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • Store • Dynamic Calc • Dynamic Calc and Store • Shared • Never Share (default) • Label Only <p>This value is passed as a string.</p>
Two Pass Calculation	<p>Boolean value to indicate if the member being loaded has the Two-Pass Calculation attribute associated in the Planning application. 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.</p>

Table 22-5 (Cont.) User-Defined Dimensions

Column	Description
Data Type	<p>Takes the data storage value. Valid values:</p> <ul style="list-style-type: none"> • 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 <i>mm/dd/yyyy</i> or <i>dd/mm/yyyy</i> • Text-Stores and displays the member's data value as text. • Unspecified-Stores and displays the member's data value as "unspecified." <p>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.</p>
Aggregation (<i>Plan1</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan1. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • . • * • / • % • ~ • Never
Aggregation (<i>Plan2</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan2. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • . • * • / • % • ~ • Never

Table 22-5 (Cont.) User-Defined Dimensions

Column	Description
Aggregation (<i>Plan3</i>)	<p>Takes the aggregation option for the member being loaded as related to Plan3. 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Aggregation (<i>Wrkforce</i>)	<p>For Workforce Planning: The Aggregation (<i>Wrkforce</i>) 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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Aggregation (<i>Capex</i>)	<p>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.</p> <p>This value is passed as a string. Valid values:</p> <ul style="list-style-type: none"> • + (default) • • * • / • % • ~ • Never
Custom Attribute	<p>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.</p>

Table 22-5 (Cont.) User-Defined Dimensions

Column	Description
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	<p>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 associated with a dimension but only one Smart List associated with a dimension member.</p> <p>These predefined Smart Lists are available in a Workforce Planning application:</p> <ul style="list-style-type: none"> • None • Status • FT_PT • HealthPlan • TaxRegion • Month • Performance • Position • EmployeeType
Description	<p>Takes a description for the member that is being loaded; empty by default.</p> <p>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.</p>
Operation	<p>Takes any of these values:</p> <ul style="list-style-type: none"> • 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. <p>Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.</p>

Attribute Dimensions

Table 22-6 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 22-6 Attribute Dimensions

Column	Description
Entity	<p>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.</p> <p>The value for this column must meet the following requirements for a successful load:</p> <p>The value for this field must meet these requirements:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string.</p>
Parent	<p>Takes the name of the parent of the member you are loading. It is used to create the hierarchy in the dimension.</p> <p>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.</p> <p>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.</p> <p>The record is not loaded if one of the following situations occurs:</p> <ul style="list-style-type: none"> • The specified parent is a descendant of the member that you are loading. • The specified parent does not exist in the Planning application.

Table 22-6 (Cont.) Attribute Dimensions

Column	Description
Default Alias	<p>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.</p> <p>The value for this column must meet the following requirements for a successful load:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string; default value: a null string.</p>
Additional Alias	<p>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.</p>
Operation	<p>Takes any of these values:</p> <ul style="list-style-type: none"> • 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. <p>Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.</p>

UDA

Table 22-7 describes the columns of the UDA table.

Table 22-7 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	<p>Takes the values of the UDA you are loading. The value for this column must meet the following requirements for a successful load:</p> <p>The value for this column must meet the following requirements for a successful load:</p> <ul style="list-style-type: none"> • 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 <i>Hyperion Essbase - System 9 Database Administrator's Guide</i> or Essbase online help. <p>This value is passed as a string; default value: a null string.</p>
Operation	<p>Takes any of these values:</p> <ul style="list-style-type: none"> • 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. <p>Note: If you delete a member, that member, its data, and any associated planning units are permanently removed and cannot be restored.</p>

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 22-8 Data Load Columns

Columns	Description
Data Load Cube Name	<p>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:</p> <ul style="list-style-type: none"> • Plan1 • Plan2 • Plan3 • Wkforce • Capex
Driver Member	<p>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.</p> <p>Note: The Smart List field on this load method does not affect this column.</p>
Point-of-View	<p>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.</p>
Column	Description
Data Load Cube Name	<p>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:</p> <ul style="list-style-type: none"> • Plan1 • Plan2 • Plan3 • Wkforce • Capex

Column	Description
Driver Member	<p>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.</p> <p>Note: The Smart List field on this load method does not affect this column.</p>
Point-of-View	<p>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.</p>

Oracle Hyperion Essbase

It is important to understand how to work with Oracle Hyperion Essbase in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up Hyperion Essbase Adapter](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Essbase Model](#)
- [Designing a Mapping](#)

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.

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 [Setting up the Topology](#).
- Reverse-engineering an Essbase application using the Reverse-engineering Knowledge Module (RKM)
See [Creating and Reverse-Engineering an Essbase Model](#).
- Extracting metadata and data using Load Knowledge Modules (LKM).
See [Designing a Mapping](#)
- Integrating the metadata and data into the Essbase application using the Integration Knowledge Modules (IKM).
See [Designing a Mapping](#)

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 23-1](#) 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.

Table 23-1 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 mappings
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.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technology Specific Requirements

There are no technology-specific requirements for using the Oracle Data Integrator Adapter for Essbase.

Connectivity Requirements

There are no connectivity-specific requirements for using the Oracle Data Integrator Adapter for Essbase.

Setting up Hyperion Essbase Adapter

The following sections explain how to set up Hyperion Essbase Adapter for ODI Studio and ODI standalone agent.

Setting up Adapter for ODI Studio

Exit from ODI Studio, before starting the setup process.

1. In Oracle Hyperion Essbase directory, locate `ess_japi.jar` and `ess_es_server.jar`.
2. If `ess_japi.jar` and `ess_es_server.jar` are not directly accessible by ODI, copy them to a location that allows ODI access.
3. Modify `<ODI_HOME>/odi/studio/bin/odi.conf` file to include `ess_japi.jar` and `ess_es_server.jar`.

For example:

```
AddJavaLibFile /server/lib/ess_japi.jar
```

```
AddJavaLibFile /server/lib/ess_es_server.jar
```

Setting up Adapter for ODI Standalone Agent

Stop ODI Agent before starting the setup process.

1. In Oracle Hyperion Essbase directory, locate `ess_japi.jar` and `ess_es_server.jar`.
2. Copy them into `<DOMAIN_HOME>/lib` directory.

For more information, see *Configuring the Domain for the Standalone Collocated Agent* in *Installing and Configuring Oracle Data Integrator*.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating an Hyperion Essbase Data Server](#)
2. [Creating an Hyperion Essbase Physical Schema](#)

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 *Administering 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.

Creating an Hyperion Essbase Physical Schema

Create a Hyperion Essbase physical schema using the standard procedure, as described in *Creating a Physical Schema of Administering 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 Administering Oracle Data Integrator* and associate it in a given context.

Creating and Reverse-Engineering an Essbase Model

This section contains the following topics:

- [Create an Essbase Model](#)
- [Reverse-engineer an Essbase Model](#)

Create an Essbase Model

Create an Essbase Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

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 Developing Integration Projects with 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 23-2](#).

Table 23-2 RKM Hyperion Essbase Options

Option	Possible Values	Description
MULTIPLE_DATA_COLUMNS	<ul style="list-style-type: none"> • No (Default) • Yes 	<p>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 <code>DATA_COLUMN_DIMENSION</code> option and as many data columns as specified by the comma separated list for the <code>DATA_COLUMN_MEMBERS</code> option.</p>
DATA_COLUMN_DIMENSION	Account	<p>This option is only applicable if <code>MULTIPLE_DATA_COLUMNS</code> 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.</p>
DATA_COLUMN_MEMBERS	Account	<p>This option is only applicable if <code>MULTIPLE_DATA_COLUMNS</code> is set to Yes. Separate the required data column members with, (Comma).</p> <p>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:</p> <ul style="list-style-type: none"> • Scenario (String) • Product (String) • Market (String)Year (String) • Sales (Numeric) • COGS (Numeric)

Table 23-2 (Cont.) RKM Hyperion Essbase Options

Option	Possible Values	Description
EXTRACT_ATTRIBUTEMEMBERS	<ul style="list-style-type: none"> No (Default) Yes 	<p>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.</p> <p>If this option is set to Yes, then the data model contains these columns.</p> <ul style="list-style-type: none"> 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.

Designing a Mapping

After reverse-engineering an Essbase application as a model, you can use the datastores in this model in these ways:

- Targets of mappings for loading data and metadata into the application
- Sources of mappings for extracting metadata and data from the application.

The KM choice for a mapping determines the abilities and performance of this mapping. 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](#)

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 mapping for each dimension that you load. You can chain mappings to load metadata into several dimensions at once.

1.

 **Note:**

The metadata datastore can also be modified by adding or deleting 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 mapping within the samples is an example of performing a metadata load using a generational dimension build rule.

Table 23-3 lists the options of the IKM SQL to Hyperion Essbase (METADATA). These options define how the adapter loads metadata into an Essbase application.

Table 23-3 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: <ul style="list-style-type: none"> • Comma • Tab • Space • Custom character; for example, @, #, ^
RESTRUCTURE_DATABASE	<ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • KEEP_ALL_DATA- Keep all the data • KEEP_INPUT_DATA Keep onlyinput data • KEEP_LEVEL0_DATA-Keep onlylevel 0 data • DISCARD_ALL_DATA-Discard alldata 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.

Table 23-3 (Cont.) IKM SQL to Hyperion Essbase (METADATA) Options

Option	Values	Description
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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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.
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.

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 23-4 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 23-4 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: <ul style="list-style-type: none"> • Comma • Tab • Space • Custom character; for example, @, #, ^
GROUP_ID	Integer	When performing multiple data loads in parallel, many mappings 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 mapping 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.

Table 23-4 (Cont.) IKM SQL to Hyperion Essbase (DATA)

Option	Values	Description
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	<ul style="list-style-type: none"> • None (Default) • All • Upper Blocks • Non-input Blocks 	<p>Enable this option to clear data from the Essbase cube before loading data into it.</p> <p>These are the valid values:</p> <ul style="list-style-type: none"> • 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 <p>Note: For ASO applications, the Upper Blocks and Non-Input Blocks options will not be applicable.</p>
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	<ul style="list-style-type: none"> • No (Default) • Yes 	<p>This option is only applicable if you have specified a calculation script in the CALCULATION_SCRIPT option.</p> <p>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.</p>
PRE_LOAD_MAXL_SCRIPT	Blank (Default)	<p>Enable this option to execute a MAXL script before loading data to the Essbase cube.</p> <p>Specify a fully qualified path name (without blank spaces) for the MAXL script file.</p> <p>Note: Essbase client must be installed and configured on the machine where the Oracle Data Integrator Agent is running.</p>
POST_LOAD_MAXL_SCRIPT	Blank (Default)	<p>Enable this option to execute a MAXL script after loading data to the Essbase cube.</p> <p>Specify a fully qualified path name (without blank spaces) for the MAXL script file.</p> <p>Note: Essbase client must be installed and configured on the machine where the Oracle Data Integrator Agent is running.</p>

Table 23-4 (Cont.) IKM SQL to Hyperion Essbase (DATA)

Option	Values	Description
ABORT_ON_PRE_MAXL_ERROR	<ul style="list-style-type: none"> No (Default) Yes 	<p>This option is only applicable if you are enabling the PRE_LOAD_MAXL_SCRIPT option.</p> <p>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.</p>
MAXIMUM_ERRORS_ALLOWED	1 (Default)	<p>Enable this option to set the maximum number of errors to be ignored before stopping a data load.</p> <p>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.</p>
COMMIT_INTERVAL	1000 (Default)	<p>Commit Interval is the chunk size of records that are loaded in the Essbase cube in a complete batch.</p> <p>Enable this option to set the Commit Interval for the records in the Essbase cube.</p> <p>Changing the Commit Interval can increase performance of data load based on design of the Essbase database.</p>
LOG_ENABLED	<ul style="list-style-type: none"> 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	<pre><? =java.lang.System. getProperty("java.io .tmpdir")?/< %=snpRef.getTargetTable("RES_NAME")%>.log (Default)</pre>	Specify a file name to log events of the IKM process.
LOG_ERRORS	<ul style="list-style-type: none"> 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	<pre><? =java.lang.System. getProperty(java.io. .tmpdir")?>/< %=snpRef.getTargetTable("RES_NAME")%>.err</pre>	Specify a file name to log error record details of the IKM process.
ERR_LOG_HEADER_ROW	<ul style="list-style-type: none"> 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.

Table 23-4 (Cont.) IKM SQL to Hyperion Essbase (DATA)

Option	Values	Description
ERR_COL_DELIMIT ER	, (Default)	Specify the column delimiter to be used for the error records file.
ERR_ROW_DELIMIT ER	\r\n (Default)	Specify the row delimiter to be used for the error records file.
ERR_TEXT_DELIMI TER	' (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.

Extracting Data

This section includes the following topics:

- [Data Extraction Methods for Essbase](#)
- [Extracting Essbase Data](#)
- [Extracting Members from Metadata](#)

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.

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 23-5 provides the options of the LKM Hyperion Essbase Data to SQL. These options define how Oracle Data Integrator Adapter for Essbase extracts data.

Table 23-5 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	<ul style="list-style-type: none"> ReportScript (Default) MDXQuery CalcScript 	<p>Specify an extraction query type—report script, MDX query, or calculation script.</p> <p>Provide a valid extraction query, which fetches all the data to fill the output columns.</p> <p>The first record (first two records in case of calculation script) contains the meta information of the extracted data.</p>
EXTRACTION_QUERY_FILE	Blank (Default)	Specify a fully qualified file name of the extraction query.
EXT_COL_DELIMITER	\t (Default)	<p>Specify the column delimiter for the extraction query.</p> <p>If no value is specified for this option, then space (" ") is considered as column delimiter.</p>
EXTRACT_DATA_FILE_IN_CALC_SCRIPT	Blank (Default)	<p>This option is only applicable if the query type in the EXTRACTION_QUERY_TYPE option is specified as CalcScript.</p> <p>Specify a fully qualified file location where the data is extracted through the calculation script.</p>
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	<ul style="list-style-type: none"> No (Default) Yes 	<p>This option is only applicable if the PRE_EXTRACT_MAXL option is enabled.</p> <p>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.</p>
LOG_ENABLED	<ul style="list-style-type: none"> 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	<pre><? =java.lang.System.get Property ("java.io.tmpdir")?/<% =snpRef.getTargetTab le("RES_NAME") %>.log (Default)</pre>	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.

Table 23-5 (Cont.) LKM Hyperion Essbase DATA to SQL Options

Option	Values	Description
LOG_ERRORS	<ul style="list-style-type: none"> 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	<pre><? =java.lang.System.get Property(java.io.tmpdir)"?>< %=snpRef.getTargetT able("RES_NAME") %>.err</pre>	Specify a file name to log error record details of the LKM process.
ERR_LOG_HEADER_ROW	<ul style="list-style-type: none"> 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	\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	<ul style="list-style-type: none"> 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.

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 23-6](#) 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 23-6 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: <ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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.get tProperty(java.io.tmpd ir")?>/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	<ul style="list-style-type: none"> 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.get tProperty(java.io.tmpd ir")?>/Extract_ %=snpRef.getFrom() %>.err	Specify a file name to log error record details of the LKM process.
ERR_LOG_HEADER_ROW	<ul style="list-style-type: none"> 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 23-6 (Cont.) LKM Hyperion Essbase METADATA to SQL

Option	Values	Description
ERR_TEXT_DELIMIT ER	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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.

Part III

Other Technologies

It is important to understand how to work with other technologies in Oracle Data Integrator.

Part III contains the following chapters:

- [JMS](#)
- [JMS XML](#)
- [LDAP Directories](#)
- [Oracle TimesTen In-Memory Database](#)
- [Oracle GoldenGate](#)
- [Oracle SOA Suite Cross References](#)

24

JMS

It is important to understand how to work with Java Message Services (JMS) in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Defining a JMS Model](#)
- [Designing a Mapping](#)
- [JMS Standard Properties](#)

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 [JMS XML](#).

Concepts

The JMS Knowledge Modules apply to most popular JMS compliant middleware, including Oracle Service Bus, Sonic MQ, and so forth. Most of these Knowledge Modules include transaction handling to ensure message delivery.

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 (JMSTDestination), 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).

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 or 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 mapping.

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 attributes in the mappings (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 attributes - , 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 attributes 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.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 24-1](#) for handling JMS messages.

Table 24-1 JMS Knowledge Modules

Knowledge Module	Description
IKM SQL to JMS Append	<p>Integrates data into a JMS compliant message queue or topic in text or binary format from any SQL compliant staging area.</p> <p>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).</p> <p>To use this IKM, the staging area must be different from the target.</p>
LKM JMS to SQL	<p>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.</p> <p>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.</p> <p>Consider using this LKM if one of your source datastores is a text or binary JMS message.</p>

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

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.

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.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating a JMS Data Server](#)
2. [Creating a JMS Physical Schema](#)

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

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 *Administering 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.

Creating a JMS Physical Schema

Create a JMS physical schema using the standard procedure, as described in *Creating a Physical Schema* in *Administering 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* in *Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using JMS follows the standard procedure. See *Creating an Integration Project* of *Developing Integration Projects with 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

Creating and Defining a JMS Model

This section contains the following topics:

- [Create a JMS Model](#)
- [Defining the JMS Datastores](#)



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.

Create a JMS Model

Create a JMS Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with 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.

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 [Files](#) for more information about this reverse-engineering method.
- Duplicate a datastore from another model into the JMS model.



Note:

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 attributes 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 attributes 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 mappings the `JMSPriority` default property on messages consumed from or pushed to a JMS queue called `CUSTOMER`, you would add an attribute called `JMSPriority` (with this exact case) to the `CUSTOMER` datastore. This attribute 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 attributes making up the message payload in a DELIMITED file format. 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:

- [JMS Standard Properties](#)
- [Using JMS Properties](#)

Designing a Mapping

You can use JMS as a source or a target of a mapping. It cannot be used as the staging area.

The KM choice for a mapping or a check determines the abilities and performance of this mapping or check. The recommendations in this section help in the selection of the KM for different situations concerning JMS messages.

Loading Data from a JMS Source

JMS can be used as a source or a target in a mapping. 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 Mapping Flow tab to load data between JMS and another type of data server is essential for the performance of a mapping.

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 24-2](#) lists the JMS specific options.

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 24-2](#) lists the JMS specific KM options of this IKM.

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

JMS Knowledge Modules Options

[Table 24-2](#) 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 24-2 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.
JMS_COMMIT=1	Read/Write	Commit the JMS read operation, immediately after the driver is done with the reading of all available messages. JMS server considers the messages read as being consumed by some client.
JMS_COMMIT=0:	Read/Write	JMS driver reads messages, but the JMS 'read' is not considered 'done' by the JMS server. This happens when the corresponding ODI session (not necessarily just the interface) finishes successfully. If the session fails, the messages are NOT consumed.
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 mappings it is advised to set this parameter to zero.
JMSPRIORITY	Write	Relative Priority of the message: 0 (lowest) to 9 (highest).

Table 24-2 (Cont.) JMS Specific KM Options

Option	Used to	Description
SENDMESSAGE TYPE	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.
MESSAGE MAXNUMBER	Read	Maximum number of messages retrieved [0 .. 4 000 000 000]. 0: All messages are retrieved.
MESSAGE TIMEOUT	Read	Time to wait for the first message in milliseconds [0 .. 4 000 000 000]. if MESSAGE TIMEOUT is equal to 0, then there is no timeout. MESSAGE TIMEOUT and MESSAGE MAXNUMBER cannot be both equal to zero. if MESSAGE TIMEOUT= 0 and MESSAGE MAXNUMBER =0, then MESSAGE TIMEOUT takes the value 1. Warning! A mapping may retrieve no message if this timeout value is too small.
NEXTMESSAGE TIMEOUT	Read	Time to wait for each subsequent message in milliseconds [0 .. 4 000 000 000]. The default value is 1000. Warning! A mapping may retrieve only part of the messages available in the topic or the queue if this value is too small.
MESSAGE SELECTOR	Read	Message selector in ISO SQL syntax. See Using JMS Properties for more information on message selectors.

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 24-3](#).

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 mappings.

For more details on using these properties in a Java program, see <http://java.sun.com/products/jms/>.

Table 24-3 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.
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 24-4](#) lists the optional JMS-defined properties in the JMS standard.

Table 24-4 Optional JMS Properties of Message Headers

Property	JMS Type	Access (Read/Write)	Description
JMSXUserID	JMS String	R	Client User ID.

Table 24-4 (Cont.) Optional JMS Properties of Message Headers

Property	JMS Type	Access (Read/Write)	Description
JMSXAppID	JMS String	R	Client Application ID.
JMSXProducerTXID	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.
JMSXConsumerTXID	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

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.

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 mappings. See [Declaring JMS Properties as Pseudo-Columns](#) for more information.

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.`

 **Note:**

- 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'
```

Filtering on the Client

Filtering is performed after receiving the messages, and is setup by creating a standard Oracle Data Integrator mapping 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.

Using Property Values as Source Data

It is possible to use the values of JMS properties as source data in a mapping. 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.

Setting Properties when Sending a Message

When sending messages it is possible to specify JMS properties by mapping values of the pseudo-columns in a mapping targeting a JMS datastore. Certain properties may be set using KM options. See [JMS Standard Properties](#) for more information.

25

JMS XML

It is important to understand how to work with Java Message Services (JMS) with a XML payload in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a JMS XML Model](#)
- [Designing a Mapping](#)

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 [JMS](#).

Concepts

The JMS XML Knowledge Modules apply to most popular JMS compliant middleware, including Oracle Service Bus, Sonic MQ, and so forth. Most of these Knowledge Modules include transaction handling to ensure message delivery.

JMS Message Structure

See [JMS Message Structure](#) for information about the JMS message structure.

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 [XML Files](#) and [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 attributes in a mapping. 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 mapping accessing the message.
2. *Extract the data:* A sequence of mappings 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 mapping 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 mapping loading the structure.
2. *Load the data:* A sequence of mappings 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 mapping 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 mapping using data extracted from the XML schema. These filters are processed in Data Integrator, after the message is synchronized to the XML schema.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 25-1](#) for handling XML messages.

Table 25-1 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.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

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.

 **Note:**

By default, a sequence of four ';' is used as fixed record separator for JMS XML driver read operations. If the XML data also contains a sequence of four or more ';' characters, an error will occur and you must set the record separator to a different value. This is achieved using the `Oracle.odi.jmsxmlColSepString` JVM option. For example, `Oracle.odi.jmsxmlColSepString="????"` will set the JMS XML driver record separator to "????" instead of ";;;;".

This option must be set in the following locations:

- In Studio, this parameter is set in the `odi.conf` parameter file. Add a new `AddVMOption` entry.
- For 12c Standalone/Colocated Agents, use `ODI_INSTANCE_OPTIONS` in the `instance.sh` script.
- For 11g Standalone Agents, use `ODI_ADDITIONAL_JAVA_OPTIONS` in the `odiparams` file.
- For JEE Agents, add it to `JAVA_OPTIONS` in the `startManagedWeblogic` script.

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.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating a JMS XML Data Server](#)
2. [Creating a JMS XML Physical Schema](#)

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.

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 Administering 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 [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 in [Table 25-2](#).
 - 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 25-2 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_ROW SEPARATOR	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.
transform_nonascii or transform_nonascii	boolean (true false)	Transform Non Ascii. Set to false to keep non-ascii characters. Default is true. This parameter is not mandatory.

Example 25-1 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>`

Creating a JMS XML Physical Schema

Create a JMS XML physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering 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* in *Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a project using JMS XML follows the standard procedure. See *Creating an Integration Project* of *Developing Integration Projects with 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

Creating and Reverse-Engineering a JMS XML Model

This section contains the following topics:

- [Create a JMS XML Model](#)
- [Reverse-Engineering a JMS XML Model](#)

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 *Developing Integration Projects with 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.

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 *Developing Integration Projects with Oracle Data Integrator*.

Oracle Data Integrator will automatically add the following attributes to the datastores generated from the XML data:

- Primary keys (PK attributes) for parent-child relationships
- Foreign keys (FK attributes) for parent-child relationships
- Order identifier (ORDER attributes) to enable the retrieval of the order in which the data appear in the XML file.

These extra attributes enable the hierarchical XML structure's mapping in a relational structure stored in the schema. See [Oracle Data Integrator Driver for XML Reference](#) for more information.

Designing a Mapping

The KM choice for a mapping or a check determines the abilities and performance of this mapping or check. The recommendations in this section help in the selection of the KM for different situations concerning XML messages.

Loading Data from a JMS XML Source

JMS XML can be used as a source or a target in a mapping. 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 Mapping 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 LKM. 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 mapping 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 mapping should commit the message consumption by setting the `JMS_COMMIT` to `Yes`.

Table 25-3 lists the JMS specific options of this knowledge module.

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 mapping 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 mappings should load the datastores in the hierarchy order, starting by the top of the hierarchy and going down. The mappings loading subsequent levels of the XML schema hierarchy should load the foreign key attribute 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 attribute 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 mapping 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` attribute 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 attribute 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 [Oracle Data Integrator Driver for XML Reference](#).

[Table 25-3](#) 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 25-3](#) 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 25-3 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 mapping 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 mappings 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 Using JMS Properties for more information on message selectors.

Table 25-3 (Cont.) JMS Specific KM Options

Option	Used to	Description
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! A mapping 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! A mapping 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. Option applicable only to first mapping loading the schema (INITIALIZE_XML_SCHEMA=true). IKM inserts a record for the root element of the XML schema, if ROOT_TABLE<>" and INITIALIZE_XML_SCHEMA=true. Warning! Use only, if no mapping will populate the root table of the XML structure. Otherwise a duplicate root element will be encountered.
SENDMESSAGE TYPE	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 mapping that writes to the schema XML.

LDAP Directories

It is important to understand how to work with LDAP directories in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering an LDAP Directory](#)
- [Designing a Mapping](#)
- [Troubleshooting](#)

Introduction

Oracle Data Integrator supports LDAP directories integration using the Oracle Data Integrator Driver for LDAP.

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.

 **Note:**

ODI LDAP driver's support for LDAP servers is limited. All the features of the driver may not work on any given instance of an LDAP server. ODI uses Java JNDI API to interact with the LDAP servers. If the LDAP server adheres exactly with LDAP specifications, then driver features will work. Otherwise, some of the features may not work.

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 [Oracle Data Integrator Driver for LDAP Reference](#) for more information on this driver.

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 [Generic SQL](#) or the technology chapters for more information on these KMs.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technologic Specific Requirements

There are no technology-specific requirements for using LDAP directories in Oracle Data Integrator.

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 [Oracle Data Integrator Driver for XML Reference](#) for more information on these concepts and configuration parameters.

Setting up the Topology

Setting up the topology consists in:

1. [Creating an LDAP Data Server](#)
2. [Creating a Physical Schema for LDAP](#)

Creating an LDAP Data Server

An LDAP data server corresponds to an LDAP tree that is accessible to Oracle Data Integrator.

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 *Administering 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 26-1](#). See [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 26-1 JDBC URL Properties

Property	Value	Notes
ldap_auth	<authentication mode>	LDAP Directory authentication method. See the auth property in Table A-1

Table 26-1 (Cont.) JDBC URL Properties

Property	Value	Notes
ldap_url	<LDAP URL>	LDAP Directory URL. The URL must not contain spaces. If there are spaces in the URL, replace them with %20. See the url property in Table A-1
ldap_user	<LDAP user name>	LDAP Directory user name. See the user property in Table A-1
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-1
ldap_basedn	<base DN>	LDAP Directory basedn. The basedn must not contain spaces. If there are spaces in the basedn, replace them with %20. See the basedn property in Table A-1

Example 26-1 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
```

Creating a Physical Schema for LDAP

Create an LDAP physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator* and associate it in a given context.

Setting Up an Integration Project

Setting up a Project using the LDAP database follows the standard procedure. See *Creating an Integration Project of Developing Integration Projects with 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

Creating and Reverse-Engineering an LDAP Directory

This section contains the following topics:

- [Create an LDAP Model](#)

- [Reverse-Engineering an LDAP Model](#)

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 [LDAP to Relational Mapping](#) for more information.

Create an LDAP Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

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 mappings.

The solution consists in creating an alias file that contains a list of short and clear table name aliases. See [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 Developing Integration Projects with 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 [LDAP Processing Overview](#) for more information.

Designing a Mapping

You can use LDAP entries as a source or a target of a mapping.

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

Loading Data from and to LDAP

An LDAP directory can be used as a mapping's source or target. The LKM choice in the Loading Knowledge Module tab that is used to load data between LDAP entries and other types of data servers is essential for the performance of the mapping.

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 26-2](#) lists some examples of KMs that you can use to load from an LDAP source to a staging area.

Table 26-2 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

Loading Data to an LDAP Directory

It is not advised to use an LDAP directory as a staging area.

Integrating Data in an LDAP Directory

LDAP can be used as a target of a mapping. The IKM choice in the Integration Knowledge Module 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 26-3](#) lists some examples of KMs that you can use to integrate data from a staging area to an LDAP target.

Table 26-3 KMs to Integrate Data in an LDAP Directory

Mode	KM	Notes
Append	IKM SQL to SQL Append	Generic KM

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 [Installation and Configuration](#) .

Oracle TimesTen In-Memory Database

It is important to understand how to work with Oracle TimesTen In-Memory Database in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Setting Up an Integration Project](#)
- [Creating and Reverse-Engineering a TimesTen Model](#)
- [Setting up Data Quality](#)
- [Designing a Mapping](#)

Introduction

The *Oracle TimesTen In-Memory Database* (TimesTen) provides real-time data management. It provides application-tier database and transaction management built on a memory-optimized architecture accessed through industry-standard interfaces. Optional data replication and Oracle caching extend the product to enable multi-node and multi-tier configurations that exploit the full performance potential of today's networked, memory-rich computing platforms.

Oracle TimesTen In-Memory Database is a memory-optimized relational database. Deployed in the application tier, TimesTen operates on databases that fit entirely in physical memory using standard SQL interfaces. High availability for the in-memory database is provided through real-time transactional replication.

TimesTen supports a variety of programming interfaces, including JDBC (Java Database Connectivity) and PL/SQL (Oracle procedural language extension for SQL).

Concepts

The TimesTen concepts map the Oracle Data Integrator concepts as follows: An Oracle TimesTen In-Memory Database instance corresponds to a data server in Oracle Data Integrator. Within this database instance, the 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 an Oracle TimesTen In-Memory Database ODBC DSN.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 27-1](#) for handling TimesTen data. These KMs use TimesTen specific features. It is also possible to use the generic SQL KMs with the TimesTen database. See [Generic SQL](#) for more information.

Table 27-1 TimesTen KMs

Knowledge Module	Description
IKM TimesTen Incremental Update (MERGE)	Integrates data from staging area into a TimesTen target table using TimesTen JDBC driver in incremental update mode. For example, inexistent rows are inserted; already existing rows are updated.
LKM SQL to TimesTen	Loads data from an ANSI SQL-92 source to a TimesTen staging table using the TimesTen JDBC driver.
LKM File to TimesTen (ttBulkCp)	Loads data from a file to a TimesTen staging table using ttBulkCp utility.

Installation and Configuration

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

- [System Requirements and Certifications](#)
- [Technology Specific Requirements](#)
- [Connectivity Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>

Technology Specific Requirements

Some of the Knowledge Modules for TimesTen use the ttBulkCp utility.

The following requirements and restrictions apply for these Knowledge Modules:

- The host of the ODI Agent running the job must have the TimesTen Client utilities installed (TTBULKCP)
- Data transformations should be executed on the staging area or target
- The correct ODBC entry must be created on the agent machine:

- Client DSN: A Client DSN specifies a remote database and uses the TimesTen Client. A Client DSN refers to a TimesTen database indirectly by specifying a hostname, DSN pair, where the hostname represents the server machine on which TimesTen Server is running and the DSN refers to a Server DSN that specifies the TimesTen database on the server host.
- Server DSN: A Server DSN is always defined as a system DSN and is defined on the server system for each database on that server that will be accessed by client applications. The format and attributes of a server DSN are very similar to those of a Data Manager DSN.

Connectivity Requirements

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

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

- [Install the TimesTen ODBC Driver](#)
- [Declare a TimesTen ODBC Data Source](#)
- [JDBC Driver](#)
- [ODI Agent](#)

Install the TimesTen ODBC Driver

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

Declare a TimesTen 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.

JDBC Driver

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

ODI Agent

The ODI Agent running the job must have the TimesTen JDBC Driver and ODBC driver installed and configured.

Setting up the Topology

Setting up the Topology consists of:

1. [Creating a TimesTen Data Server](#)
2. [Creating a TimesTen Physical Schema](#)

Creating a TimesTen Data Server

A TimesTen data server corresponds to a TimesTen database.

Creation of the Data Server

Create a data server for the TimesTen technology using the standard procedure, as described in *Creating a Data Server of Administering Oracle Data Integrator*. This section details only the fields required or specific for defining a TimesTen 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:** TimesTen user with its password
2. In the JDBC tab:
 - **JDBC Driver:** `org.TimesTen.Driver`
 - **JDBC URL:** `jdbc:timesten:direct:dsn=<DSNname>`
where `DSNname` is the name of an ODBC datasource configured on the machine running the agent



Note:

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

Creating a TimesTen Physical Schema

Create a TimesTen physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator* and associate it in a given context.

Creating and Reverse-Engineering a TimesTen Model

This section contains the following topics:

- [Create a TimesTen Model](#)
- [Reverse-engineer a TimesTen Model](#)

Create a TimesTen Model

Create a TimesTen Model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Reverse-engineer a TimesTen Model

TimesTen 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 TimesTen JDBC driver.

Standard Reverse-Engineering

To perform a Standard Reverse-Engineering on TimesTen use the usual procedure, as described in Reverse-engineering a Model of *Developing Integration Projects with Oracle Data Integrator*.

Customized Reverse-Engineering

To perform a Customized Reverse-Engineering on TimesTen with a RKM, use the usual procedure, as described in Reverse-engineering a Model of *Developing Integration Projects with Oracle Data Integrator*. This section details only the fields specific to the TimesTen technology:

1. In the Reverse Engineer tab of the TimesTen Model, select the KM: `RKM SQL (Jython).<project name>`.

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

Setting up Data Quality

Oracle Data Integrator provides the CKM SQL for checking data integrity against constraints defined on a TimesTen table. See Flow Control and Static Control in *Developing Integration Projects with Oracle Data Integrator* for details.

See [Generic SQL](#) for more information.

Designing a Mapping

You can use TimesTen as a source, staging area, or a target of a mapping.

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

Loading Data from and to TimesTen

TimesTen can be used as a source, target or staging area of a mapping. The LKM choice in the Loading Knowledge Module tab to load data between TimesTen and another type of data server is essential for the performance of a mapping.

Loading Data from TimesTen

Use the [Generic SQL](#) KMs or the KMs specific to the other technology involved to load data from a TimesTen database to a target or staging area database.

For extracting data from a TimesTen staging area to a TimesTen table, use the IKM TimesTen Incremental Update (MERGE). See [Loading Data from TimesTen](#) for more information.

Loading Data to TimesTen

Oracle Data Integrator provides Knowledge Modules that implement optimized methods for loading data from a source or staging area into a TimesTen database. These optimized TimesTen KMs are listed in [Table 27-2](#). In addition to these KMs, you can also use the [Generic SQL](#) KMs or the KMs specific to the other technology involved.

Table 27-2 KMs for loading data to TimesTen

Source or Staging Area Technology	KM	Notes
SQL	LKM SQL to TimesTen	Loads data from an ANSI SQL-92 source to a TimesTen staging table using the TimesTen JDBC driver.
File	LKM File to TimesTen (ttBulkCp)	Loads data from a file to a TimesTen staging table using ttBulkCp utility.

Integrating Data in TimesTen

Oracle Data Integrator provides Knowledge Modules that implement optimized data integration strategies for TimesTen. These optimized TimesTen KMs are listed in [Table 27-3](#). In addition to these KMs, you can also use the [Generic SQL](#) KMs.

The IKM choice in the Integration Knowledge Module tab determines the performances and possibilities for integrating.

Table 27-3 KMs for integrating data to TimesTen

KM	Notes
IKM TimesTen Incremental Update (MERGE)	Integrates data from staging area into a TimesTen target table using TimesTen JDBC driver in incremental update mode. For example, inexistent rows are inserted; already existing rows are updated.

Setting Up an Integration Project

Setting up a project using the TimesTen database follows the standard procedure. See *Creating an Integration Project* of *Developing Integration Projects with Oracle Data Integrator*.

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

- CKM SQL
- IKM SQL Control Append
- IKM TimesTen Incremental Update (MERGE)
- LKM SQL to TimesTen

- LKM File to TimesTen (ttBulkCp)
- RKM SQL (Jython)

Oracle GoldenGate

You can work with Oracle GoldenGate to capture changes on source transactional systems and replicate them in a staging server for consumption by Oracle Data Integrator mappings.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Working with the Oracle GoldenGate JKMs](#)
- [Advanced Configuration](#)
- [Integrated Capture](#)
- [Using Different Capture and Apply Modes Together](#)
- [Switching to Different Process Mode](#)
- [Upgrading GoldenGate Classic Extract to Integrated](#)

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:

<http://www.oracle.com/technetwork/middleware/goldengate/overview/index.html>

Overview of the GoldenGate CDC Process

Oracle Data Integrator can capture changes in a source database using Oracle GoldenGate to process them in the ODI CDC framework. Oracle Data Integrator uses Oracle GoldenGate to replicate 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 can be stored in an Oracle or Teradata 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 the source tables using either the JKM <database> to Oracle Consistent (OGG Online) or the JKM <database> to Teradata Consistent (OGG Online).
3. The journals are started in either online mode or offline mode.

- **Online mode:** Starting the journals in online mode configures and starts the GoldenGate Capture (Extract) process to capture the changes in the source database and corresponding Delivery (Replicat) processes to replicate the changes in the staging database. Changes are replicated into both the replicated source table and the CDC infrastructure.

The GoldenGate Capture and Delivery processes are deployed and started using the GoldenGate JAgent interface. The GoldenGate JAgent facilitates communication between Oracle Data Integrator and Oracle GoldenGate.

- **Offline mode:** Starting the journals in offline mode creates the 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 this stage.

Using the generated configuration files, an Oracle GoldenGate Capture process is configured and started to capture changes from the source database, and corresponding Delivery 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.

Note:

The offline mode requires an Oracle GoldenGate data server to be first created in Topology. See [Define the Oracle GoldenGate Data Servers](#) for instructions on how to create one.

4. ODI mappings can source from the replicated tables and use captured changes seamlessly within any ODI scenario.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules listed in [Table 28-1](#) for replicating online data from a source to a staging database. Like any other CDC JKMs, the Oracle GoldenGate JKMs journalize data in the source server.

The JKM <database> to Oracle Consistent (OGG Online) and the JKM <database> to Teradata Consistent (OGG Online) perform the same tasks:

- Create and manage the ODI CDC framework infrastructure on the replicated tables.
- If the journals are started in online mode, configure and start the Oracle Capture and Delivery processes on the GoldenGate servers using the GoldenGate JAgent.
- If the journals are started in offline mode, generate the parameter files to set up the Oracle GoldenGate Capture and Delivery processes and the `Readme.txt` explaining how to complete the setup.

- Provide extra steps to check the configuration of the source database and proposes tips to correct the configuration.

Table 28-1 Oracle GoldenGate Knowledge Modules

Knowledge Module	Description
JKM Oracle to Oracle Consistent (OGG Online)	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 Online)	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 Online)	Creates the infrastructure for consistent set journalizing on an Oracle staging server and generates the Oracle GoldenGate configuration for replicating data from a Sybase ASE source to this staging server.
JKM MSSQL to Oracle Consistent (OGG Online)	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.
JKM Oracle to Teradata Consistent (OGG Online)	Creates the infrastructure for consistent set journalizing on a Teradata staging server and generates the Oracle GoldenGate configuration for replicating data from an Oracle source to this staging server.
JKM DB2 UDB to Teradata Consistent (OGG Online)	Creates the infrastructure for consistent set journalizing on a Teradata 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 Teradata Consistent (OGG Online)	Creates the infrastructure for consistent set journalizing on a Teradata staging server and generates the Oracle GoldenGate configuration for replicating data from a Sybase ASE source to this staging server.
JKM MSSQL to Teradata Consistent (OGG Online)	Creates the infrastructure for consistent set journalizing on a Teradata staging server and generates the Oracle GoldenGate configuration for replicating data from a Microsoft SQL Server source to this staging server.

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](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

See also the Oracle GoldenGate documentation on OTN for source and staging database version platform support.

Technology Specific Requirements

In order to run the *Capture* and *Delivery* 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 Capture or Delivery can be started, and must remain running during their execution for resource management.

In order to perform online journalizing, the Oracle GoldenGate JAgent process must be configured and running on the Oracle GoldenGate instances.

Oracle GoldenGate has specific requirement and installation instructions that must be performed before starting the Capture and Delivery processes configured with the Oracle GoldenGate JKMs. See the Oracle GoldenGate Documentation on OTN for more information.

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.

Working with the Oracle GoldenGate JKMs

To use the JKM <database> to Oracle Consistent (OGG Online) or the JKM <database> to Teradata Consistent (OGG Online) 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 Source Datastores](#)
5. [Configure and Start Oracle GoldenGate Processes \(Offline mode only\)](#)
6. [Design Mappings Using Replicated Data](#)

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 Source Data Server](#)
2. [Create the Source Physical Schema](#)
3. [Define the Staging Server](#)
4. [Create the Staging Physical Schema](#)
5. [Define the Oracle GoldenGate Data Servers](#)
6. [Create the Oracle GoldenGate Physical Schemas](#)
7. [Create the Oracle GoldenGate Logical Schemas](#)

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:

- [Creating an Oracle Data Server](#)
- [Creating a Microsoft SQL Server Data Server](#)
- [Creating a DB2/400 Data Server](#)

This data server represents the source database instance.

Create the Source Physical Schema

Create a physical schema under the data server that you have created in [Define the Source Data Server](#). Use the standard procedure, as described in *Creating a Physical Schema* in *Administering Oracle Data Integrator*.

Create for this physical schema a logical schema using the standard procedure, as described in *Creating a Logical Schema* in *Administering Oracle Data Integrator* and associate it in a given context.

Define the Staging Server

Create a data server for the Oracle or Teradata technology. For more information, see:

- [Creating an Oracle Data Server](#)
- [Creating a Teradata Data Server](#)

Create the Staging Physical Schema

Create an Oracle or Teradata physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering 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 in Administering Oracle Data Integrator* and associate it in a given context.

Define the Oracle GoldenGate Data Servers

An Oracle GoldenGate data server corresponds to the Oracle GoldenGate JAgent process in Oracle Data Integrator (ODI). The Oracle GoldenGate JAgent process facilitates communication between ODI and the Oracle GoldenGate servers. You must create a JAgent process for both the source and the target Oracle GoldenGate servers.

Create a data server for the Oracle GoldenGate technology using the standard procedure, as described in *Creating a Data Server of Administering Oracle Data Integrator*. This section details only the fields required or specific for defining an Oracle GoldenGate data server:

1. In the Definition tab:
 - **Name:** Name of the data server that will appear in the Oracle Data Integrator.
 - **Host:** Hostname or the IP address of the server where the JAgent process is running.
 - **JMX Port:** Port number of the JAgent process.
 - **Manager Port:** Port number of the Oracle GoldenGate manager instance.
 - **JMX User:** User name to connect to the JAgent.
 - **Password:** Password of the user credentials.
 - **Installation Path:** Location path for the Oracle GoldenGate installation. You must use this path when you create the capture process definitions from a model.

Create the Oracle GoldenGate Physical Schemas

The Oracle GoldenGate physical schemas in ODI correspond to the GoldenGate Capture and Delivery processes that perform CDC in Oracle GoldenGate. You must define the Oracle GoldenGate physical schemas to configure the Capture process on the source GoldenGate server and Delivery process on the target GoldenGate server.

Create a physical schema under the Oracle GoldenGate data server that you have created in [Define the Oracle GoldenGate Data Servers](#). Use the standard procedure, as described in Creating a Physical Schema in *Administering Oracle Data Integrator*. This section details only the fields required or specific to create the physical schemas to configure the Oracle GoldenGate Capture and Replicate processes.

 **Note:**

Alternatively, you can create the Oracle GoldenGate physical schemas from the model. See [Create Oracle GoldenGate Physical Schemas from the model](#) for information about how to create physical schemas from the model.

GoldenGate Capture Process Fields

Note that the GoldenGate Capture process must be configured on the source GoldenGate server.

1. In the Process Definition tab:
 - **Process Type:** Type of the process that you want to configure. Select Capture as the process type.
 - **Name:** Name of the process (physical schema) in Oracle Data Integrator. Process name cannot exceed 8 characters and only upper case is allowed.
 - **Trail File Path:** Location of the Oracle GoldenGate trail file. Only two characters for the file name part are allowed.
 - **Remote Trail File Path:** Location of the remote trail file. Only two characters for the file name part are allowed.
 - **Trail File Size:** Size of the Oracle GoldenGate trail file in Megabytes.
 - **Report Fetch:** Enables report information to include the fetching statistics.
 - **Report Count Frequency:** Reports the total operations count at specific intervals. If the interval is not specified the entry is not added to the parameter file.
 - **Select a parameter:** List of available Oracle GoldenGate parameters. Only the parameters for the supported database are listed. Select a parameter and click **Add**. A template of the selected parameter is added to the text box.

See the *Oracle GoldenGate Reference Guide* on OTN for information about the GoldenGate parameters.

Delivery Process Fields

Note that the GoldenGate Delivery process must be configured on the target GoldenGate server.

1. In the Process Definition tab:
 - **Process Type:** Type of the process that you want to configure. Select Delivery as the process type.
 - **Name:** Name of the process (physical schema) in Oracle Data Integrator. Process name cannot exceed 7 characters and only uppercase is allowed.

- **Trail File Path:** Location of the trail file. Only two characters for the filename part are allowed.
- **Discard File Path:** Location of the discard file.
- **Definition File Path:** Location of the definition file.
- **Report Detail:** Enables report information to include any collision counts.
- **Report Count Frequency:** Report the total operations count at specific intervals. If the interval is not specified the entry is not added to the parameter file.
- **Select a parameter:** List of available Oracle GoldenGate parameters. Only the parameters for the supported database are listed. Select a parameter and click **Add**.

See the *Oracle GoldenGate Reference Guide* on OTN for information about the GoldenGate parameters.

 **Note:**

In the definition of the logical schema, you must select the logical schema for the staging database.

Create the Oracle GoldenGate Logical Schemas

Create logical schemas for the GoldenGate physical schemas (GoldenGate Capture and Delivery processes) that you created in section [Create the Oracle GoldenGate Physical Schemas](#). You must create a logical schema for both the Capture process and the Delivery process.

To create logical schemas:

1. In the Topology Navigator expand the Technologies node in the Logical Architecture accordion.
2. Right-click Oracle GoldenGate and select New Logical Schema.
3. Fill in the Logical Schema Name.
4. Select the appropriate process type, either *Capture* or *Delivery*, to which you want to attach your logical schema.
5. For each Context in the left column, select an existing Physical Schema in the right column. This Physical Schema is automatically associated to the logical schema in this context. Repeat this operation for all necessary contexts.

 **Note:**

If the process type is set to 'Delivery', you must select the name of the logical schema that GoldenGate will use to deliver the changes. In this case, select a logical schema name for the 'Target DB Logical Schema'.

6. From File menu, click **Save**.

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 or Teradata technology. This model must use the logical schema created using the instructions in [Create the Staging Physical Schema](#).

See *Creating a Model in Developing Integration Projects with 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 specifying a capture or delivery process to perform the initial load or by setting the USE_OGG_FOR_INIT JKM option to `Yes` to create a process to perform the initial load when you [Configure CDC for the Source Datastores](#).

Note:

See *Creating Data Models with Common Format Designer in Developing Integration Projects with Oracle Data Integrator* for more information on diagrams, generating DDL, and generating Interface IN features.

Set Up an Integration Project

Setting up a project using Oracle GoldenGate features follows the standard procedure. See *Creating an Integration Project of Developing Integration Projects with Oracle Data Integrator*.

Depending on the technology of your source data server and staging server, import one of the following JKMs into your project:

- JKM Oracle to Oracle Consistent (OGG Online)
- JKM DB2 UDB to Oracle Consistent (OGG Online)
- JKM Sybase ASE to Oracle Consistent (OGG Online)
- JKM MSSQL to Oracle Consistent (OGG Online)
- JKM Oracle to Teradata Consistent (OGG Online)
- JKM DB2 UDB to Teradata Consistent (OGG Online)

- JKM Sybase ASE to Teradata Consistent (OGG Online)
- JKM MSSQL to Teradata Consistent (OGG Online)

Configure CDC for the Source Datastores

Changed Data Capture must be configured for the source datastores. 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 source datastore. In the Journalizing tab of the data model, set the Journalizing Mode to Consistent Set and select the appropriate JKM <database> to Oracle Consistent (OGG Online) or JKM <database> to Teradata Consistent (OGG Online).

Select the following GoldenGate processes (physical schemas) using the process selection drop-down list:

- Capture Process
- Delivery Process
- Initial Load Capture Process
- Initial Load Delivery Process

If you do not want to use an existing GoldenGate process, you can create new processes from here using the Create button next to the <Process Name> field. See [Create Oracle GoldenGate Physical Schemas from the model](#) for information about how to create GoldenGate processes from the model.

Set the KM options as follows:

- **ONLINE:** If you set this option to true, the JKM configures the CDC infrastructure and configures and starts the GoldenGate Capture and Delivery processes. If you set this option to false, the JKM generates the CDC infrastructure and the configuration files that are required to set up the GoldenGate Capture and Delivery processes. It also generates the `Readme.txt` that contains the instructions to configure and start the GoldenGate processes.

For more information about online and offline mode, see [Overview of the GoldenGate CDC Process](#).

For information about how to configure and start GoldenGate processes using the configuration files, see [Configure and Start Oracle GoldenGate Processes \(Offline mode only\)](#).

- **LOCAL_TEMP_DIR:** Full path to a temporary folder into which the Oracle GoldenGate configuration files will be generated
- **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 Online).

 **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
```

- **USE_OGG_FOR_INIT:** Applicable for offline mode only. 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`.
- **USE_INTEGRATED_REPLICAT_MODE:** This KM option is required when the delivery mode is classic or integrated replicat.

Values

True: Use integrated replicat mode

False: Use classic mode (default value)

Only the following KMs have this parameter implemented:

KM_JKM DB2 UDB to Oracle Consistent (OGG Online).xml

KM_JKM MSSQL to Oracle Consistent (OGG Online).xml

KM_JKM Oracle to Oracle Consistent (OGG Online).xml

KM_JKM Sybase ASE to Oracle Consistent (OGG Online).xml
KM_JKM DB2 UDB to Oracle Consistent (OGG Online).xml

KM_JKM MSSQL to Oracle Consistent (OGG Online).xml

KM_JKM Oracle to Oracle Consistent (OGG Online).xml

KM_JKM Sybase ASE to Oracle Consistent (OGG Online).xml

2. Select the datastores that you want to replicate or the model if want to replicate all datastores, 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**. If journals are started in online mode (ONLINE option for the JKM is set to true), the JKM creates the CDC infrastructure and configures and starts the Oracle GoldenGate processes. If journals are started in offline mode (ONLINE option for the JKM is set to false), the JKM creates the CDC infrastructure and generates the configuration files that are required to configure the Oracle GoldenGate processes. It also generates `Readme.txt` that contains the instructions to configure and start the GoldenGate processes.

For information about how to configure and start GoldenGate processes, see [Configure and Start Oracle GoldenGate Processes \(Offline mode only\)](#).

You can review the result of the journal startup action:

- If journals are started in online mode, the Oracle GoldenGate processes are configured and started. The changed data in the source datastores is captured and replicated in the staging tables.

- If the journals are started in offline mode, 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 \(Offline mode only\)](#).
- 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 Using Journalizing in *Developing Integration Projects with 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 Tables tab of the model after adding them to CDC.

Create Oracle GoldenGate Physical Schemas from the model

You can create the Oracle GoldenGate physical schemas for the following GoldenGate processes from the Journalizing tab of the Model Editor.

- Capture Process
- Delivery Process
- Initial Capture Process (Capture process to be used for initial load)
- Initial Delivery Process (Delivery process to be used for initial load)

When you create the Oracle GoldenGate physical schemas from the models, the default values are derived from the JAgent and the Model details.

To create the Oracle GoldenGate physical schemas from the model:

1. In the Designer Navigator expand the Models panel.
2. Expand the Models folder that contains the model from which you want to create the physical schemas.
3. Right-click the Model and select **Open**.
4. Click the Journalizing tab of the Model Editor.
5. Click **Create** button next to the Capture Process field.
6. Select the appropriate JAgent and Context.
7. Fill in the Process Name and Logical Process Name.
8. Click **OK** to create and select the Capture process.

 **WARNING:**

The physical schema generated for the Capture process needs to be changed manually. The Remote Trail File Path property of the physical schema uses the path for the Capture instance and needs to be changed to use the path for the Delivery instance.

9. Click **Create** button next to the Delivery Process field.
10. Select the appropriate JAgent and Context.
11. Fill in the Process Name and Logical Process Name.
12. Select the Target Database Logical Schema for the Delivery process.
13. Click **OK**.
14. Similarly, click **Create** buttons next to the Initial Load Capture Process and Initial Load Delivery Process fields to create physical schemas for them.

Configure and Start Oracle GoldenGate Processes (Offline mode only)

 **Note:**

- This section is applicable only if the journals are started in offline mode. That means only if the `ONLINE` option for the JKM is set to `false`.
- Connection to a JAgent is not required to configure Oracle GoldenGate Processes in offline mode. However, the necessary information must be available in Topology.

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 Capture 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 Delivery processes, using the `ggsci` command line.

See the Oracle GoldenGate documentation on OTN for more information on OBEY files, the `ggsci` and `defgen` utilities.

Design Mappings Using Replicated Data

You can use the data in the replicated data as a source in your mappings. This process is similar to using a source datastore journalized in consistent set mode. See *Using Changed Data: Consistent Set Journalizing in Developing Integration Projects with Oracle Data Integrator* for more information.

Advanced Configuration

This section includes the following advanced configuration topics:

- [Initial Load Method](#)
- [Tuning Replication Performances](#)
- [One Source Multiple Staging Configuration \(Offline mode only\)](#)

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 mappings to transfer the data.
- *Using database backup/restore tools* to copy data and structures.

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 Delivery 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.

One Source Multiple Staging Configuration (Offline mode only)

Note that one source multiple staging configuration can be done only in the offline journalizing mode.

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. Edit the source model and ensure that the logical schema for STG1 is selected.
2. Start the journals in offline mode and follow the instructions in the readme to set up the Oracle GoldenGate processes in SRC and STG1.
3. Edit the source model again, and select the logical schema for STG2.
4. Start the journals in offline mode and follow the instructions in the readme to set up the Oracle GoldenGate process in SRC and STG2.

Note:

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.

Integrated Capture

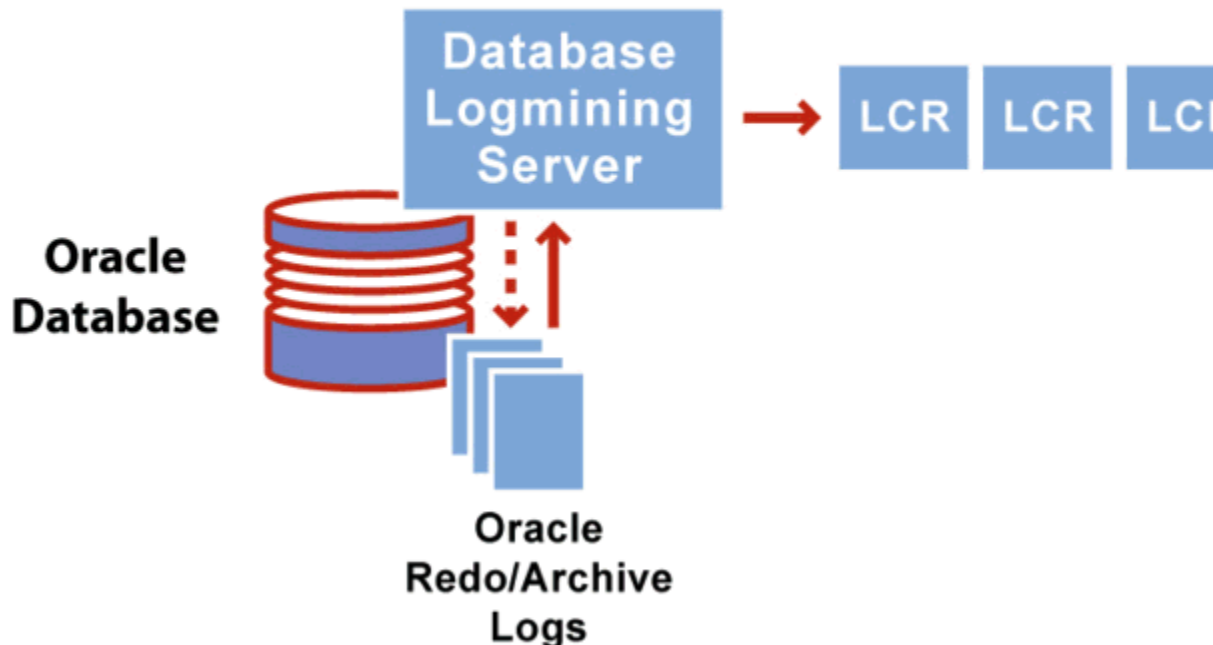
In the Integrated Capture mode, the Oracle GoldenGate extract process interacts directly with a database logmining server, to receive data changes in the form of logical change records (LCR).

The following are the benefits of Integrated Capture:

As the Integrated Capture uses the database logmining server to access the Oracle redo stream, you can automatically switch between different copies of archive logs or different mirrored versions of the online logs.

- Being fully integrated with the database, no additional steps are required to work with Oracle RAC, ASM, and TDE
- Enables faster filtering of tables
- Handles point-in-time recovery and RAC integration more efficiently
- Enables integrated log management, as the Oracle Recovery Manager (RMAN) automatically retains the archive logs required for the extract

- Supports capture from a multi-tenant container database
- As the Integrated Capture and the Integrated Apply are both database objects, the objects naming follows the same rules as other Oracle database objects
- For a release 11.2.0.4 source database and later (with source compatibility set to 11.2.0.4 or higher), the capture of DDL is performed by the logmining server asynchronously and requires no special triggers, tables, or other data objects installation
- DDL trigger and supporting objects are required when extract is in Integrated mode with a Oracle 11g source database earlier than version 11.2.0.4
- Oracle GoldenGate upgrades can be performed without stopping the user applications
- **Figure 28-1 Configuration of Extract in Integrated Capture**



Integrated Capture Deployment Options

Depending on where the mining database is deployed, you have two deployment options for integrated capture. The mining database is the one where the logmining server is deployed.

Local Deployment

For local deployment, the source database and the mining database are the same. The source database is the database:

- For which you want to mine the redo stream to capture changes.
- Where you deploy the logmining server.

As Integrated Capture is fully integrated with the database, this mode does not require any special database setup.

Downstream Deployment

In downstream deployment, the source and mining databases are different databases. When using a downstream mining configuration, the source database and mining database must be of the same platform. For example, if the source database is running on Windows 64-bit, the downstream database must also be on a Windows 64-bit platform.

1. Create the logmining server at the downstream database.
2. Configure redo transport at the source database to ship the redo logs to the downstream mining database for capture at that location.

Note:

Using a downstream mining server for capture is recommended to offload the capture overhead, and any other overhead from transformation or other processing from the production server, but requires log shipping and other configuration.

Deciding Which Apply Method to Use

The Replicat process enables the application of replicated data to an Oracle target database. For more information about Oracle GoldenGate processes, see *Administering Oracle GoldenGate for Windows and UNIX*.

For an Oracle target database, you can run Replicat in either nonintegrated or integrated mode. The following section explains these modes and the database versions that each mode supports:

Nonintegrated Replicat

In nonintegrated mode, the Replicat process uses standard SQL to apply data directly to the target tables.

You can apply transactions in parallel with a nonintegrated Replicat, by using a coordinated Replicat configuration. For more information, see *Administering Oracle GoldenGate for Windows and UNIX*.

Use nonintegrated Replicat when:

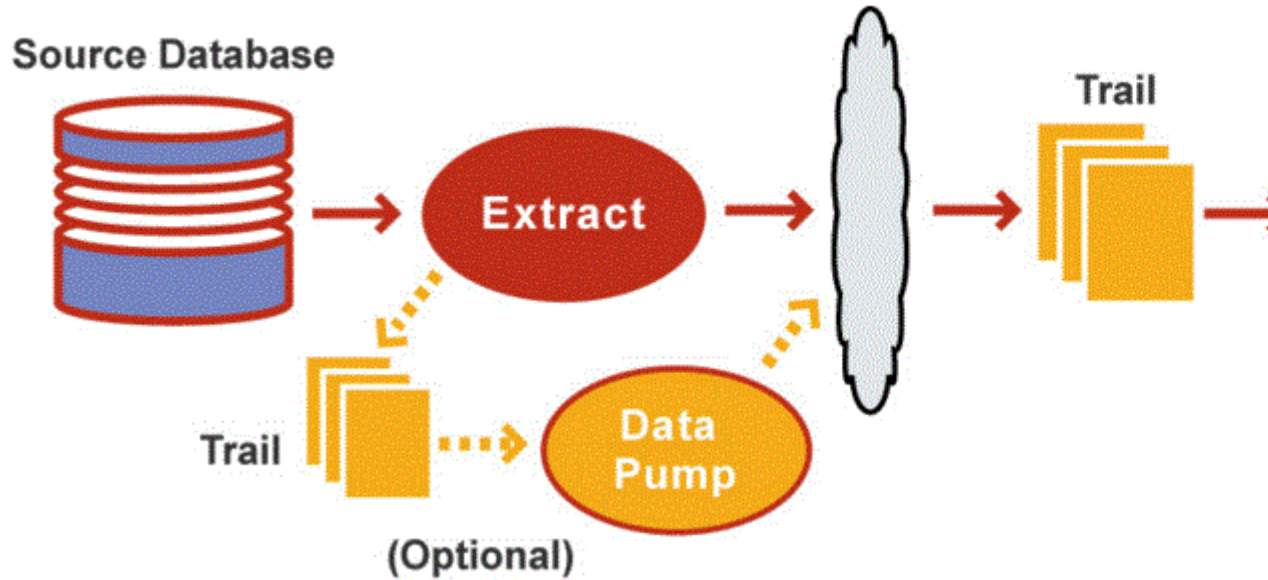
- The target Oracle database is a version earlier than Oracle 11.2.0.4.
- You want to extensively use features that are not supported in integrated Replicat mode.

In nonintegrated mode, Replicat operates as follows:

1. Reads the Oracle GoldenGate trail.
2. Performs data filtering, mapping, and conversion.
3. Constructs SQL statements that represent source database DML or DDL transactions (in committed order).

4. Applies the SQL to the target through Oracle Call Interface (OCI).

Figure 28-2 Nonintegrated Replicat Configuration

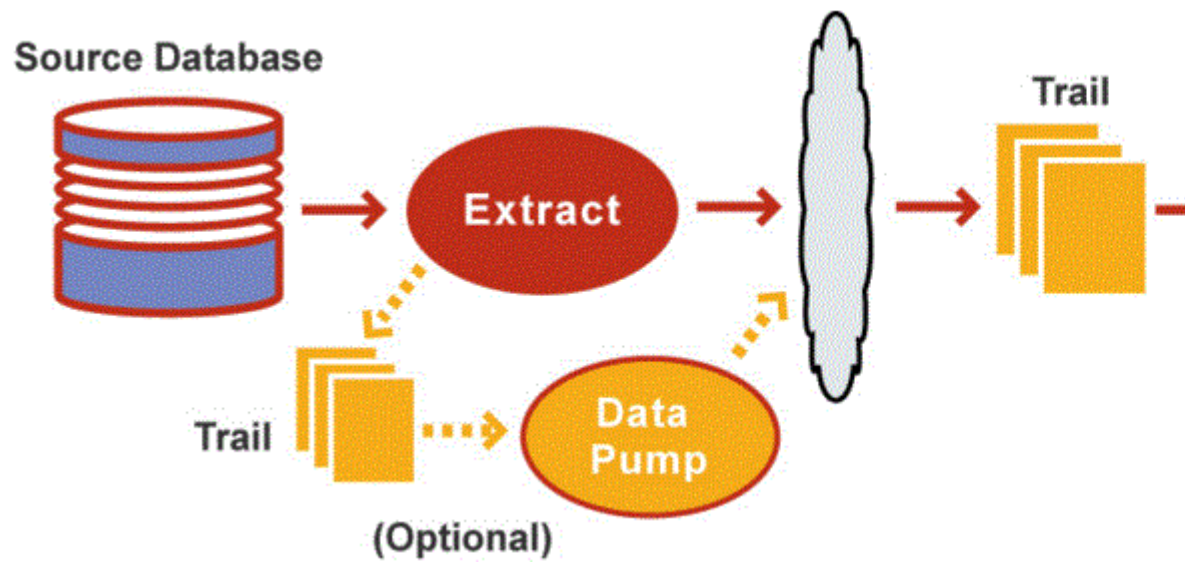


Integrated Replicat

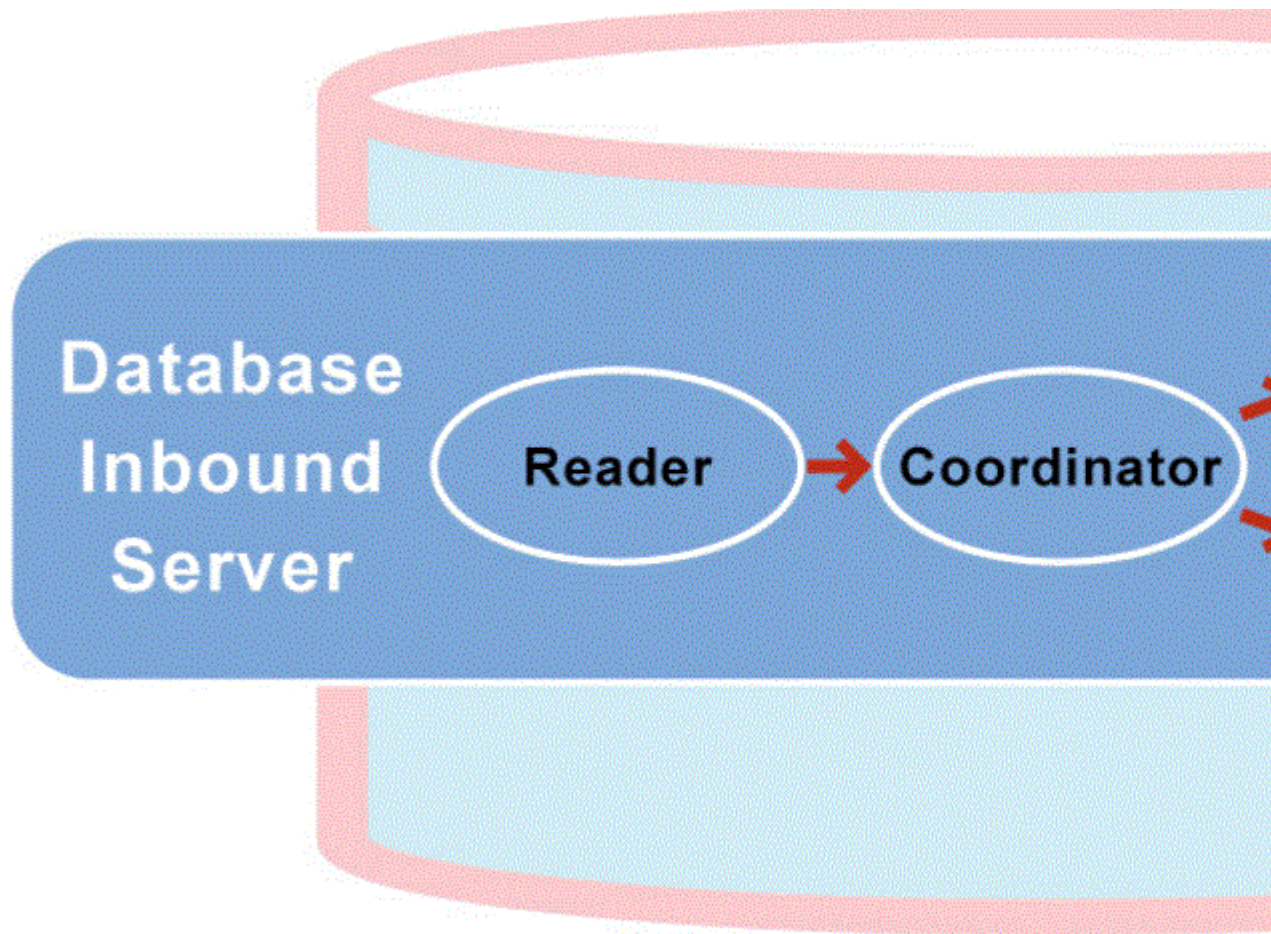
In integrated mode, the Replicat process leverages the apply processing functionality that is available within the Oracle database. In this mode, Replicat operates as follows:

- Reads the Oracle GoldenGate trail.
- Performs data filtering, mapping, and conversion.
- Constructs logical change records (LCR) that represent source database DML transactions (in committed order). DDL is applied directly by Replicat.
- Attaches to a background process in the target database known as a database inbound server by means of a lightweight streaming interface.
- Transmits the LCRs to the inbound server, which applies the data to the target database.

Figure 28-3 Integrated Replicat Configuration



Within a single Replicat configuration, multiple inbound server child processes known as apply servers apply transactions in parallel, while preserving the original transaction atomicity. You can increase this parallelism as much as your target system will support, when you configure the Replicat process or dynamically as needed.

Figure 28-4 Integrated Replicat with Two Parallel Apply Servers

Integrated Replicat applies transactions asynchronously. Transactions that do not have interdependencies can be safely executed and committed out of order to achieve fast throughput. Transactions with dependencies are guaranteed to be applied in the same order as on the source.

A reader process in the inbound server computes the dependencies among the transactions in the workload based on the constraints defined at the target database (primary key, unique, foreign key). Barrier transactions and DDL operations are managed automatically, as well. A coordinator process coordinates multiple transactions and maintains order among the apply servers.

If the inbound server does not support a configured feature or column type, Replicat disengages from the inbound server, waits for the inbound server to complete transactions in its queue, and then applies the transaction to the database in direct apply mode through OCI. Replicat resumes processing in integrated mode after applying the direct transaction.

The following features are applied in direct mode by Replicat:

- DDL operations
- Sequence operations
- SQLEXEC parameter within a TABLE or MAP parameter

- EVENTACTIONS processing
- UDT Note, if the extract uses USENATIVEOBJSUPPORT to capture the UDT, then integrated Replicat will apply it with the inbound server, otherwise it will be handled by Replicat directly.

 **Note:**

Because transactions are applied serially in direct apply mode, heavy use of such operations may reduce the performance of the integrated Replicat mode. Integrated Replicat performs best when most of the apply processing can be performed in integrated mode.

User exits are executed in integrated mode. The user exit may produce unexpected results, if the exit code depends on data in the replication stream.

Integrated Replicat Requirements

To use integrated Replicat, the following must be true:

- The target Oracle database must be Oracle 11.2.0.4 or later.
- Supplemental logging must be enabled on the source database to support the computation of dependencies among tables and scheduling of concurrent transactions on the target.
- Supplemental logging can be enabled at any time up to, but before, you start the Oracle GoldenGate processes.

Using Different Capture and Apply Modes Together

You can use the following capture and apply modes together:

- Classic capture (Oracle or non-Oracle source) and nonintegrated Replicat
- Classic capture (Oracle or non-Oracle source) and integrated Replicat
- Integrated capture and nonintegrated Replicat
- Integrated capture and integrated Replicat

You can use integrated capture and classic capture concurrently within the same source Oracle GoldenGate instance, and you can use integrated Replicat and nonintegrated Replicat concurrently within the same target Oracle GoldenGate instance.

This configuration requires careful placement of your objects within the appropriate process group, because there is no coordination of DDL or DML between classic and integrated capture modes, nor between nonintegrated and integrated Replicat modes. Each Extract group must process objects that are suited to the processing mode, based on table data types and attributes. No objects in one Extract can have DML or DDL dependencies on objects in the other Extract. The same type of segregation must be applied to the Replicat configuration.

The recommended Oracle GoldenGate configuration, when supported by the Oracle version, is to use one integrated capture on an Oracle source and one integrated Replicat per source database on an Oracle target. Integrated capture supports certain data types more completely than classic capture. One integrated Replicat configuration supports all Oracle data types either through the inbound server or by switching to direct apply when necessary, and it preserves source transaction integrity. You can adjust the parallelism settings to the desired apply performance level as needed.

If the target database is an Oracle version that does not support integrated Replicat, or if it is a non-Oracle database, you can use a coordinated Replicat configuration. For more information, see *Administering Oracle GoldenGate for Windows and UNIX*.

Switching to Different Process Mode

You can switch between the process modes. For example, you can switch from classic capture to integrated capture, or from integrated capture to classic capture. For instructions, see *Administering Oracle GoldenGate for Windows and UNIX*.

Upgrading GoldenGate Classic Extract to Integrated

To run integrated extract in GoldenGate 11.2.1, the following requirements should be met:

- Oracle RDBMS must be 11.2.0.3 or higher
- RDBMS (Database) patches must be applied:
 - 11.2.0.3 Database specific bundle patch for Integrated Extract 11.2.x
 - Redo compatibility should be set to 11.2.0.3, matching the DB version

The following section explains the upgrade procedure:

1. If you are using RAC environments and OGG versions 11.2.1.0.23+, execute the steps a to d. If you are using OGG version prior to 11.2.1.0.23, skip these steps and proceed with step 2.
2.
 - a. For a running extract, issue the following command:
 - b. `SEND extract <extract name> tranlogoptions prepareforupgradetoie`
 - c. For a stopped extract, start it after adding the following line to the parameter file:
 - d. `TRANLOGOPTIONS PREPAREFORUPGRADETOIE`
 - e. Monitor the ggserr.log file or corresponding extract report file for an INFO GG-01873 message, indicating that the change has taken affect, and that you can proceed with the upgrade.

 **Note:**

For the INFO message to be displayed, extract has to process a committed transaction on all the RAC nodes for a table being captured. As an alternative, a dummy table can be added to the extract parameter file, and doing DML on this table from all the threads will give extract commit boundary current checkpoints for all the threads.

Example from report file:

```
2014-06-05 17:06:09 INFO OGG-01873 The parameter TRANLOGOPTIONS
PREPAREFORUPGRADETOIE has taken effect. Proceed to the next step in the
upgrade process.
```

Example from ggserr.log file:

```
2014-06-05 17:06:09 INFO OGG-01873 Oracle GoldenGate Capture for
Oracle, src.prm: The parameter TRANLOGOPTIONS PREPAREFORUPGRADETOIE has
taken effect.
```

- f. Once the message appears, stop the extract, perform dblogin, and alter for conversion to Integrated as follows:
3. Connect to the Extract database, and grant the following privilege to GG Admin user:
 4. SQL>exec dbms_goldengate_auth.grant_admin_privilege('<ggadmin>')
 5. Login into GGSCI.
 6. Check to see if upgrade is possible.
 7. GGSCI>DBLOGIN USERID <ID> PASSWORD <PW>
GGSCI> INFO <extract_name> UPGRADE
 8. If there are existing open transactions, the upgrade may fail:
 9. GGSCI>stop extract <extract_name>
GGSCI>dblogin userid <ggadmin>,password <password>
 10. Register the extract in the database, if not done already.
 11. GGSCI>register extract <extract_name> database

GGSCI>alter extract <extract_name>,upgrade integrated tranlog

GGSCI>start extract <extract_name>

Oracle SOA Suite Cross References

It is important to understand how to work with Oracle SOA Suite cross references in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Working with XREF using the SOA Cross References KMs](#)
- [Knowledge Module Options Reference](#)

Introduction

Oracle Data Integrator features are designed to work best with Oracle SOA Suite cross references, including mappings that load a target table from several source tables and handle cross references.

Concepts

Cross-referencing is the Oracle Fusion Middleware Function, available through the Oracle BPEL Process Manager and Oracle Mediator, previously Enterprise Service Bus (ESB), and leveraged typically by any loosely coupled integration built on the Service Oriented Architecture. It is used to manage the runtime correlation between the various participating applications of the integration.

General Principles

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-referencing (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 a database schema to store a cross reference information to reference records across systems and data stores.

For more information about cross references, see *Working with Cross References* in the *Developer's Guide for Oracle SOA Suite*.

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.

Cross Reference Table Structures

The XREF data can be stored in multiple cross reference tables and in two formats:

- **Generic (legacy) table** - The table name is XREF_DATA and the table structure stores the cross references for all entities. The table format is as follows:

```
XREF_TABLE_NAME NOT NULL VARCHAR2(2000)
XREF_COLUMN_NAME NOT NULL VARCHAR2(2000)
ROW_NUMBER NOT NULL VARCHAR2(48)
VALUE NOT NULL VARCHAR2(2000)
IS_DELETED NOT NULL VARCHAR2(1)
LAST_MODIFIED NOT NULL TIMESTAMP(6)
```

This table stores cross references for multiple entities. In this table:

- XREF_TABLE_NAME is the name of the cross reference table
- XREF_COLUMN_NAME is the name of the column to be populated. This column name, for example the application name, is used as a unique identifier for the cross reference table.
- ROW_NUMBER stores a unique identifier (*Row Number*) for a given entity instance, regardless of the application
- VALUE is the value of the record identifier for a given entity in this application

A specific XREF_COLUMN_NAME entry called *COMMON* exists to store a generated identifier that is common to all applications.

For example, an ORDER existing in both SIEBEL and EBS will be mapped in a generic table as shown below:

Table 29-1 Example of an XREF_DATA (Partial)

XREF_TABLE_NAME	XREF_COLUMN_NAME	ROW_NUMBER	VALUE
ORDER	SIEBEL	100012345	SBL_101
ORDER	EBS	100012345	EBS_002
ORDER	COMMON	100012345	COM_100

- **Custom (new) table structure** - The table is specific to one entity and has a custom structure. For example:

```
ROW_ID VARCHAR2(48) NOT NULL PK,
APP1 VARCHAR2(100),
APP2 VARCHAR2(100),
...
COMMON VARCHAR2(100),
LAST_MODIFIED TIMESTAMP NOT NULL
```

Where:

- Columns such as APP1 and APP2 are used to store PK values on different applications and link to the same source record
- ROW_ID (*Row Number*) is used to uniquely identify records within a XREF data table.

- COM holds the common value for the integration layer and is passed among participating applications to establish the cross reference

The same ORDER existing in both SIEBEL and EBS would be mapped in a custom XREF_ORDER table as shown below:

Table 29-2 Example of a Custom Table: XREF_ORDERS (Partial)

ROW_ID	SIEBEL	EBS	COMMON
100012345	SBL_101	EBS_002	COM_100

See [Designing a Mapping with the Cross-References KMs](#) and [Knowledge Module Options Reference](#) for more information.

Handling Cross Reference Table Structures

The IKM SQL Control Append (SOA XREF) provides the following parameters to handle these two table structures:

- XREF_DATA_STRUCTURE: This option can be set to `legacy` to use the XREF_DATA generic table, or to `new` to use the custom table structure.

If using the generic table structure, you must set the following options:

- XREF_TABLE_NAME: Value inserted in the XREF_TABLE_NAME column of the XREF_DATA table. In the example above (See [Table 29-1](#)) this option would be ORDER.
- XREF_COLUMN_NAME: Value inserted in the XREF_COLUMN_NAME column of the XREF_DATA table. This value corresponds to the application that is the target of the current mapping. In the example above (See [Table 29-1](#)), this option would take either the value SIEBEL or EBS depending on which system is targeted.

If using the custom table structure, you must use the following options:

- XREF_DATA_TABLE: Name of the cross reference table. It defaults to XREF_DATA. In the example above (See [Table 29-2](#)), this table name would be XREF_ORDER.
- XREF_DATA_TABLE_COLUMN: Name of the column that stores the cross references for the application that is the target of the current mapping. In the example above (See [Table 29-2](#)), this option would take either the value SIEBEL or EBS depending on which system is targeted.

Knowledge Modules

Oracle Data Integrator provides the Knowledge Modules (KM) listed in [Table 29-3](#) for handling SOA cross references (XREF).

These new Knowledge Modules introduce parameters to support SOA cross references. See [Cross Reference Table Structures](#) and [Designing a Mapping with the Cross-References KMs](#) for more information on these parameters.

Table 29-3 SOA XREF KMs

Knowledge Module	Description
LKM SQL to SQL (SOA XREF)	<p>This KM replaces the LKM SQL to SQL (ESB XREF).</p> <p>This KM supports cross references while loading data from a standard ISO source to any ISO-92 database.</p> <p>Depending of the option SRC_UPDATE_DELETE_ACTION, this LKM can DELETE or UPDATE source records.</p> <p>The LKM SQL to SQL (SOA XREF) has to be used in conjunction with the IKM SQL Control Append (SOA XREF) in the same mapping.</p>
LKM MSSQL to SQL (SOA XREF)	<p>This KM replaces the LKM MSSQL to SQL (ESB XREF).</p> <p>This KM is a version of the LKM SQL to SQL (SOA XREF) optimized for Microsoft SQL Server.</p>
IKM SQL Control Append (SOA XREF)	<p>This KM replaces the IKM SQL Control Append (ESB XREF).</p> <p>This KM provides support for cross references while integrating data in any ISO-92 compliant database target table in truncate/insert (append) mode. This KM provides also data control: Invalid data is isolated in an error table and can be recycled. When loading data to the target, this KM also populates PK/GUID XREF table on a separate database.</p> <p>This IKM SQL Control Append (SOA XREF) has to be used in conjunction with the LKM SQL to SQL (SOA XREF) or LKM MSSQL to SQL (SOA XREF).</p>

Overview of the SOA XREF KM Process

To load the cross reference tables while performing integration with Oracle Data Integrator, you must use the SOA XREF knowledge modules. These knowledge modules will load the cross reference tables while extracting or loading information across systems.

Note:

In order to maintain the cross referencing between source and target systems, the LKM and IKM supporting cross referencing must be used in conjunction.

The overall process can be divided into the following three main phases:

1. Loading Phase (LKM)
2. Integration and Cross-Referencing Phase (IKM)
3. Updating/Deleting Processed Records (LKM)

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 mapping) 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* can also be automatically 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. 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 mappings 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 mapping are deleted.
- If SRC_UPDATE_DELETE_ACTION takes the UPDATE value, a source column of the source table will be updated with an expression for all the processed source records. The following KM options parameterize this behavior:
 - SRC_UPD_COL: Name of the source column to update
 - SRC_UPD_COL_EXPRESSION: Expression used to generate the value to update in the column

It is possible to execute delete and update operations on a table different table from the source table. To do this, you must set the following KM options in the LKM:

- SRC_PK_LOGICAL_SCHEMA: Oracle Data Integrator Logical schema containing the source table to impact.
- SRC_PK_TABLE_NAME: Name of the source table to impact.

- SRC_PK_TABLE_ALIAS: Table alias for this table.

Installation and Configuration

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

- [System Requirements and Certifications](#)
- [Technology Specific Requirements](#)
- [Connectivity Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>.

Technology Specific Requirements

There are no technology requirements for using Oracle SOA Suite cross references in Oracle Data Integrator. The requirements for the Oracle Database and Microsoft SQL Server apply also to Oracle SOA Suite cross references. For more information, see:

- [Oracle Database](#)
- [Microsoft SQL Server](#)

Connectivity Requirements

There are no connectivity requirements for using Oracle SOA Suite cross references in Oracle Data Integrator. The requirements for the Oracle Database and Microsoft SQL Server apply also to Oracle SOA Suite cross references. For more information, see:

- [Oracle Database](#)
- [Microsoft SQL Server](#)

Working with XREF using the SOA Cross References KMs

This section consists of the following topics:

- [Defining the Topology](#)
- [Setting up the Project](#)
- [Designing a Mapping with the Cross-References KMs](#)

Defining the Topology

The steps to create the topology in Oracle Data Integrator, which are specific to projects using SOA XREF KMs, are the following:

1. Create the data servers, physical and logical schemas corresponding to the sources and targets.
2. Create a data server and a physical schema for the Oracle or Microsoft SQL Server technology as described in the following sections:
 - [Creating an Oracle Data Server](#) and [Creating an Oracle Physical Schema](#)
 - [Creating a Microsoft SQL Server Data Server](#) and [Creating a Microsoft SQL Server Physical Schema](#)

This data server and this physical schema must point to the Oracle instance and schema or to the Microsoft SQL Server database containing the cross reference tables.

3. Create a logical schema called *XREF* pointing to the physical schema. containing the cross reference table.

See [Creating a Logical Schema](#) in *Administering Oracle Data Integrator* for more information.

Setting up the Project

Import the following KMs into your project, if they are not already in your project:

- IKM SQL Control Append (SOA XREF)
- LKM SQL to SQL (SOA XREF) or LKM MSSQL to SQL (SOA XREF) if using Microsoft SQL Server

Designing a Mapping with the Cross-References KMs

To create a mapping, 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 a mapping with the source and target datastores which will have the cross references.
2. Create joins, filters and mappings as usual.

Mapping the Common ID: If you want to map in a target column the *Common ID* generated for the cross reference table, check the UD1 flag for this column and enter a dummy mapping. For example a constant value such as 'X'.

3. In the Physical diagram of the mapping, select the access point for the execution unit containing the source table to cross reference. The Property Inspector for this object opens.
4. In the Loading Knowledge Module tab, select the LKM SQL to SQL (SOA XREF) or LKM MSSQL to SQL (SOA XREF) if the source data store is in Microsoft SQL Server.
5. Specify the KM options as follows:

- Specify in SRC_PK_EXPRESSION the expression representing the *Source Primary Key* value that you want to store in the XREF table.
If the source table has just one attribute defined as a key, enter the attribute name (for example SEQ_NO).
If the source key has multiple attributes, specify the expression to use for deriving the key value. For example, if there are two key attributes SEQ_NO and DOC_DATE in the table and you want to store the concatenated value of those attributes as your source value in the XREF table enter SEQ_NO || DOC_DATE. This option is mandatory.
- Optionally set the SRC_UPDATE_DELETE_ACTION to impact the source table, as described in [Updating/Deleting Processed Records \(LKM\)](#)
- 6. In the Physical diagram of the mapping, select the access point for your staging area. The Property Inspector opens for this object.
- 7. In the Integration Knowledge Module tab, select the **IKM SQL Control Append (SOA XREF)**.
- 8. Specify the KM options as follows:
 - XREF_DATA_STRUCTURE: Enter *New* to use the new XREF_DATA Table structure. Otherwise enter *Legacy* to use legacy XREF_DATA Table. Default is *New*. Configure the options depending on the table structure you are using, as specified in [Handling Cross Reference Table Structures](#)
 - XREF_SYS_GUID_EXPRESSION: Enter the 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. 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 attributes, you must use the standard mapping rules of ODI to indicate which source identifier to populate in which attribute.

If the target attribute 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 attribute. At runtime, this dummy mapping will be overwritten with the generated common identifier by the integration knowledge module. Make sure to flag this target attribute with UD1.

Knowledge Module Options Reference

This section lists the KM options for the following Knowledge Modules:

- [Table 29-4](#)
- [LKM MSSQL to SQL \(SOA XREF\)](#)
- [Table 29-5](#)

Table 29-4 LKM SQL to SQL (SOA 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 Updating/Deleting Processed Records (LKM) for more information.
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 mapping) 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 or update 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 mapping. 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.
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.

Table 29-4 (Cont.) LKM SQL to SQL (SOA XREF)

Option	Values	Mandatory	Description
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 (SOA XREF)

See [Table 29-4](#) for details on the LKM MSSQL to SQL (SOA XREF) options.

Table 29-5 IKM SQL Control Append (SOA XREF)

Option	Values	Mandatory	Description
INSERT	Yes No	Yes	Automatically attempts to insert data into the Target Datastore of the Mapping.
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, if using Legacy XREF table structure.	Table Name to use in the XREF table. Example: ORDERS. See Handling Cross Reference Table Structures for more information.
XREF_COLUMN_NAME	Column name	Yes, if using Legacy XREF table structure.	Primary key column name to use as a literal in the XREF table. See Handling Cross Reference Table Structures for more information.

Table 29-5 (Cont.) IKM SQL Control Append (SOA XREF)

Option	Values	Mandatory	Description
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.
XREF_DATA_STRUCTURE	New Legacy	Yes	Enter <i>New</i> to use the new XREF_DATA Table structure.. Otherwise enter <i>Legacy</i> to use legacy XREF_DATA Table. Default is <i>New</i> . See Handling Cross Reference Table Structures for more information.
XREF_DATA_TABLE	XREF table name	No. Can be used with custom XREF table structure.	Enter the name of the table storing cross reference information. Default is XREF_DATA. See Handling Cross Reference Table Structures for more information.
XREF_DATA_TABLE_COLUMN	XREF data table column name	Yes, if using custom XREF table structure	For new XREF data structure only: Enter the column name of the XREF data table to store the source key values. See Handling Cross Reference Table Structures for more information.

Oracle Object Storage

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

NOT_SUPPORTED:

This chapter applies only to Data Integration Platform Cloud.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Object Storage Model](#)
- [Working with Oracle Object Storage Tools](#)
- [Designing a Mapping](#)
- [Setting up an Integration Project](#)

Introduction

Oracle Object Storage is a highly secure, better performant, durable storage platform. It allows you to store or retrieve unlimited data anytime, safely and securely using its web-based console within the cloud platform.

Oracle Data Integrator (ODI) seamlessly integrates with Oracle Object Storage. With this integration, you can now connect to Oracle Object Storage from ODI for uploading, downloading and deleting files/objects onto/from local directory or HDFS.

Concepts

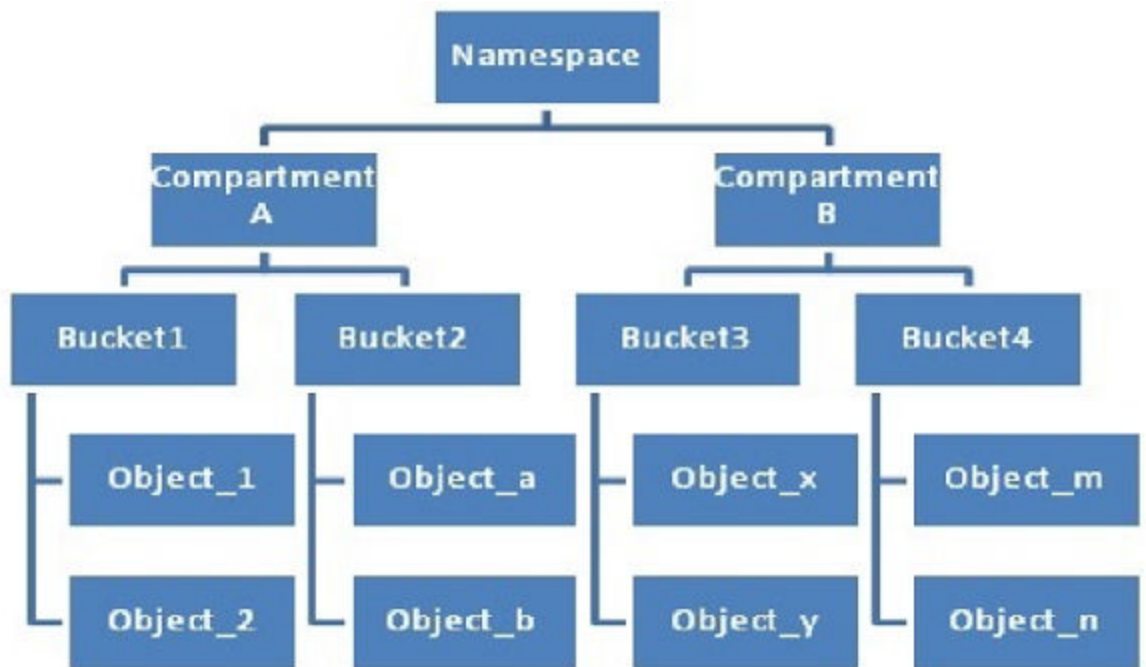
Oracle Object Storage comprises of the following components:

- **Objects:** All data, regardless of content type, is stored as an object in Oracle Object Storage.
- **Buckets:** A bucket is a container that stores objects. You can store objects in one or multiple buckets under a single tenancy.
- **Namespace:** A namespace is the logical construct that lets you own your personal bucket namespace. Since bucket names are not global you have more freedom and flexibility with how you assign names to buckets. Each Object Storage Cloud tenant is associated with one default system assigned namespace.

- **Compartments:** A compartment is a collection of related resources that can be accessed only by those groups that have been given permission by an administrator to access those resources. Compartment is not necessarily a concept that is exclusively associated with the Oracle Object Storage, but it's pertinent because all buckets exist in a compartment.

Below image helps you to understand how the Oracle Object Storage components fit together. If a user has been granted access to Compartment A, they will only be able to access Bucket1 or Bucket2 and objects that are stored in these buckets.

Figure 30-1 Object Storage Components



Refer to <https://docs.us-phoenix-1.oraclecloud.com/Content/Object/Concepts/objectstorageoverview.htm> , for more details on Oracle Object Storage.

Installation and Configuration

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

- [System Requirements & Certifications](#)
- [Technology Specific Requirements](#)

System Requirements & 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/technetwork/middleware/data-integrator/documentation/index.html>

Technology Specific Requirements

The technology specific requirement for using Oracle Object Storage in ODI are:

- A new, dedicated pre-built technology called “Oracle Object Storage”.
- Create a new Data Server from this technology and then create corresponding Physical and Logical schema.

Data Type mappings to and from some major platforms are defined for these data types. Supported data types for this technology are:

- Array
- Boolean
- Bytes
- Complex
- Date
- Double
- Enum
- Fixed
- Float
- Integer
- Long
- Map
- Number
- String
- Struct
- Union

Setting up the Topology

Setting up the topology consists of:

- [Creating an Oracle Object Storage Data Server](#)
- [Creating an Oracle Object Storage Physical Schema](#)

Creating an Oracle Object Storage Data Server

Create a data server for the Oracle Object Storage technology using the standard procedure, as described in *Creating a Data Server in Administering Oracle Data Integrator*. This section details only the fields required or specific for defining Oracle Object Storage data server:

- In the **Definition** tab:

The Data Server page contains the following fields and they are grouped into different categories such as –

- **Data Server**

- * **Name** – Name of the data server that will appear in Oracle Data Integrator.

- **General**

- * **Region** – Oracle Object Storage region. A region is a localized geographic area, and an availability domain is one or more data centers located within a region. A region is composed of several availability domains. Most Oracle Cloud Infrastructure resources are either region-specific, such as a virtual cloud network, or availability domain-specific, such as a compute instance.

- * **Tenant OCID** – Tenant's Oracle Cloud ID. Every Oracle Cloud Infrastructure resource has an Oracle-assigned unique ID called an Oracle Cloud Identifier (OCID). It's included as part of the resource's information in both the Console and API. To find your tenancy's OCID, navigate to Console -> Menu -> Administration -> Tenancy Details -> Tenancy Information -> OCID.

For example, `ocid1.tenancy.oc1..aaaaaaaaawjnv47knr7uuuvqar5bs
hnspi6xoxsfebh3vy72fi4swgrkvuvq`

- * **User OCID** – Oracle Cloud ID of the user logging into Oracle Object Storage.

In the Console on the page showing the user's details. To get to that page:

- * If you're signed in as the user, click the user icon present in the top-right corner of the Console, and then click User Settings.
- * If you're an administrator doing this for another user, instead click Identity, click Users, and then select the user from the list.

- **Security**

- * **Private Key File** – Click the browse button to choose the location of the private key file (in PEM format)
- * **Passphrase** – Passphrase is the password used while generating the private key

You can use the following OpenSSL commands to generate the key pair in the required PEM format. If you're using Windows, you'll need to install Git Bash for Windows and run the commands with that tool.

1. If you haven't already, create a `.oci` directory to store the credentials:

```
mkdir ~/.oci
```

2. Generate the private key with one of the following commands:

- * To generate the key, encrypted with a passphrase you provide when prompted:

```
openssl genrsa -out ~/.oci/oci_api_key.pem -aes128 2048
```


- * For Windows, you may need to insert `-passout stdin` to be prompted for a passphrase. The prompt will just be the blinking cursor, with no text.

```
openssl genrsa -out ~/.oci/oci_api_key.pem -aes128 2048
```

- * To generate the key with no pass-phrase:

```
openssl genrsa -out ~/.oci/oci_api_key.pem 2048
```

- * **Fingerprint** – Fingerprint that is generated for the public key

You can get the key's fingerprint with the following OpenSSL command. If you're using Windows, you'll need to install Git Bash for Windows and run the command with that tool.

```
openssl rsa -pubout -outform DER -in ~/.oci/oci_api_key.pem |  
openssl md5 -c
```

When you upload the public key in the Console, the fingerprint is also automatically displayed there. For example, 12:34:56:78:90:ab:cd:ef:
12:34:56:78:90:ab:cd:ef

– Swift Connectivity

- * **Tenant Name** – Name of the tenant
- * **User Name** – Name of the user logging into Oracle Object Storage
- * **Swift Password** – Password generated for swift connectivity.

These Swift Connectivity parameters are used by the Autonomous Data Warehouse Cloud Service to connect to Oracle Object Storage technology.

Creating an Oracle Object Storage Physical Schema

Create an Oracle Object Storage physical schema using the standard procedure, as described in [Creating a Physical Schema in Administering Oracle Data Integrator](#). Oracle Object Storage specific parameters are:

- **Name:** Name of the physical schema created
- **Bucket (Schema):** It specifies the Oracle Object Storage Bucket name from which upload, download or the delete operation will happen. Select the required bucket from the Bucket Name drop-down list.
- **Directory (Work Schema):** This is the temporary folder on the local system used for getting files from Oracle Object Storage bucket during reverse engineering. If the directory does not exist it will be created. Specify the required location in the local system.

Create a logical schema for this physical schema using the standard procedure, as described in [Creating a Logical Schema in Administering Oracle Data Integrator](#) and associate it with a relevant context. We use the created logical schema for getting Oracle Object Storage instance details. These details are used for connecting to Oracle Object Storage technology. It is also used to get the bucket details associated with the physical schema which in turn is associated to this logical schema.

Creating and Reverse-Engineering an Oracle Object Storage Model

This section contains the following topics:

- [Creating an Oracle Object Storage Model](#)
- [Reverse Engineer an Oracle Object Storage Model](#)

Creating an Oracle Object Storage Model

An Oracle Object Storage model is a set of data stores, corresponding to files stored in an Oracle Object Storage bucket. In a given context, the logical schema corresponds to one physical schema. You can create a model from the logical schema for the Oracle Object Storage technology. The bucket schema of this physical schema is the Oracle Object Storage bucket containing all the files. You can create new ODI Data store that will represent a file in Oracle Object Storage so that it can be used in mappings.

Create an Oracle Object Storage model using the standard procedure, as described in [Creating a Model of Developing Integration Projects with Oracle Data Integrator](#).

Reverse Engineering an Oracle Object Storage Model

Oracle Data Integrator provides specific methods for reverse-engineering Oracle Object Storage files. Oracle Object Storage supports the following types of reverse engineering:

- [Reverse-Engineering Delimited Files from Oracle Object Storage](#)
- [Reverse-engineering Fixed Files from Oracle Object Storage](#)
- [Reverse Engineering HDFS Files from Oracle Object Storage](#)

Reverse-Engineering Delimited Files from Oracle Object Storage

To perform a delimited file reverse engineering:

1. In the **Models** accordion, right click your **Object Storage Model** and select **New Data store**. The **Data Store Editor** opens.
2. In the **Definition** tab, enter the following fields:
 - **Name**: Name of this data store
 - **Resource Name**: Click the Search icon, to select the required file from the list of files present in Oracle Object Storage for the configured bucket.
3. Go to the **Storage** tab, to describe the type of file. Set the fields as follows:
 - **File Format**: Delimited
 - **Heading (Number of Lines)**: Enter the number of lines of the header. Note that if there is a header, Oracle Data Integrator uses the first line of the header 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 **Data Store Editor**, go to the **Attributes** tab.
 6. In the editor toolbar, click **Reverse Engineer**.
 7. Verify the data type and length for the reverse engineered attributes. Oracle Data Integrator infers the field data types 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**.

Reverse-engineering Fixed Files from Oracle Object Storage

Oracle Data Integrator provides a graphic wizard to define the columns of a fixed file. To reverse-engineer a fixed file from Oracle Object Storage using the wizard:

1. In the **Models** accordion, right click your **Object Storage Model** and select **New Data store**. The **Data store Editor** opens.
2. In the **Definition** Tab, enter the following fields:
 - **Name**: Name of this data store
 - **Resource Name**: Sub-directory (if needed) and name of the file. It lists all the files present in Oracle Object Storage for the configured bucket.
3. Go to the **Storage** 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 Data store Editor, go to the **Attributes** tab.
6. In the editor toolbar, click **Reverse Engineer**. The **Attributes Setup Wizard** appears. The **Attributes Setup Wizard** displays the first records of your file.
7. Click on the ruler (above the file contents) to create markers delimiting the attributes. You can right-click within the ruler to delete a marker.
8. Attributes are created with pre-generated names (C1, C2, and so on). You can edit the attribute name by clicking in the attribute header line (below the ruler).
9. In the properties panel (on the right), you can edit all the parameters of the selected attribute. You should set at least the Attribute Name, Data type, and Length for each attribute.
10. Click **OK**, when the attributes definition is complete.
11. From the **File** main menu, select **Save**.

Reverse-Engineering JSON, Avro and Parquet Storage Formats

Oracle Object Storage technology supports reverse engineering JSON, Avro and Parquet storage formats, attributes, data types, and data type properties. If one of these storage format types is selected, then reverse engineering is based on the Schema File specified, and not on a sample data file. The schema file should be accessible in local file system.

To reverse-engineer JSON, Avro and Parquet storage formats, perform the following steps:

1. In the Models accordion, right click your Object Storage Model and select **New Data store**. The **Data Store Editor** opens.
2. In the **Definition** tab, enter the following fields:
 - **Name**: Name of this data store
 - **Resource Name**: Click the Search icon, to select the required file from the list of files present in Oracle Object Storage for the configured bucket.
3. From the **Storage** Tab, select the Storage Format from the **Storage Format** drop-down list and specify the complete path of the schema file in the **Schema File** field.

The schema file should be located in the local file system.

4. From the File main menu, select **Save**.



Note:

There is no need to import an RKM into the project.

Working with Oracle Object Storage Tools

You can upload, download and delete files to/ from Oracle Object Storage through Oracle Data Integrator. The Object Storage tools that are helpful to perform the following operations are:

- [Object Storage Upload tool](#)
- [Object Storage Download tool](#)
- [Object Storage Delete tool](#)



Note:

Apart from ODI Studio, you can also work with ODI Object Storage Tools from command line.

Uploading Files/Objects to Oracle Object Storage

ODI Object Storage Upload tool is used to upload single, multiple files, or an entire directory from HDFS or a local file system on to Oracle Object Storage.

 **Note:**

The upload operation fails if the selected bucket does not exist.

To upload file(s) or directories to Object Storage,

- Create a new **Project**.

For more details on how to create a project, see [Creating an Integration Project of Developing Integration Projects with Oracle Data Integrator](#).

- Below the created Project folder, create a **Package** .

For more details on how to create a package, see [Creating and Using Packages of Developing Integration Projects with Oracle Data Integrator](#).

- Select `OdiObjectStorageUpload` tool available in the Toolbox. Move it to the created package.

 **Note:**

All the properties of the tool are displayed under **General** Tab.

- Configure the required properties:

Table 30-1 ODI Object Storage Upload Tool Properties

Parameter	Description
Target Logical schema	Target Logical schema name configured for Oracle Object Storage Data Server.
Source Logical schema	Name of the source Logical schema configured for File or HDFS Data Server for upload of Local or HDFS Files to Oracle Object Storage.

Table 30-1 (Cont.) ODI Object Storage Upload Tool Properties

Parameter	Description
File Names filter	Field to specify one or more files to be uploaded to Oracle Object Storage recursively. It also supports the list of files separated by as a delimiter. The pattern followed is: <ul style="list-style-type: none"> – *.txt - should upload all the files ending with .txt – test* - uploads all the files and directories that matches with prefix "test" – *test* - uploads all the files and directories having substring "test" – test.xml test1.xml test2.xml - Uploads all the files specified – test* test1* - Uploads all the files matching pattern test* and test1* – test.xml - Only one file is uploaded
Overwrite	This parameter indicates if upload operation should overwrite an existing file or not. Default value for this parameter is No .
Retry on error	It represents the number of times the retry attempt should occur when a failure or error happens during upload.
Retry interval seconds	Retry interval indicates after how many seconds a retry attempt should happen.

For more details on the usage of the above parameters, refer to ODI Object Storage Upload tool in Oracle Data Integrator Tools Reference.

- Save and execute the package.

The required files from the source directory are uploaded to the target location of Oracle Object storage.

- Upon successful upload, you can find a detailed log of this upload operation at the **Details** tab. To get to the details tab, from the Operator tab, expand the associated session for the upload tool and open the Session task window to find the Details tab with the required log information.

The details include:

- Source directory is : <source directory path>
- Target bucket is : < Object storage bucket name>
- Filter used is : <input filter>
- Number of files uploaded:<Total number of files that were uploaded>
- Uploaded files are:<File1, File2>
- Number of files failed:<Total number of files that were not uploaded>
- Failed files are: <File1, File2>

Downloading Files/Objects from Oracle Object Storage

ODI Object Storage Download tool is used to download single, multiple files, or an entire directory to HDFS or a local file system from Oracle Object Storage. To download file(s) or directories from Object Storage,

- Create a new **Project**

For more details on how to create a project, see *Creating an Integration Project of Developing Integration Projects with Oracle Data Integrator*.

- Below the created Project folder, create a **Package**

For more details on how to create a package, see *Creating and Using Packages of Developing Integration Projects with Oracle Data Integrator*.

- Select `OdiObjectStorageDownload` tool available in the Toolbox. Move it to the created package.

 **Note:**

All the properties of the tool are displayed under General Tab.

- Configure the required properties:

Table 30-2 ODI Object Storage Download Tool Properties

Parameter	Description
Source Logical schema	Name of the Source Logical schema configured for Oracle Object Storage Data Server.
Target Logical schema	Generally, the download operation downloads the file from Oracle Object Storage to Local or HDFS file system. The Target logical schema specifies whether the files are downloaded to Local or HDFS file system.
File Names filter	Field to specify one or more files to be downloaded from Oracle Object Storage recursively. It also supports delimiter for separated files list. The pattern followed is: <ul style="list-style-type: none"> – *.txt - should download all files ending with .txt – test* - Downloads all the files and directories that matches with prefix “test” – *test* - Downloads all the files and directories having substring “test” – test.xml test1.xml test2.xml - Downloads all the files specified – test* test1* - Downloads all the files matching pattern test* and test1* – test.xml - Only one file is downloaded
Overwrite	This parameter indicates, if download operation should overwrite an existing file or not. The default value for this parameter is No .
Retry on error	It represents the number of times the retry attempt should occur when a failure or error happens during download.

Table 30-2 (Cont.) ODI Object Storage Download Tool Properties

Parameter	Description
Retry interval seconds	Retry interval indicates after how many seconds a retry attempt should happen.

For more details on the usage of the above parameters, refer to ODI Object Storage Download tool of Oracle Data Integrator Tools Reference.

- Save and execute the package
The required files from Oracle Object storage are downloaded to the configured target location.
- Upon successful Download, you can find a detailed log of this download operation at the **Details** tab. To get to the details tab, from the Operator tab, expand the associated session for the download tool and open the Session task window to find the Details tab with the required log information.
 - Source bucket is : <Object storage bucket name>
 - Target directory is : <target directory path>
 - Filter used is : <input filter>
 - Number of files downloaded:<Total number of files that were downloaded>
 - Downloaded files are:<File1, File2>
 - Number of files failed:<Total number of files that were not downloaded>
 - Failed files are: <File1, File2>

Deleting Files/Objects from Oracle Object Storage

ODI Object Storage Delete tool is used to delete single, multiple files, or an entire directory present in Oracle Object Storage.

To delete file(s) or directories from Object Storage,

- Create a new **Project**
For more details on how to create a project, see Creating an Integration Project of Developing Integration Projects with Oracle Data Integrator.
- Below the created Project folder, create a **Package**
For more details on how to create a package, see Creating and Using Packages of Developing Integration Projects with Oracle Data Integrator.
- Select `OdiObjectStorageDelete` tool available in the Toolbox. Move it to the created package.

Note:

All the properties of the tool are displayed under General Tab.

- Configure the required properties:

Table 30-3 ODI Object Storage Delete Tool Properties

Parameter	Description
Target Logical schema	Target logical schema has the details of Oracle Object Storage Data Server which contains the files and directories that are to be deleted.
File Names filter	Field to specify one or more files or directories to be deleted from Oracle Object Storage recursively. It also supports delimiter for separated files list. The pattern followed is: <ul style="list-style-type: none"> – *.txt - should delete all files ending with .txt – test* - Deletes all the files and directories that matches with prefix "test" – *test* - Deletes all the files and directories having substring "test" – test.xml test1.xml test2.xml - Deletes all the files specified – test* test1* - Deletes all the files and directories matching pattern test* and test1* – test.xml - Only one file is deleted.
Retry on error	It represents the number of times the retry attempt should occur when a failure or error happens during delete.
Retry interval seconds	Retry interval indicates after how many seconds a retry attempt should happen.

For more details on the usage of above parameters, refer to ODI Object Storage Delete Tool of Oracle Data Integrator Tools Reference.

- Save and execute the package.
The selected files are deleted from Oracle Object Storage.
- Upon successful Deletion, you can find a detailed log of this delete operation at the **Details** tab. To get to the details tab, from the Operator tab, expand the associated session for the delete tool and open the Session task window to find the Details tab with the required log information.
 - Target bucket is : <Object storage bucket name>
 - Filter used is : <input filter>
 - Number of files deleted:<Total number of files that were deleted>
 - Deleted files are:<File1, File2>
 - Number of files failed:<Total number of files that were not deleted>
 - Failed files are: <File1, File2>

Designing a Mapping

You can use a file/HDFS or SQL technology such as Oracle as a source of a mapping and its target as Oracle Object Storage technology.

Oracle Object Storage physical schema is represented by Object Storage Bucket. Some properties such as user name/password are retrieved from Oracle Object Storage data server. If temporary local files are created (if needed), its location can be defined through the option `TEMP_SCHEMA KM`. These temporary Object Storage files can be removed through new ODI cleanup tools. If you use transform components, they need to be moved to the source execution unit in case of SQL as a source. File as a source does not support source transformations and no transformations are supported for the target as well.

The KM choice for a mapping or a check determines the abilities and performances of this mapping or check. The recommendations listed here help in the selection of the KM for different situations concerning an Oracle Object Storage data server.

Setting up an Integration Project

Setting up a project using the Oracle Object Storage technology follows the standard procedure. See *Creating an Integration Project of the Developing Integration Projects with Oracle Data Integrator*.

You can use the following knowledge modules for loading data into Oracle Object Storage:

- [LKM File to Oracle Object Storage](#)
- [LKM File to Oracle Object Storage Direct](#)
- [LKM SQL to Oracle Object Storage](#)
- [LKM SQL to Oracle Object Storage Direct](#)

LKM File to Oracle Object Storage

This LKM helps to upload data from local or HDFS File to Oracle Object Storage. The name of the file as well as its structure remains the same. No transformation on the data can be performed. This LKM is used in a staging area physical node.

This KM invokes the ODI tool `OdiObjectStorageUpload`, for uploading file(s) to Oracle Object Storage.

For Example — `OdiObjectStorageUpload "-TRG_LOGICAL_SCHEMA=Object Storage - SRC1" "-SRC_LOGICAL_SCHEMA=FILE_GENERIC_TMP" "- FILE_NAMES_FILTER=person_upload.csv" "-OVERWRITE=true"`

This KM has the following option:

- `CLEANUP_TEMPORARY_OBJECTS`— It cleans-up temporary objects.

Set this property to **True**, if you wish to automatically clean-up the temporary objects created.

LKM File to Oracle Object Storage Direct

This LKM helps to upload local or HDFS file(s) onto Oracle Object Storage target directly. The name of the file as well as its structure remains the same. No transformation on the data can be performed.

This KM invokes the ODI tool `OdiObjectStorageUpload`, for uploading file(s) to Oracle Object Storage.

For Example — `OdiObjectStorageUpload "-TRG_LOGICAL_SCHEMA=Object Storage - SRC1" "-SRC_LOGICAL_SCHEMA=FILE_GENERIC_TMP" "- FILE_NAMES_FILTER=person_upload.csv" "-OVERWRITE=true"`

This KM has the following options:

- `CLEANUP_TEMPORARY_OBJECTS`— It is used to clean-up temporary objects created. Set this property to **True**, if you wish to automatically clean-up the temporary objects created.
- `OVERWRITE_TARGET_FILE` — It is used to overwrite the target file. If set to **True**, the Oracle Object Storage target files are overwritten.
- `ADD_COMPRESSION`— It is used to compress data before loading. Set this property to **True**, if you wish to compress source data before loading it onto Oracle Object Storage. Use the additional options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` to define compression preferences.
- `COMPRESSION_TYPE`— It is used to define the compression type. Select the type of compression you wish to apply on source data before loading it onto Oracle Object Storage.
- `KEEP_SOURCE_FILES`— It is used to retain the original source files after compression. Set this property to **True** (by default), if you wish to compress the target files and retain the original files.

Note:

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

LKM SQL to Oracle Object Storage

This LKM helps to upload the result of a SQL query to Oracle Object Storage. Only transformation on source are supported. SQL data is unloaded to a temporary local file, which is then uploaded to Object Storage. The temporary file location is specified through the `TEMP_SCHEMA` KM option. This LKM is used in a staging area physical node.

This KM invokes the ODI tools `OdiSqlUnload` to unload SQL query data to a file and `OdiObjectStorageUpload` for uploading the file(s) onto Oracle Object Storage.

Listed below are examples for these tools:

- `OdiSqlUnload`

```
OdiSqlUnload "-FILE=/tmp/person_upload.csv" "-
DRIVER=oracle.jdbc.OracleDriver" "-URL=jdbc:oracle:thin:@//slc03sap:
```

```
1521/flex" "-USER=system" "-PASS=<@=odiRef.getInfo("SRC_ENCODED_PASS")
@>" "-FILE_FORMAT=VARIABLE" "-FIELD_SEP=," "-ROW_SEP=

" "-DATE_FORMAT=yyyy/MM/dd HH:mm:ss" "-CHARSET_ENCODING=ISO8859_1" "-
FETCH_SIZE=2"

SELECT
  PER.PID AS PID ,
  PER.PNAME AS PNAME
FROM
  UT_TD_D_1.PERSON PER
WHERE
  (PER.PID = 2)
```

- OdiObjectStorageUpload

```
OdiObjectStorageUpload "-TRG_LOGICAL_SCHEMA=Object Storage - SRC1" "-
SRC_LOGICAL_SCHEMA=FILE_GENERIC_TMP" "-
FILE_NAMES_FILTER=person_upload.csv" "-OVERWRITE=true"
```

This KM has the following options:

- **CLEANUP_TEMPORARY_OBJECTS**— It is used to clean-up temporary objects created. Set this property to **True**, if you wish to automatically clean-up the temporary objects created.
- **TEMP_SCHEMA** — It is used to specify the name of logical schema defining the location the temporary file that will be stored before uploading the data onto Oracle Object Storage. This is a File technology logical schema. The temporary file is stored in a local file system where ODI agent is running.
- **DATE_FORMAT** — It specifies the output format used for date data types.
- **CHARSET_ENCODING** — It specifies the character set encoding.
- **FETCH_SIZE** — It specifies the number of rows (records read) requested by ODI agent on each communication with the data server.

LKM SQL to Oracle Object Storage Direct

This LKM helps to upload the result of a SQL query to Oracle Object Storage. Only transformation on source are supported. SQL data is unloaded to a temporary local file, which is then uploaded to Oracle Object Storage. The temporary file location is specified through the **TEMP_SCHEMAKM** option. Data can be uploaded in Delimited, Fixed or XML formats. This LKM is a transparent target KM (loads data directly into target). It has to be assigned to target execution unit AP node.

This KM invokes the ODI tools `OdiSqlUnload` to unload SQL query data to a file and `OdiObjectStorageUpload` for uploading the file(s).

Listed below are examples for these tools:

- OdiSqlUnload

```
OdiSqlUnload "-FILE=/tmp/person_upload.csv" "-
DRIVER=oracle.jdbc.OracleDriver" "-URL=jdbc:oracle:thin:@//slc03sap:
1521/flex" "-USER=system" "-PASS=<@=odiRef.getInfo("SRC_ENCODED_PASS")
```

```
@>" -FILE_FORMAT=VARIABLE" "-FIELD_SEP=," "-ROW_SEP=
" "-DATE_FORMAT=yyyy/MM/dd HH:mm:ss" "-CHARSET_ENCODING=ISO8859_1" "-
FETCH_SIZE=2"
```


```
SELECT
  PER.PID AS PID ,
  PER.PNAME AS PNAME
FROM
  UT_TD_D_1.PERSON PER
WHERE
  (PER.PID = 2)
```

- OdiObjectStorageUpload

```
OdiObjectStorageUpload "-TRG_LOGICAL_SCHEMA=Object Storage - SRC1" "-
SRC_LOGICAL_SCHEMA=FILE_GENERIC_TMP" "-
FILE_NAMES_FILTER=person_upload.csv" "-OVERWRITE=true"
```

This KM has the following options:

- CLEANUP_TEMPORARY_OBJECTS— It is used to clean-up temporary objects created. Set this property to **True**, if you wish to automatically clean-up the temporary objects created.
- OVERWRITE_TARGET_FILE— It is used to overwrite target file. If set to **True**, the Oracle Object Storage target file will be overwritten.
- TEMP_SCHEMA — It is used to specify the name of logical schema defining the location the temporary file that will be stored before uploading the data onto Oracle Object Storage. This is a File technology logical schema. The temporary file is stored in a local file system where ODI agent is running.
- DATE_FORMAT — It specifies the output format used for date data types.
- STORE_AS_XML — It is used to store data as XML file. Set this property to **true**, if you wish to store data as XML file in Oracle Object Storage.
- CHARSET_ENCODING — It specifies the character set encoding.
- XML_CHARSET_ENCODING— It specifies the character encoding indicated in the XML declaration header of the export file.
- ADD_COMPRESSION — It is used to compress data before loading. Set this property to **True**, if you want to compress source data before loading onto Oracle Object Storage. Use the additional options `COMPRESSION_TYPE` and `KEEP_SOURCE_FILES` to define compression preferences.
- FETCH_SIZE — It specifies the number of rows (records read) requested by ODI agent on each communication with the data server.
- COMPRESSION_TYPE — It is used to specify the compression type. Select the type of compression you wish to apply on source data before loading onto Oracle Object Storage.
- KEEP_SOURCE_FILES — It is used to retain the original source files after compression. Set this property to **True** (by default), if you wish to compress the target files and retain the original files.

 **Note:**

gzip supports `KEEP_SOURCE_FILES` option, starting from version 1.6 only.

31

Oracle Storage Cloud Service

This chapter describes how to work with Oracle Storage Cloud Service in Oracle Data Integrator.

NOT_SUPPORTED:

This chapter applies only to Data Integration Platform Cloud.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Oracle Storage Cloud Service Model](#)
- [Working with Oracle Storage Cloud Service Tools](#)

Introduction

Oracle Storage Cloud Service provides a reliable, secure, and scalable object storage solution for storing unstructured data that can be accessed anytime anywhere. It serves as a gateway for data consumption to many OPC services. These cloud services directly pick up files from Oracle Storage Cloud Service and therefore, integration with Oracle Storage Cloud Service becomes very essential and useful to manage end-to-end integration flows using Oracle Data Integrator.

Oracle Data Integrator (ODI) seamlessly integrates with Oracle Storage Cloud Service. With this integration, you can now connect to Oracle Storage Cloud Service from ODI for uploading or downloading files/objects onto/from local directory or HDFS present in Oracle Storage Cloud Service.

Concepts

Oracle Storage Cloud Service is an Infrastructure as a Service (IaaS) product, which provides an enterprise-grade, large-scale, object storage solution for files and unstructured data.

Oracle Storage Cloud Service comprises of the following concepts:

Oracle Storage Cloud Service Hierarchy

Oracle Storage Cloud Service stores data as objects within a flat hierarchy of containers. You can create an object within a container most commonly by uploading a file or from ephemeral unstructured data. A single object can hold up to 5 GB of data, but multiple objects can be linked together to hold more than 5 GB of contiguous data.

A container is a user-created resource, which can hold an unlimited number of objects, unless you specify a quota for the container. Note that containers cannot be nested. You can define custom metadata for both objects and containers.

Storage Types

This integration provides support for both Oracle Standard Storage and Archive Storage.

- Standard Storage - It is useful for storing one or more files that are accessed frequently.
- Archive Storage — It is ideal for storing data that are not frequently accessed such as email archives, data backups, and digital video and so on.

Installation and Configuration

Make sure you have read the information in this section before you start working with the Oracle Storage Cloud Service technology:

- [System Requirements and Certifications](#)
- [Technology Specific Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>

Technology Specific Requirements

The technology specific requirement for using Oracle Storage Cloud Service in ODI are:

- You should have a dedicated pre-built technology named “Oracle Storage Cloud Service” defined similar to Oracle Object Storage.
- As an ODI user, you should be able to create a Data Server from this technology and corresponding Physical and Logical schemas for the created Data Server. These physical and logical schemas are used by ODI Tools supported for Oracle Storage Cloud Service integration, used for uploading and downloading files/objects.

Supported datatypes for this technology are:

- Array
- Boolean
- Bytes
- Complex

- Date
- Double
- Enum
- Fixed
- Float
- Integer
- Long
- Map
- Number
- String
- Struct
- Union

Setting up the Topology

Setting up the topology consists of:

- [Creating an Oracle Storage Cloud Service Data Server](#)
- [Creating an Oracle Storage Cloud Service Physical Schema](#)

Creating an Oracle Storage Cloud Service Data Server

Create a data server for the Oracle Storage Cloud Service technology using the standard procedure, as described in *Creating a Data Server of Administering Oracle Data Integrator* guide. This section details only the fields required or specific for defining Oracle Storage Cloud Service data server:

- In the **Definition** tab:
 1. **Data Server**
 - **Name** – Name of the data server that will appear in Oracle Data Integrator.
 2. **Connection**
 - **Service URL** – Oracle Cloud Storage Service URL. For Example: `https://<identity-domain>.storage.oraclecloud.com`
 - **Service Name** – It denotes the name of the service for the created service URL. For Example - Storage
 - **User Name** - Name of the user logging into the Oracle Storage Cloud Service

 **Note:**

User Names should start with upper case and should not be real server names.

- **Password** – Password of the logged in user
- **Identity Domain** - It denotes the domain specific to the created storage instance. For Example – `https://<identity-domain>.storage.oraclecloud.com`

Creating an Oracle Storage Cloud Service Physical Schema

Create an Oracle Storage Cloud Service physical schema using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator* guide.

Oracle Storage Cloud Service specific parameters are:

- **Name**— Name of the physical schema created.
- **Container Name** — It specifies the container to which you wish to associate the created physical schema. Select the required container from the Container Name drop-down list.
- **Directory (Work Schema)** — This is the temporary folder on the local system used for getting files from Oracle Storage Cloud Service. If the directory does not exist, it is created. Specify the required location in the local system.

Create a logical schema for this physical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator* and associate it with a relevant context.

We use the created logical schema for getting Oracle Storage instance details. These details are used for connecting to Oracle Storage Cloud Service technology.

Creating and Reverse-Engineering an Oracle Storage Cloud Service Model

This section contains the following topics:

- [Creating an Oracle Storage Cloud Service Model](#)
- [Reverse Engineering an Oracle Storage Cloud Service Model](#)

Creating an Oracle Storage Cloud Service Model

An Oracle Storage Cloud Service model is a set of data stores, corresponding to files stored in an Oracle Storage Cloud Service directory.

In a given context, the logical schema corresponds to one physical schema. You can create a model from the logical schema for the Oracle Storage Cloud Service technology. The physical schema is the Oracle Storage Cloud Service directory containing all the files. You can create new ODI Data store that will represent a file in Oracle Storage Cloud Service, so that it can be used in mappings.

Create an Oracle Storage Cloud Service model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Reverse Engineering an Oracle Storage Cloud Service Model

Oracle Data Integrator provides specific methods for reverse-engineering Oracle Storage Cloud Service files.

Oracle Storage Cloud Service supports the following types of reverse engineering:

- [Reverse-Engineering Delimited Files from Oracle Storage Cloud Service](#)
- [Reverse-engineering Fixed Files from Oracle Storage Cloud Service](#)
- [Reverse-Engineering JSON, Avro and Parquet Storage Formats](#)

Reverse-Engineering Delimited Files from Oracle Storage Cloud Service

To perform a delimited file reverse engineering:

1. In the **Models** accordion, right click your **Storage Cloud Service Model** and select **New Data store**. The **Data Store Editor** opens.
2. In the **Definition** tab, enter the following fields:
 - **Name**: Name of this data store
 - **Resource Name**: Sub-directory (if needed) and name of the file. It lists all the files present in Oracle Storage Cloud Service for the configured bucket.
3. Go to the **Storage** tab, to describe the type of file. Set the fields as follows:
 - **File Format**: Delimited
 - **Heading (Number of Lines)**: Enter the number of lines of the header. Note that if there is a header, Oracle Data Integrator uses the first line of the header 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 **Data Store Editor**, go to the **Attributes** tab.
6. In the editor toolbar, click **Reverse Engineer**.
7. Verify the data type and length for the reverse engineered attributes. Oracle Data Integrator infers the field data types 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**.

Reverse-engineering Fixed Files from Oracle Storage Cloud Service

Oracle Data Integrator provides a graphic wizard to define the columns of a fixed file. To reverse-engineer a fixed file from Oracle Storage Cloud Service using the wizard:

1. In the **Models** accordion, right click your **Object Storage Cloud Service** and select **New Data store**. The **Data store Editor** opens.
2. In the **Definition** Tab, enter the following fields:
 - **Name**: Name of this data store
 - **Resource Name**: Sub-directory (if needed) and name of the file. It lists all the files present in Oracle Object Storage for the configured bucket.
3. Go to the **Storage** 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 Data store Editor, go to the **Attributes** tab.
6. In the editor toolbar, click **Reverse Engineer**. The **Attributes Setup Wizard** appears. The **Attributes Setup Wizard** displays the first records of your file.
7. Click on the ruler (above the file contents) to create markers delimiting the attributes. You can right-click within the ruler to delete a marker.
8. Attributes are created with pre-generated names (C1, C2, and so on). You can edit the attribute name by clicking in the attribute header line (below the ruler).
9. In the properties panel (on the right), you can edit all the parameters of the selected attribute. You should set at least the Attribute Name, Data type, and Length for each attribute.
10. Click **OK**, when the attributes definition is complete.
11. From the **File** main menu, select **Save**.

Reverse-Engineering JSON, Avro and Parquet Storage Formats

Oracle Storage Cloud Service technology supports reverse engineering JSON, Avro and Parquet storage formats , attributes, data types, and data type properties. If one of these storage format types is selected, then reverse engineering is based on the Schema File specified, and not on a sample data file. The schema file should be accessible in local file system.

To reverse-engineer JSON, Avro and Parquet storage formats, perform the following steps:

1. In the Models accordion, right click your Oracle Storage Cloud Service Model and select **New Data store**. The **Data Store Editor** opens.
2. In the **Definition** tab, enter the following fields:
 - **Name**: Name of this data store
 - **Resource Name**: Click the Search icon, to select the required file from the list of files present in Oracle Storage Cloud Service for the configured bucket.
3. From the **Storage** Tab, select the Storage Format from the **Storage Format** drop-down list and specify the complete path of the schema file in the **Schema File** field.

The schema file should be located in the local file system.

4. From the File main menu, select **Save**.

 **Note:**

There is no need to import an RKM into the project.

Working with Oracle Storage Cloud Service Tools

You can upload and download files to or from Oracle Storage Cloud Service through the following tools:

- [Uploading Files/Objects to Oracle Storage Cloud Service](#)
- [Downloading File/Objects from Oracle Storage Cloud Service](#)

 **Note:**

Apart from ODI Studio, you can also work with ODI Storage Cloud Service Tools from command line.

Uploading Files/Objects to Oracle Storage Cloud Service

ODI Storage Cloud Service Upload tool is used to upload single, multiple files, or an entire directory from HDFS or a local file system on to Oracle Storage Cloud Service.

To upload file(s) or directories to Oracle Storage Cloud Service,

1. Create a new **Project**.

For more details on how to create a project, see [Creating an Integration Project of Developing Integration Projects with Oracle Data Integrator](#).

2. Below the created Project folder, create a **Package**.

For more details on how to create a package, see [Creating and Using Packages of Developing Integration Projects with Oracle Data Integrator](#).

3. Select **OdiStorageCSUpload** tool available in the Toolbox. Add it to the created package.

 **Note:**

All the parameters of the tool are displayed under **General** Tab.

Configure the required parameters:

Table 31-1 ODI Storage Cloud Service Upload Tool Parameters

Parameter	Description
Target Logical schema	Target Logical schema name of Oracle Storage Cloud Service instance. Container information is obtained from Logical schema through configured physical schema.
Source Logical schema	Name of the Source Logical schema configured for File or HDFS Data Server for upload of Local or HDFS Files to Oracle Storage Cloud Service. Directory structure is obtained from Logical schema through configured physical architecture.
File Names filter	Field to specify one or more files or directories to be uploaded to Oracle Storage Cloud Service recursively. It also supports the list of files separated by as a delimiter. The pattern followed is: <ul style="list-style-type: none"> • *.txt - should upload all the files ending with .txt • test* - uploads all the files and directories that matches with prefix "test" • *test* - uploads all the files and directories having substring "test" • test.xml test1.xml test2.xml - Uploads all the files specified • test* test1* - Uploads all the files matching pattern test* and test1* • test.xml - Only one file is uploaded
Overwrite	This parameter indicates if upload operation should overwrite an existing file or not. Default value for this parameter is No .
Retry on error	It represents the number of times the retry attempt should occur when a failure or error happens during upload.
Retry interval seconds	Retry interval indicates after how many seconds a retry attempt should happen.
Encrypt Key	This is the user provided key used for encrypting objects while uploading files or directories to Oracle Storage Cloud Service.

 **Note:**

This parameter cannot be null, if you want to encrypt objects while upload.

For more details on the above parameters, refer to OdiStorageCSUpload Tool in Oracle Data Integrator Tools Reference guide.

4. Save and execute the package.

The required files from the source directory are uploaded to the target container of the Oracle Storage Cloud Service.

5. Upon successful upload, you can find a complete log of this upload operation at the **Details** tab. To get to the details tab, from the Operator tab, expand the associated session for the upload tool and open the Session task window to find the Details tab with the required log information.

The details include:

- Source directory is : <source directory path>
- Target container is : <Storage container name>
- Filter used is : <input filter>
- Number of file uploaded:<Total number of files that were uploaded>
- Uploaded files are: <File1, File2>
- Number of files failed:<Total number of files that were not uploaded>
- Failed files are:<File1, File2>

Downloading File/Objects from Oracle Storage Cloud Service

ODI Storage Cloud Service Download tool is used to download single, multiple files, or an entire directory to HDFS or a local file system from Oracle Storage Cloud Service. For HDFS files, the files from Oracle Storage Cloud Service are first copied to the local directory (as you specified in Directory (Work Schema) for Oracle Storage Cloud Service Physical Schema) and then from local directory, files are downloaded to HDFS.

To download file(s) or directories from Oracle Storage Cloud Service,

- Create a new **Project**
For more details on how to create a project, see [Creating an Integration Project of Developing Integration Projects with Oracle Data Integrator](#).
- Below the created Project folder, create a **Package**
For more details on how to create a package, see [Creating and Using Packages of Developing Integration Projects with Oracle Data Integrator](#).
- Select **OdiStorageCSDownload** tool available in the Toolbox. Add it to the created Package.

Note:

All the parameters of the tool are displayed under **General** Tab.


- Configure the required parameters:

Table 31-2 ODI Storge CS Download Tool Parameters

Parameter	Description
Source Logical schema	Source Logical Schema name configured for Oracle Storage Cloud Service instance. Container information is obtained from Logical schema through configured physical schema.
Target Logical schema	Logical Schema name configured for File or HDFS Data Server for download of Local or HDFS Files from Oracle Storage Cloud Service. Directory structure is obtained from Logical schema through configured physical architecture.
File Names filter	Field to specify one or more files or directories to be downloaded from Oracle Storage Cloud Service recursively. It also supports delimiter for separated files list. The pattern followed is: <ul style="list-style-type: none"> – *.txt - should download all files ending with .txt – test* - Downloads all the files and directories that matches with prefix "test" – *test* - Downloads all the files and directories having substring "test" – test.xml test1.xml test2.xml - Downloads all the files specified – test* test1* - Downloads all the files matching pattern test* and test1* – test.xml - Only one file is downloaded
Overwrite	This parameter indicates, if download operation should overwrite an existing file or not. The default value for this parameter is No .
Retry on error	It represents the number of times the retry attempt should occur when a failure or error happens during download.
Retry interval seconds	Retry interval indicates after how many seconds a retry attempt should happen.

Table 31-2 (Cont.) ODI Storage CS Download Tool Parameters

Parameter	Description
Decrypt Key	This is the user provided key used for decrypting objects while downloading from Oracle Storage Cloud Service. This key should be same as the encrypt key provided during the upload of the same file (that you had uploaded earlier) to Oracle Storage Cloud Service. If you provide the wrong key then the download operation fails.

 **Note:**
This parameter cannot be null, if you want to decrypt objects while download.

For more details on the above parameters, refer to OdiStorageCSDownload Tool Oracle Data Integrator Tools Reference.

- Save and execute the package.

The required files from Oracle Storage Cloud Service are downloaded to the directory specified in the Target logical schema.

- Upon successful download, you can find a detailed log of this download operation at the **Details** tab. To get to the details tab, from the Operator tab, expand the associated session for the download tool and open the Session task window to find the Details tab with the required log information.

The details include:

- Source container is : <Storage container name>
- Target directory is : <target directory path>
- Filter used is : <input filter>
- Number of file downloaded:<Total number of files that were downloaded>
- Downloaded files are:<File1, File2>
- Number of files failed:<Total number of files that were not downloaded>
- Failed files are:< File1, File2>

Part IV

SaaS Applications

It is important to understand how to work with SaaS Applications in Oracle Data Integrator.

Part IV contains the following chapters:

- [Oracle Enterprise Resource Planning Cloud](#)
- [Oracle Marketing Cloud](#)
- [Oracle Sales Cloud](#)
- [Oracle Service Cloud](#)

Oracle Enterprise Resource Planning Cloud

It is important to understand how to work with Oracle Enterprise Resource Planning (ERP) Cloud in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Prerequisites](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Oracle ERP Cloud Datastore](#)
- [Designing a Mapping](#)
- [Troubleshooting](#)

Introduction

Oracle Enterprise Resource Planning (ERP) Cloud is a suite of cloud applications for finance, project management, procurement, risk management and other core day-to-day activities important in every business, regardless of size, industry or geography.

Oracle Data Integrator (ODI) seamlessly integrates with Oracle Enterprise Resource Planning (ERP) Cloud. Oracle Data Integrator features are designed to work best with ERP Cloud, including reverse-engineering and mappings.

Concepts

The Oracle ERP Cloud technology concepts map the Oracle Data Integrator concepts as follows: An Oracle ERP Cloud Instance corresponds to a data server in Oracle Data Integrator. Within this instance, a schema maps to an Oracle Data Integrator physical schema.

Knowledge Modules

Oracle Data Integrator provides the following Knowledge Modules (KMs) for handling Oracle ERP Cloud data.

Table 32-1 Oracle ERP Cloud Knowledge Modules

Knowledge Module	Description
LKM Oracle ERP Cloud to SQL	Extracts data from an existing BI Publisher report and inserts it into a staging table.

Table 32-1 (Cont.) Oracle ERP Cloud Knowledge Modules

Knowledge Module	Description
LKM Oracle ERP Cloud to File Direct	Extracts data from an existing BI Publisher report and inserts it into a file.

Prerequisites

The following prerequisites are essential for working with the Oracle ERP Cloud technology.

Make sure you go through the following prerequisites before working with Oracle ERP Cloud:

- Create a BI Publisher data model. For more details on BI Publisher data models, refer to [Using the Data Model Editor](#) section of *Data Modeling Guide for Oracle Business Intelligence Publisher*.
- Create a BI Publisher report. For more details on BI Publisher reports, refer [Using Oracle BI Publisher to Extract Data From Oracle Sales and ERP Clouds](#).
- Export the BI Publisher report file to the schema directory defined in the Oracle ERP Cloud physical schema. For more details on Oracle ERP Cloud physical schema, see [Creating an Oracle ERP Cloud Physical Schema](#).

Installation and Configuration

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

- [System Requirements and Certifications](#)
- [Technology Specific Requirements](#)
- [Connectivity Requirements](#)

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 Technology Network (OTN):

<http://www.oracle.com/technetwork/middleware/data-integrator/documentation/index.html>

Technology Specific Requirements

There are no technology-specific requirements for using Oracle ERP Cloud in Oracle Data Integrator.

Connectivity Requirements

There are no specific connectivity requirements for using Oracle ERP Cloud in Oracle Data Integrator.

Setting up the Topology

Setting up the topology consists of:

- [Creating an Oracle ERP Cloud Data Server](#)
- [Creating an Oracle ERP Cloud Physical Schema](#)

Creating an Oracle ERP Cloud Data Server

Create a data server for the Oracle ERP Cloud technology using the standard procedure, as described in *Creating a Data Server* of *Administering Oracle Data Integrator*. This section details only the fields required or specific for defining an Oracle ERP Cloud data server:

1. In the **Definition** tab:
 - a. **Name** : Enter a name for the data server definition
 - b. **WSDL URL** : Enter the BI Publisher web service used for the Oracle ERP Cloud instance.

You can specify either of the following BI Publisher web services in the WSDL URL field:

- **ReportService**: Provides methods to interact with BI Publisher Report object, such as to run reports, get information about reports, define and modify reports, and upload report templates. For more details on ReportService, refer to [ReportService](#) section of *Developer's Guide for Oracle Business Intelligence Publisher*.
- **ScheduleService**: Provides methods for executing scheduler tasks, such as to schedule report jobs, retrieve report outputs, and manage report history. For more details on ScheduleService, refer to [ScheduleService](#) section of *Developer's Guide for Oracle Business Intelligence Publisher*.

WSDL URL examples:

- `https://efsdcr12pt05.fs.efops.oraclecorp.com/xmlpserver/services/PublicReportWSSService?wsdl`
 - `https://efsdcr12pt05.fs.efops.oraclecorp.com/xmlpserver/services/ScheduleReportWSSService?wsdl`
2. Under **Connection**, enter a user name and password for connecting to the Oracle ERP Cloud instance.

Creating an Oracle ERP Cloud Physical Schema

Create a physical schema for the Oracle ERP Cloud data server using the standard procedure, as described in *Creating a Physical Schema* in *Administering Oracle Data*

Integrator. This section details only the fields required or specific for defining an Oracle ERP Cloud physical schema:

1. In the **Definition** tab:
 - a. **Directory (Schema)**: Directory where the BI Publisher report is placed after extract. This location is also required for reverse engineering. The ODI Agent needs access to this location to perform the mapping, while ODI Studio requires it to perform reverse engineering. If it is not possible for this location to be shared by the ODI Agent and ODI Studio, then you will need to set up two separate physical schemas and have a reverse engineering context and a runtime context.
 - b. **Directory (Work Schema)**: Directory where log files and temporary files are located (for example, responses from SOAP requests). The ODI Agent needs access to this location.
2. Check the **Default** box if you want this schema to be the default one for this data server (The first physical schema is always the default one).

Create for this physical schema a logical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator* and associate it in a given context.

Creating and Reverse-Engineering an Oracle ERP Cloud Datastore

This section contains the following topics:

- [Creating an Oracle ERP Cloud Model](#)
- [Creating an Oracle ERP Cloud Datastore](#)
- [Reverse-Engineering an Oracle ERP Cloud Datastore](#)

Creating an Oracle ERP Cloud Model

Create an Oracle ERP Cloud model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Creating an Oracle ERP Cloud Datastore

Create a datastore for the Oracle ERP Cloud technology using the standard procedure, as described in [Creating a Datastore of Developing Integration Projects with Oracle Data Integrator](#). This section details only the fields required or specific for defining an Oracle ERP Cloud datastore:

1. In the **Definition** tab:
 - a. **Resource Name**: Name of the BI Publisher report file (whose output format is `.csv`) which is used for reverse-engineering. This name will also serve as the name of the extract file (downloaded from UCM).
2. Select **Delimited** as the **Storage Format** in the **Storage** tab.
3. In the **Properties** tab:

- a. Enter **BIPReportLocation** in the **Key** field.
- b. Enter the location of the BI Publisher report file in the **Value** field corresponding to the key.

 **Note:**

The value for **BIPReportLocation** must not be empty. This refers to the location of the BI publisher report on the BI server, and can be found on the BI server once the corresponding report is open.

Defining Parameters for BI Publisher Report

The BI Publisher report can have various parameters to restrict the data coming into the report. You can define all the parameters that are required in the format of a key/value pair in the **Properties** tab. The name must match the parameter name defined in the BI Publisher report. The value can be either a real value (for example, "PRIMARY") or an ODI variable (for example, "#PROJ_ERP.PARAMS1").

Reverse-Engineering an Oracle ERP Cloud Datastore

Oracle ERP Cloud supports delimited file reverse-engineering. To perform a delimited file reverse-engineering:

1. In the Datastore Editor, go to the **Attributes** tab.
2. In the editor toolbar, click **Reverse Engineer**.
3. Verify the data type and length for the reverse-engineered attributes. Oracle Data Integrator infers the field's data types and length from the first record of the file, but may set default values (for example, 50 for the string field length) or incorrect data types in this process. In case of an empty field, data type is set to String with length 50.

Attributes are created with pre-generated names (C1, C2 and so on) if the file has no header and it is the first non-header record.

4. Select **Save** from the File main menu.

Designing a Mapping

You can use Oracle ERP Cloud as a source of a mapping.

The KM choice for a mapping determines the abilities and performance of this mapping. The recommendations in this section help in the selection of the KM for different situations concerning an Oracle ERP Cloud server.

Loading Data from Oracle ERP Cloud

Oracle ERP Cloud can be used as a source of a mapping. The LKM choice in the Mapping's Loading Knowledge Module tab to load data between Oracle ERP Cloud and another type of data server is essential for the performance of a mapping.

Use the knowledge modules listed in the below table to load data from an Oracle ERP Cloud server to a target or staging area database.

Table 32-2 KMs for loading data from Oracle ERP Cloud

Staging Area/Target Technology	KM	Notes
SQL	LKM Oracle ERP Cloud to SQL	<p>Extracts data from an existing BI Publisher report and inserts it into a staging table, where data can be loaded into any target using an IKM. The LKM will not return until the BI Publisher job is finished and data is loaded into the staging table. This LKM only supports BI Publisher reports that output in CSV format.</p> <p>The DELETE_TEMPORARY_OBJECTS LKM option should be set to Yes. This option is set in order to delete temporary objects at the end of the mapping, including the staging table and all the response files from SOAP requests.</p>
File	LKM Oracle ERP Cloud to File Direct	<p>Extracts data from an existing BI Publisher report and inserts it into a file. The LKM will not return until the BI Publisher job is finished and data is loaded into the file. This LKM only supports BI Publisher reports that output in CSV format.</p> <p>The DELETE_TEMPORARY_OBJECTS LKM option should be set to Yes. This option is set in order to delete temporary objects at the end of the mapping, including the staging table and all the response files from SOAP requests.</p>

Remote Agent Configuration

Data extraction can be done on a remote agent. To perform data extraction on a remote agent:

1. Create the physical and logical agent using the standard procedure, as described in [Creating a Physical Agent](#) of *Administering Oracle Data Integrator*.
2. Edit the `ODI_HOME/user_projects/domains/base_domain/config/fmwconfig/components/ODI/<Agent_Name>/bin/instance.cmd(.sh)` configuration file as follows:

- **No proxy:** Comment the following code:

```
#if [ ! -z $IS_AGENT_SCRIPT ] ; then
# ODI_SSL_PROPERTIES="-Djavax.net.ssl.trustStore=${WL_HOME}/
server/lib/DemoTru
st.jks -Djavax.net.ssl.keyStore=${DOMAIN_HOME}/security/
DemoIdentity.jks"
#else
#ODI_SSL_PROPERTIES="-Djavax.net.ssl.trustStore=${WL_HOME}/
server/lib/DemoTrust
.jks -
Djavax.net.ssl.trustStorePassword=DemoTrustKeyStorePassPhrase"
#fi
```

- **With proxy:** Set ODI_INSTANCE_JAVA_OPTIONS as follows:

```
ODI_INSTANCE_JAVA_OPTIONS="$ODI_ADDITIONAL_JAVA_OPTIONS $ODI_SSL_PRO
PERTIES
-Doracle.odi.standalone.agent.useauthenticator=false
-Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80
-Dhttps.proxyHost=www-proxy.us.oracle.com -Dhttps.proxyPort=80
-Dhttp.nonProxyHosts=localhost|127.0.0.0/8|localhost.localdomain|
127.0.0.1|::1
|adc01j1l.us.oracle.com|adc01j1l.us.oracle.com|10.229.118.112"
```

Troubleshooting

This section provides information on how to troubleshoot problems that you might encounter when using the Oracle ERP Cloud technology in Oracle Data Integrator.

Please find below the most common problems and the ways to resolve them:

- Run the BI Publisher report as standalone from the Fusion Apps BI Console and ensure that it executes properly.
- Execute the BI Publisher report using a SOAP client such as SoapUI to ensure that it can be called successfully from an external client.
- Review log files in Fusion Console to ensure successful completion or find any errors.
- Ensure that the path of the BI Publisher report is provided correctly, as this is the most common error.
- Advanced errors need to be tracked by Fusion System Administrators to look into BI server error logs.

Oracle Marketing Cloud

It is important to understand how to work with Oracle Marketing Cloud in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Oracle Marketing Cloud Model](#)
- [Designing a Mapping](#)

Introduction

Oracle Marketing Cloud simplifies digital marketing by providing one place for marketing teams to connect data, orchestrate experiences, and optimize interactions for each individual customer.

Oracle Data Integrator (ODI) seamlessly integrates with Oracle Marketing Cloud. Oracle Data Integrator features are designed to work best with Oracle Marketing Cloud, including reverse-engineering and mappings.

Concepts

The Oracle Marketing Cloud technology concepts map the Oracle Data Integrator concepts as follows: An Oracle Marketing Cloud Instance corresponds to a data server in Oracle Data Integrator. Within this instance, a schema maps to an Oracle Data Integrator physical schema.

Knowledge Modules

Oracle Data Integrator provides no Knowledge Module (KM) specific to the Oracle Marketing Cloud technology. You can use the generic SQL KMs to perform the data integration and transformation operations of Oracle Marketing Cloud data. See [Generic SQL](#) for more information.

Installation and Configuration

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

- [System Requirements and Certifications](#)
- [Technology Specific Requirements](#)
- [Connectivity Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>

Technology Specific Requirements

There are no technology-specific requirements for using Oracle Marketing Cloud in Oracle Data Integrator.

Connectivity Requirements

This section lists the requirements for connecting to the Oracle Marketing Cloud database.

Oracle Marketing Cloud (Eloqua) JDBC Driver

Oracle Data Integrator uses the Oracle Marketing Cloud (Eloqua) JDBC Driver to connect to the Oracle Marketing Cloud database.

Setting up the Topology

Setting up the topology consists of:

- [Creating an Oracle Marketing Cloud Data Server](#)
- [Creating an Oracle Marketing Cloud Physical Schema](#)

Creating an Oracle Marketing Cloud Data Server

Create a data server for the Oracle Marketing Cloud technology using the standard procedure, as described in *Creating a Data Server* of *Administering Oracle Data Integrator*. This section details only the fields required or specific for defining an Oracle Marketing Cloud data server:

1. Duplicate the Oracle technology with the name "Oracle Marketing (Eloqua)".
2. In the **Definition** tab, enter the following fields:
 - a. **Name** : Name of the data server that will appear in Oracle Data Integrator.
 - b. **User**: User name for connecting to the data server.
 - c. **Password**: Password for connecting to the data server.
3. In the **JDBC** tab, enter the following values:
 - a. **JDBC Driver**: `weblogic.jdbc.eloqua.EloquaDriver`

- b. JDBC URL** : jdbc:weblogic:eloqua://
login.elqqa01.com;Company=PaasDi;User=<user>;Password=<password>;
4. Click **Test Connection**, to test the established connection.
5. Delete all the data types under Oracle Marketing (Eloqua)/Data Types node in Topology Navigator.
6. Reverse-engineer the Oracle Marketing Cloud (Eloqua) data types by right-clicking **Oracle Marketing (Eloqua)** and selecting **Datatypes Reverse-Engineering** from the popup menu. In the Reverse Engineer Datatypes dialog, select the Data Server name from the drop-down menu.

Creating an Oracle Marketing Cloud Physical Schema

Create a physical schema for the Oracle Marketing Cloud data server using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*.

Create a logical schema for this physical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator*, and associate it in a given context.

Creating and Reverse-Engineering an Oracle Marketing Cloud Model

This section contains the following topics:

- [Creating an Oracle Marketing Cloud Model](#)
- [Reverse-engineer an Oracle Marketing Cloud Model](#)

Creating an Oracle Marketing Cloud Model

Create an Oracle Marketing Cloud model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Reverse-engineer an Oracle Marketing Cloud Model

Oracle Marketing Cloud supports **Standard** reverse-engineering - which uses only the abilities of the JDBC driver.

To perform a Standard reverse-engineering on an Oracle Marketing Cloud model, use the usual procedure, as described in *Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator*.

Designing a Mapping

You can use Oracle Marketing Cloud as a source or a target of a mapping. The KM choice for a mapping determines the abilities and performance of this mapping.

Oracle Data Integrator does not provide specific knowledge modules for Oracle Marketing Cloud. Use the [Generic SQL](#) KMs or the KMs specific to the technology used as the staging area.

34

Oracle Sales Cloud

It is important to understand how to work with Oracle Sales Cloud in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Oracle Sales Cloud Model](#)
- [Designing a Mapping](#)

Introduction

Oracle Sales Cloud delivers high-value, industry-specific sales automation and sales performance management solutions.

Oracle Data Integrator (ODI) seamlessly integrates with Oracle Sales Cloud. Oracle Data Integrator features are designed to work best with Oracle Sales Cloud, including reverse-engineering and mappings.

Concepts

The Oracle Sales Cloud technology concepts map the Oracle Data Integrator concepts as follows: An Oracle Sales Cloud Instance corresponds to a data server in Oracle Data Integrator. Within this instance, a schema maps to an Oracle Data Integrator physical schema.

Knowledge Modules

Oracle Data Integrator provides no Knowledge Module (KM) specific to the Oracle Sales Cloud technology. You can use the generic SQL KMs to perform the data integration and transformation operations of Oracle Sales Cloud data. See [Generic SQL](#) for more information.

Installation and Configuration

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

- [System Requirements and Certifications](#)
- [Technology Specific Requirements](#)
- [Connectivity Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>

Technology Specific Requirements

There are no technology-specific requirements for using Oracle Sales Cloud in Oracle Data Integrator.

Connectivity Requirements

This section lists the requirements for connecting to the Oracle Sales Cloud database.

Oracle Sales Cloud JDBC Driver

Oracle Data Integrator uses the Oracle Sales Cloud JDBC Driver to connect to the Oracle Sales Cloud database.

Setting up the Topology

Setting up the topology consists of:

- [Creating an Oracle Sales Cloud Data Server](#)
- [Creating an Oracle Sales Cloud Physical Schema](#)

Creating an Oracle Sales Cloud Data Server

Create a data server for the Oracle Sales Cloud technology using the standard procedure, as described in *Creating a Data Server of Administering Oracle Data Integrator*. This section details only the fields required or specific for defining an Oracle Sales Cloud data server:

1. Duplicate the Oracle technology with the name "Oracle Sales Cloud".
2. In the **Definition** tab, enter the following fields:
 - a. **Name** : Name of the data server that will appear in Oracle Data Integrator.
 - b. **User**: User name for connecting to the data server.
 - c. **Password**: Password for connecting to the data server.
3. In the **JDBC** tab, enter the following values:
 - a. **JDBC Driver**: `weblogic.jdbc.oraclesalescloud.OracleSalesCloudDriver`
 - b. **JDBC URL** : `jdbc:weblogic:oraclesalescloud://<host>;User=<user name>;Password=<password>`

4. Click **Test Connection**, to test the established connection.
5. Delete all the data types under Oracle Sales Cloud/Data Types node in Topology Navigator.
6. Reverse-engineer the Oracle Sales Cloud data types by right-clicking **Oracle Sales Cloud** and selecting **Datatypes Reverse-Engineering** from the pop-up menu. In the Reverse Engineer Datatypes dialog, select the Data Server name from the drop-down menu.

Creating an Oracle Sales Cloud Physical Schema

Create a physical schema for the Oracle Sales Cloud data server using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*.

Create a logical schema for this physical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator*, and associate it in a given context.

Creating and Reverse-Engineering an Oracle Sales Cloud Model

This section contains the following topics:

- [Creating an Oracle Sales Cloud Model](#)
- [Reverse-engineer an Oracle Sales Cloud Model](#)

Creating an Oracle Sales Cloud Model

Create an Oracle Sales Cloud model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Reverse-engineer an Oracle Sales Cloud Model

Oracle Sales Cloud supports **Standard** reverse-engineering - which uses only the abilities of the JDBC driver.

To perform a Standard reverse-engineering on an Oracle Sales Cloud model, use the usual procedure, as described in *Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator*.

Designing a Mapping

You can use Oracle Sales Cloud as a source or a target of a mapping. The KM choice for a mapping determines the abilities and performance of this mapping.

Oracle Data Integrator does not provide specific knowledge modules for Oracle Sales Cloud. Use the [Generic SQL](#) KMs or the KMs specific to the technology used as the staging area.

Oracle Service Cloud

It is important to understand how to work with Oracle Service Cloud in Oracle Data Integrator.

This chapter includes the following sections:

- [Introduction](#)
- [Installation and Configuration](#)
- [Setting up the Topology](#)
- [Creating and Reverse-Engineering an Oracle Service Cloud Model](#)
- [Designing a Mapping](#)

Introduction

Oracle Service Cloud delivers comprehensive customer experience applications that drive revenue, increase efficiency, and build loyalty.

Oracle Data Integrator (ODI) seamlessly integrates with Oracle Service Cloud. Oracle Data Integrator features are designed to work best with Oracle Service Cloud, including reverse-engineering and mappings.

Concepts

The Oracle Service Cloud technology concepts map the Oracle Data Integrator concepts as follows: An Oracle Service Cloud Instance corresponds to a data server in Oracle Data Integrator. Within this instance, a schema maps to an Oracle Data Integrator physical schema.

Knowledge Modules

Oracle Data Integrator provides no Knowledge Module (KM) specific to the Oracle Service Cloud technology. You can use the generic SQL KMs to perform the data integration and transformation operations of Oracle Service Cloud data. See [Generic SQL](#) for more information.

Installation and Configuration

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

- [System Requirements and Certifications](#)
- [Technology Specific Requirements](#)
- [Connectivity Requirements](#)

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/technetwork/middleware/data-integrator/documentation/index.html>

Technology Specific Requirements

There are no technology-specific requirements for using Oracle Service Cloud in Oracle Data Integrator.

Connectivity Requirements

This section lists the requirements for connecting to the Oracle Service Cloud database.

Oracle Service Cloud JDBC Driver

Oracle Data Integrator uses the Oracle Service Cloud JDBC Driver to connect to the Oracle Service Cloud database.

Setting up the Topology

Setting up the topology consists of:

- [Creating an Oracle Service Cloud Data Server](#)
- [Creating an Oracle Service Cloud Physical Schema](#)

Creating an Oracle Service Cloud Data Server

Create a data server for the Oracle Service Cloud technology using the standard procedure, as described in *Creating a Data Server of Administering Oracle Data Integrator*. This section details only the fields required or specific for defining an Oracle Service Cloud data server

1. Duplicate the Oracle technology with the name "Oracle Service Cloud".
2. In the **Definition** tab, enter the following fields:
 - a. **Name** : Name of the data server that will appear in Oracle Data Integrator.
 - b. **User**: User name for connecting to the data server.
 - c. **Password**: Password for connecting to the data server.
3. In the **JDBC** tab, enter the following values:
 - a. **JDBC Driver**:
`weblogic.jdbc.oracleservicecloud.OracleServiceCloudDriver`

- b. JDBC URL :**
`jdbc:weblogic:oraclservicecloud:LoginHost=<host>;interfacename=<interface name>;user=<user name>;password=<password>`
- 4.** Click **Test Connection**, to test the established connection.
- 5.** Delete all the data types under Oracle Service Cloud/Data Types node in Topology Navigator.
- 6.** Reverse-engineer the Oracle Service Cloud data types by right-clicking **Oracle Service Cloud** and selecting **Datatypes Reverse-Engineering** from the pop-up menu. In the Reverse Engineer Datatypes dialog, select the Data Server name from the drop-down menu.

Creating an Oracle Service Cloud Physical Schema

Create a physical schema for the Oracle Service Cloud data server using the standard procedure, as described in *Creating a Physical Schema in Administering Oracle Data Integrator*.

Create a logical schema for this physical schema using the standard procedure, as described in *Creating a Logical Schema in Administering Oracle Data Integrator*, and associate it in a given context.

Creating and Reverse-Engineering an Oracle Service Cloud Model

This section contains the following topics:

- [Creating an Oracle Service Cloud Model](#)
- [Reverse-engineer an Oracle Service Cloud Model](#)

Creating an Oracle Service Cloud Model

Create an Oracle Service Cloud model using the standard procedure, as described in *Creating a Model of Developing Integration Projects with Oracle Data Integrator*.

Reverse-engineer an Oracle Service Cloud Model

Oracle Service Cloud supports **Standard** reverse-engineering - which uses only the abilities of the JDBC driver.

To perform a Standard reverse-engineering on an Oracle Service Cloud model, use the usual procedure, as described in *Reverse-engineering a Model of Developing Integration Projects with Oracle Data Integrator*.

Designing a Mapping

You can use Oracle Service Cloud as a source or a target of a mapping. The KM choice for a mapping determines the abilities and performance of this mapping.

Oracle Data Integrator does not provide specific knowledge modules for Oracle Service Cloud. Use the [Generic SQL](#) KMs or the KMs specific to the technology used as the staging area.

Part V

Appendices

You can find out more information on the various drivers available for Oracle Data Integrator.

Part IV contains the following appendices:

- [Oracle Data Integrator Driver for LDAP Reference](#)
- [Oracle Data Integrator Driver for XML Reference](#)
- [Oracle Data Integrator Driver for Complex Files Reference](#)

A

Oracle Data Integrator Driver for LDAP Reference

The Oracle Data Integrator Driver for LDAP (LDAP driver) allows Oracle Data Integrator to manipulate complex LDAP trees using standard SQL queries. This appendix includes the following sections:

- [Introduction to Oracle Data Integrator Driver for LDAP](#)
- [LDAP Processing Overview](#)
- [Installation and Configuration](#)
- [SQL Syntax](#)
- [JDBC API Implemented Features](#)

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

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.

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](#)

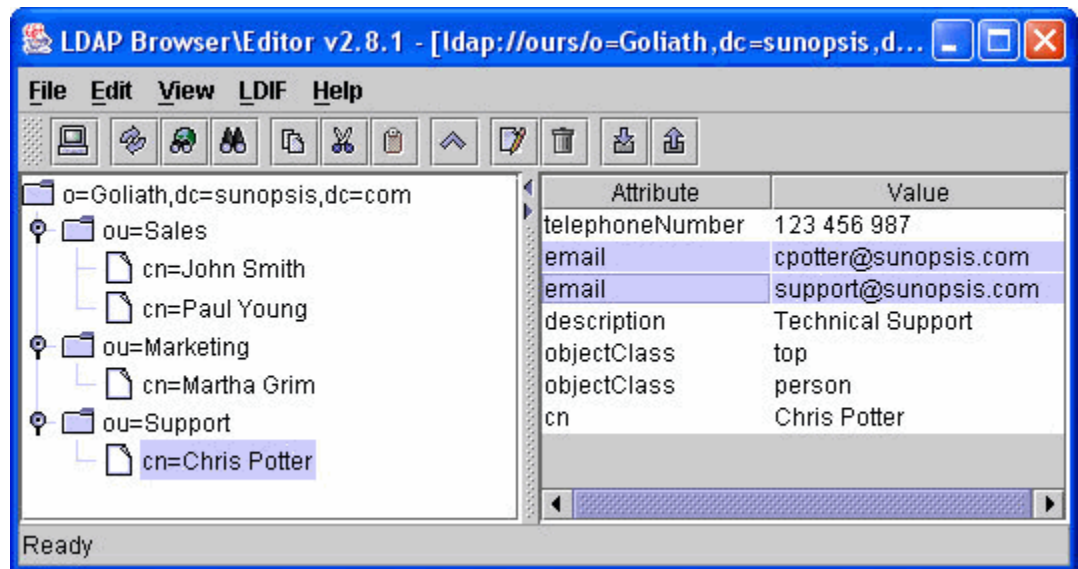
General Principle

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

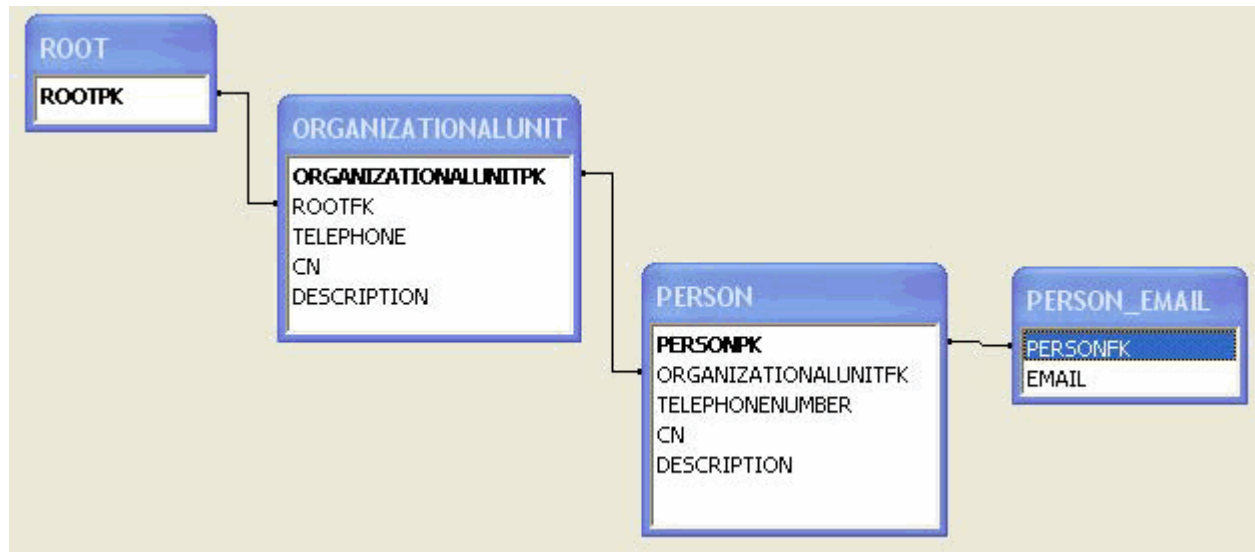


This LDAP tree will be mapped into the following relational structure:

- 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



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.

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_`.

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 `ldap_metadata` 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.

Managing Relational Schemas

This section contains the following topics:

- [Relational Schema Storage](#)
- [Accessing Data in the Relational Structure](#)

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.
- Access 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.

See [Using an External Database to Store the Data](#) for more information on how to set up external storage.

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.

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 an External Database to Store the Data](#)
- [LDAP Directory Connection Configuration](#)
- [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.

Driver Configuration

 **Note:**

ODI LDAP driver's support for LDAP servers is limited. All the features of the driver will work on any given instance of an LDAP server. ODI uses Java JNDI API to interact with the LDAP servers. If the LDAP server adheres exactly with LDAP specifications, then driver features will work. Otherwise, some of the features may not work.

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>[&...]`

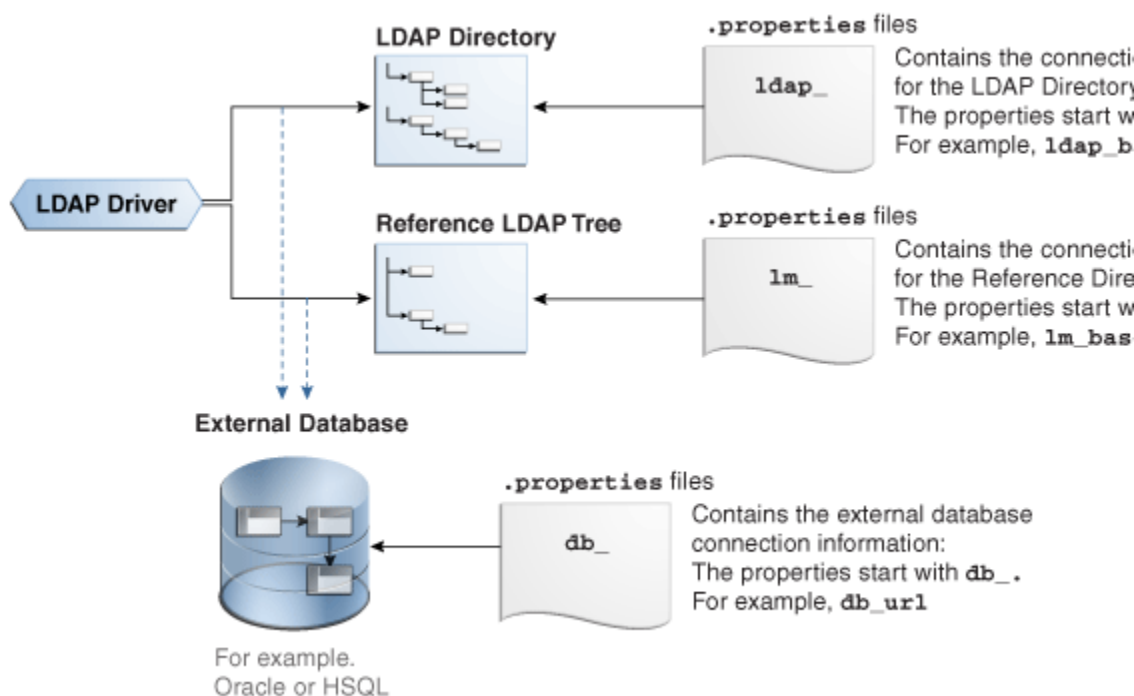
The first URL requires the LDAP directory password to be encoded. The second URL allows you to give the LDAP directory password without encoding it.

 **Note:**

It is recommended to use the first URL to secure the LDAP directory password.

The LDAP driver uses different properties depending on the established connection. [Figure A-3](#) shows when to use which properties.

Figure A-3 Properties Files for LDAP Driver



The LDAP driver connects to the LDAP directory. You can configure this connection with the properties that start with `ldap_`. For example, `ldap_basedn`. Instead of passing the LDAP directory properties in the driver URL, you can use a *properties file* for the configuration of the connection to the LDAP directory. This properties file must be specified in the `ldap_props` property of the driver URL.

If you want to use the hierarchical structure of the LDAP tree without the accompanying data volume, you can use the Reference LDAP tree. The connection to the Reference LDAP tree is configured with the properties that start with `lm_`. For example, `lm_basedn`. Instead of passing the `lm_` properties in the driver URL, you can use a properties file. This properties file must be specified in the `ldap_metadata` property of the driver URL. See [Reference LDAP Tree](#) for more information.

To configure the connection of the LDAP driver to an external database, use the properties that start with `db_`. For example, `db_url`. Instead of passing the external database properties in the driver URL, you can use a *properties file* for the configuration of the connection to the external database. This properties file must be specified in the `db_props` property of the driver URL. See [Using an External Database to Store the Data](#) for more information.

[Table A-1](#) describes the properties that can be passed in the driver URL.

Table A-1 URL Properties


Property	Mandatory	Type	Default	Description
db_props or dp	No	string (file location)	Empty string	<p>Name of a <code>.properties</code> file containing the external database connection configuration. See Using an External Database to Store the Data for the details of this file content.</p> <p>Note: This property should contain the name of the <code>.properties</code> file without the file extension.</p> <p>Note: This <code>.properties</code> file must be in the run-time agent classpath.</p> <p>Note: You can specify the external database connection configuration using all the <code>db_</code> properties listed below in this table.</p>
ldap_props or lp	No	string (file location)	N/A	<p>Name of a <code>.properties</code> file containing the directory connection configuration. See LDAP Directory Connection Configuration for the details of this file content.</p> <p>Note: This property should contain the name of the <code>.properties</code> file without the file extension.</p> <p>Note: This <code>.properties</code> file must be in the run-time agent classpath.</p> <p>Note: You can specify the LDAP directory connection configuration using all the <code>ldap_</code> properties listed below in this table.</p>
ldap_metadata or lm	No	string (file location)	N/A	<p>Name of a <code>.properties</code> file containing the directory connection configuration for the <i>Reference LDAP Tree</i>. See LDAP Directory Connection Configuration for the details of this file content, and Reference LDAP Tree for an explanation of the reference tree.</p> <p>Note: This property should contain the name of the <code>.properties</code> file without the file extension.</p> <p>Note: This <code>.properties</code> file must be in the run-time agent classpath.</p> <p>Note: You can specify the reference LDAP directory connection configuration using all the <code>lm_</code> properties listed below in this table.</p>
case_sensitive or cs	No	boolean (true false)	false	<p>Enable / disable case sensitive mode for both LDAP- and RDBMS-managed objects.</p>
alias_bundle or ab	No	string (file location)	Empty string	<p>Full name of a properties file including both the absolute path to the properties file and the file extension. The properties file is a file that contains the list of aliases for the LDAP to Relational Mapping. If this file does not exist, it will be created by the driver. See Table Aliases Configuration for more information.</p> <p>Note: The file extension does not need to be <code>.properties</code>.</p>
alias_bundle_encoding or abe	No	string (encoding code)	Default encoding	<p>Alias bundle file encoding. This encoding is used while reading and overwriting the <code>alias_bundle</code> file. If it is not defined then the default encoding would be used.</p> <p>You will find a list of supported encoding at the following URL: https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html.</p>

Table A-1 (Cont.) URL Properties

Property	Mandatory	Type	Default	Description
grouping_factor or gf	No	integer	2	Determines how many object classes will be grouped together to set up a single relational table mapping. See Grouping Factor for more information.
key_column or kc	No	boolean (true false)	false	If set to true, a technical column called <code>SNPSLDAPKEY</code> is created to store the Relative Distinguished Name (RDN) for each LDAP entry. See Mapping Exceptions for more information.
numeric_ids 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 types.
table_prefix or tp	No	string	N/A	Prefix added to relational tables of the current connection.
ldap_auth	No	string	simple	LDAP Directory authentication method. See the <code>auth</code> property in LDAP Directory Connection Configuration .
ldap_url	Yes	string	N/A	LDAP Directory URL. See the <code>url</code> property in LDAP Directory Connection Configuration .
ldap_user	No	string	Empty string	LDAP Directory user name. See the <code>user</code> property in LDAP Directory Connection Configuration .
ldap_password	No	string	Empty string	LDAP Directory user password. See the <code>password</code> property in LDAP Directory Connection Configuration .
ldap_basedn	No	string	N/A	LDAP Directory basedn. See the <code>basedn</code> property in LDAP Directory Connection Configuration .
lm_auth	No	string	simple	Reference LDAP authentication method. See the <code>auth</code> property in LDAP Directory Connection Configuration .
lm_url	Yes	string	N/A	Reference LDAP URL. See the <code>url</code> property in LDAP Directory Connection Configuration .
lm_user	No	string	Empty string	Reference LDAP Directory user name. See the <code>user</code> property in LDAP Directory Connection Configuration .
lm_password	No	string	Empty string	Reference LDAP Directory user password. See the <code>password</code> property in LDAP Directory Connection Configuration .
lm_basedn	No	string	N/A	Reference LDAP Directory basedn. See the <code>basedn</code> property in LDAP Directory Connection Configuration .
db_driver	Yes	string	N/A	External Database JDBC Driver. See the <code>driver</code> property in Using an External Database to Store the Data .
db_url	Yes	string	N/A	External Database JDBC URL. See the <code>url</code> property in Using an External Database to Store the Data .
db_user	No	string	Empty string	External Database user. See the <code>user</code> property in Using an External Database to Store the Data .
db_password	No	string	Empty string	External Database password. See the <code>password</code> property in Using an External Database to Store the Data .
db_schema	No	string	Empty string	External Database schema. See the <code>schema</code> property in Using an External Database to Store the Data .

Table A-1 (Cont.) URL Properties

Property	Mandatory	Type	Default	Description
db_catalog	No	string	Empty string	External Database catalog. See the <code>catalog</code> property in Using an External Database to Store the Data .
db_drop_on_disconnect or db_dod	No	boolean (true false)	true	Drop tables on disconnect on the external database. See the <code>drop_on_disconnect</code> property in Using an External Database to Store the Data .
db_load_mode or db_lm	No	string	ci	Loading method for the external database. See the <code>load_mode</code> property in Using an External Database to Store the Data .
page_size	No	integer	0	Read data from LDAP servers with this page size limit. Setting this property to a positive value will cause the LDAP driver to try to use pagination to retrieve all the results, in case the LDAP driver has enforced pagination on search results.
transform_non_ascii or tna	No	boolean (true false)	true	Transform Non Ascii. Set to false to keep non-ascii characters.

 **Note:**
The value set for `page_size` must match the maximum page size (maximum number of results) set on the LDAP server.

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=KPLEKFMJKCLFJMDFDGPGPDB&ldap_user=cn=orcladmin&ldap_basedn=ou=applications
```

Using an External Database to Store the Data

The relational structure resulting from the LDAP to relational mapping of the LDAP tree can be stored in the run-time agent's memory or in an external database.

Note:

The list of technologies that support external storage is available on Oracle Technical Network (OTN) :

http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

The external storage is configured with a set of properties described in [Table A-2](#).

The external storage properties can be passed in several ways:

- [Passing the Properties in the Driver URL](#)
- [Setting the Properties in ODI Studio](#)
- [Setting the Properties in a Properties File](#)

Passing the Properties in the Driver URL

The properties can be directly set in the driver URL. When using this method, the properties have to be prefixed with `db_`. For example, if connecting to an Oracle database, specify the Oracle JDBC driver name in the `driver` parameter as follows:

```
db_driver=oracle.jdbc.OracleDriver.
```

Setting the Properties in ODI Studio

The properties can be specified on the Properties tab of the Data Server editor in Topology Navigator. When using this method, the properties have to be prefixed with `db_`. For example, if you want to set the `driver` parameter:

1. In the **Key** column, enter `db_driver`
2. In the **Value** column, enter `oracle.jdbc.OracleDriver` if you are connecting to an Oracle database.

Setting the Properties in a Properties File

The properties can be set in an *external database properties file*. This properties file, also called *property bundle*, is a text file with the `.properties` extension containing a set of lines with on each line a `<property>=<value>` pair. This external database properties file contains the properties of a JDBC connection to the relational database schema. The properties file is referenced using the `db_props` property in the JDBC URL.

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.

When using this method, note the following:

- The properties in the properties file are not prefixed and used as described in [Table A-2](#).
- The `db_props` property is set to the name of the properties file without the `.properties` extension. For example, if you have in your classpath the `prod_directory.properties` file, you should refer to this file as follows:
`db_props=prod_directory`.

The `db_props` property indicates that the schema must be loaded in a database schema whose connection information is stored in a *external database properties file*.
- The properties files have to be deployed by the agent using the LDAP connection. The location the properties file depends on the agent you are using:
 - *Local agent (Studio)*: Place the external DB properties file in the `<user.dir>/odi/oracledi/userlib` folder
 - *Standalone Agent*: Place the external DB properties file in `oracledi/agent/drivers` folder
 - *JavaEE Agent*: The external DB properties file should be packed into a JAR or ZIP file and added to the template generated by the Java EE agent. See *Deploying an Agent in a Java EE Application Server (Oracle WebLogic Server)* in *Administering Oracle Data Integrator* for more information.
- 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.

Note:

When connecting to the external database, the LDAP 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 `db_`. For example:

```
jdbc:snps:ldap?ldap_url=ldap://localhost:389/
&ldap_basedn=o=company&db_driver=oracle.jdbc.OracleDriver&db_url=<external_db_url
>
```

The properties for configuring external storage are described in [Table A-2](#).

Table A-2 External Storage Configuration Properties

Property	Man data ry	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
password	No	string	Empty string	Encrypted database user password. Note: To encrypt the password, use the <code>encode.bat (cmd sh)</code> command. See <i>Encoding a Password in Administering Oracle Data Integrator</i> 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 drop	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 lmode	No	string	ci	The loading method. Values may be: <ul style="list-style-type: none"> 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.
unicode	No	boolean (true false)		For MS SQL Server: If unicode = true, nvarchar is used. If unicode = false or not set, varchar is used.
varchar_length or vlength	No	integer	255	Size of all the columns of the relational structure that will be used to contain string data.

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
```

LDAP Directory Connection Configuration

The Oracle Data Integrator driver for LDAP uses the properties described in [Table A-3](#) 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-3](#).

Table A-3 Directory Connection Properties

Property	Mandatory	Type	Default	Description
auth	No	string	simple	The authentication method
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 <i>keytool</i> 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 <code>encode.bat (cmd sh)</code> command. See <i>Encoding a Password in Administering Oracle Data Integrator</i> 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
```

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=C:/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 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.
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 [Reverse-Engineering an LDAP Model](#). Oracle Data Integrator will create datastores with the table names defined as aliases in the alias file.

Example A-1 Alias File

```
INETORGPERSO_N_ORGANIZATIONALPERSON_PERSON_BISOBJECT_MAIL = PERSONMAIL
ORGANIZATIONALUNIT_RFC822MAILMEMBER = ORG_228MAIL
INETORGPERSO_N_ORGANIZATIONALPERSON_PERSON = ORG_PERSON
ORGANIZATIONALUNIT_MEMBER = ORG_UN_MEMBER
ORGANIZATIONALUNIT = ORG_UNIT
ROOT = ROOT
....
```

 **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.

SQL Syntax

The SQL statements described in [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 Operations 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

SQL Statements

Any number of commands may be combined. The semicolon (;) may be used to separate each command but is not necessary.

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.

INSERT INTO

Insert one or more new rows of data into a table.

```
INSERT INTO <table_name> [ ( <column_name> [,...] ) ]
    { VALUES (<expression> [,...]) | <SELECT Statement> }
```

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 ]
```

UPDATE

Modifies data of a table in the database.

```
UPDATE table SET column = <expression> [, ...] [WHERE <expression>]
```

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).

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

Table A-5 (Cont.) Numeric Functions

Function	Description
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
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

Table A-6 (Cont.) String Functions

Function	Description
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
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

Table A-7 (Cont.) Date and Time Functions

Function	Description
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

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
Struct	2.0 Core	No
XAConnection	2.0 Optional	No
XADataSource	2.0 Optional	No

B

Oracle Data Integrator Driver for XML Reference

The Oracle Data Integrator Driver for XML (XML driver) allows Oracle Data Integrator to use XML documents as data servers.

This appendix includes the following sections:

- [Introduction to Oracle Data Integrator Driver for XML](#)
- [XML Processing Overview](#)
- [Installation and Configuration](#)
- [Detailed Driver Commands](#)
- [SQL Syntax](#)
- [JDBC API Implemented Features](#)
- [Rich Metadata](#)
- [XML Schema Supported Features](#)

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

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.

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. In case of XML recursion the parent element or ancestors of the parent element can be located in the same table.
- 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 record 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 reserved 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 reserved 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.

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.

 **Note:**

In v3 mode, the table names are not prefixed with `<namespace>_`.

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.

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 storageSee [Using an External Database to Store the Data](#) for more information.

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 [Detailed Driver Commands](#) for more information.

The *default schema* is a specific schema that is used for storing temporary data. The default schema is read-only and cannot be used to store XML files. It is recommended to create a schema for each XML file.

It is also possible to automatically create additional schemas with different XML structures when creating the connection to the driver. See [Driver Configuration](#) for more information.

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.

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.

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 [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 [SYNCHRONIZE](#).

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.

XML Schema (XSD) Support

XSD is supported by the XML driver for describing XML file structures. See [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.

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.

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`

The properties to be entered in Properties table are detailed in [Table B-1](#).

Table B-1 Driver Properties

Property	Default Value	Mandatory	Description
blank_attribute_as_column	False	No	If this property is set to true, any empty element in the XML file that does not have child element of its own is considered as a column rather than a table.
file	-	Yes	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	-	No	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 <code>.dtd</code> extension. A DTD file may be created from the XML file structure depending on the <code>generate_dtd</code> 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	-	No	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	False	No	Open the XML file in read only mode.

Table B-1 (Cont.) Driver Properties

Property	Default Value	Mandatory	Description
schema	-	No	<p>It is the database schema storing the relational schema and the XML data. 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.</p> <p>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.</p> <p>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.</p> <p>Important: The schema name should be specified in uppercase.</p> <p>Important: It is forbidden to have a schema name identical to an XML ELEMENT name.</p>
standalone	False	No	<p>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. The schema is restricted to this connection and only this one. Other connections cannot see this schema. This option is active only for In-Memory HSQL intermediate database. Using this option causes increased memory consumption by the agent, as for every staging schema, an entirely new HSQL instance is created in the in-memory. Useful for parallel jobs with the same topology in order to avoid that the jobs overlap each other. Note: This option is not applicable when an external database is used. If a data server has its 'standalone' property set to 'true,' then it cannot be used as a target data store(s) to store data, and then write it out. This is because of the complete isolation of 'standalone' instances.</p> <p>Note: The property 'standalone' can be used:</p> <ul style="list-style-type: none"> • When f= parameter is not present in the JDBC connection URL/properties. • And you are not trying to load the same file using 'LOAD FILE' command in parallel. <p>This is because, before reading a source file, it is locked by the driver, until all operations on it are done. Hence, if you have parallel sessions, the first session will lock the file, until the session is complete.</p> <p>Caution: If f= parameter is present, immediately on opening connection, the driver will lock the file and read from it. It will be unlocked only when all connections to the file are closed. There is an explicit 'UNLOCK' command, but use it with extreme caution after you are sure of what you are doing.</p>

Table B-1 (Cont.) Driver Properties

Property	Default Value	Mandatory	Description
ns_prefix_generation	auto	No	<p>This option defines how namespace prefixes are generated and written in the XML file.</p> <ul style="list-style-type: none"> • 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. <p>Note that the xsd option value assumes that a similar prefix is not used in several XSD files to reference a different namespace.</p>
no_default_ns	False	No	If this property is set to true, the driver generates the target file with no default namespace entry.
no_closing_tags	False	No	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	-	No	<p>This property is used to use an external database instead of the memory engine to store the schema.</p> <p>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.</p>
load_data_on_connect	True	No	<p>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.</p> <p>This option is useful to test the connection or browse metadata without loading all the data.</p>
drop_on_disconnect	False	No	<p>Drop automatically the schema when closing the JDBC connection.</p> <p>If true, the schema is stored in the built-in engine, it is always dropped.</p> <p>If true and the data is on an external database, only the current reference to the schema in memory will be dropped, but the tables will remain in the external database. This means that if you try to connect to this schema again, it will reuse the tables in the external database rather than starting from scratch (as it would when the data is loaded in memory).</p>
ignore_unknown_elements	True	No	Ignore all elements in the XML file that do not exist in the associated DTD (Document Type Definition) or XSD (XML Schema Definition) file.
useimplicitmaxvalue	False	No	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.

Table B-1 (Cont.) Driver Properties

Property	Default Value	Mandatory	Description
generate_dtd	auto	No	<p>Defines if a DTD file must be created from the XML file structure:</p> <ul style="list-style-type: none"> • 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. <p>Warning: DTD files created using this option contain only the definition of XML elements appearing in the XML file, and may not be complete.</p>
java_encoding	UTF8	No	<p>Target file encoding (for example: ISO8859_1). You will find a list of supported encoding at the following URL: https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html.</p> <p>Note that if the Java encoding is specified, the XML encoding should also be specified.</p>
useimplicitmaxvalue	False	No	<p>With this property set to yes, an elements for which maxOccurs is not specified in the XSD is considered as multivalued (maxOccurs="unbounded").</p>
xml_encoding	UTF8	No	<p>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://download.oracle.com/javase/6/docs/technotes/guides/intl/encoding.doc.html.</p> <p>Note that if the XML encoding is specified, the Java encoding should also be specified.</p>
v1_compatibility	False	No	<p>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.</p>

Table B-1 (Cont.) Driver Properties

Property	Default Value	Mandatory	Description
compat_mode	v3	No	<p>Indicates the compatibility with mapping modes. This property can take the following values:</p> <ul style="list-style-type: none"> v1 is equivalent to <code>v1_compatibility=true</code> which is the 1.x compatibility mode v2 indicates the 10g/11g compatibility mode where the custom written XSD parser is used <p>Please note that when you use a DTD or only a XML file, you must specify <code>compat_mode=v2</code> in the JDBC URL. For example:</p> <pre>jdbc:snps:xml?file=/tmp/myfile.xml&compat_mode=v2</pre> <pre>jdbc:snps:xml?f=/tmp/myfile.xml&compat_mode=v2</pre> <ul style="list-style-type: none"> v3 indicates the compability with the XDK XSD parser. <p>Please note that <code>compat_mode=v3</code> is not supported when you use a DTD or only a XML file. For example, the following syntaxes are not supported:</p> <pre>jdbc:snps:xml?file=/tmp/myfile.xml&compat_mode=v3</pre> <pre>jdbc:snps:xml?f=/tmp/myfile.xml&compat_mode=v3</pre> <p>If <code>compat_mode=v3</code>, the <code>v1_compatibility</code> property will be ignored.</p>
numeric_ids	True	No	If set to true, all internal Primary and Foreign Keys are of NUMERIC type. Otherwise, they are of the VARCHAR type.
id_length	-	No	The length of the internal Primary and Foreign Key columns. The default is 10 for NUMERIC column types and 30 for VARCHAR column.
no_batch_update	False	No	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.
add_schema_bundle	-	No	<p>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 <code>add_schema_bundle</code> 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.</p> <p>All these additional schemas are created with the <code>drop_on_disconnect</code> option set to true by default.</p> <p>Example of additional schemas property files contents:</p> <pre>addschema_1=jdbc:snps:xml?f=c:/myfile.xml&ro=true&s=myschema1</pre> <pre>addschema_2=jdbc:snps:xml?file=c:/myfile2.xml&s=myschema2</pre> <pre>addschema_3=jdbc:snps:xml?d=c:/myfile3.dtd&s=myschema3</pre>

Table B-1 (Cont.) Driver Properties

Property	Default Value	Mandatory	Description
add_schema_path	-	No	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). Note: This option is not supported in v3 mode.
transform_nonascii	True	No	Transform Non Ascii. Set to false to keep non-ascii characters.
max_table_name_length	-	No	Maximum length of table names irrespective of the value as supported by internal/external DB.
max_column_name_length	-	No	Maximum length of column names irrespective of the value as supported by internal/external DB.
case_sens	True	No	Indicates whether the table and column names are case sensitive or not. Name comparisons are carried out accordingly.
default_length_varchar	255	No	Indicates the default length of the VARCHAR column used for storing XML annotation and documentation elements.
default_type_varchar	False	No	If this property is set to true, the default datatype will be VARCHAR of size 255 else on false, the LONG datatype is used.
pipeline_config_file	-	-	Pre/post processing configuration file.

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.

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 store the relational schema in an *external relational database*.

 **Note:**

The use of the In-Memory Engine is not a best practice, and may cause issues, when using the Complex File Technology.

Cause of the error: In-memory HSQL is not recommended for production use. It has memory leaks, beyond the scope of ODI that will eventually bring the JVM down. It is only meant for development use. It is recommended to switch to external DB such as Oracle, MySQL, or MS SQLServer.

Use external storage:

- When loading very large XML files with the XML driver into the relational schema derived by the XML driver
- To reduce the overall time taken to process the files with the built-in engine of the XML driver
- To avoid timeouts to the ODI repositories. Please note that the time taken to process an XML file is dependent on:
 - The complexity of the XML file structure
 - The length of XML file content
 - The host server RAM resources
 - The host server CPU resources

Before using external storage, ensure that you have understood the impacts of its usage and that you have increased the ODI timeout to values which conform to your performance requirements.

 **Note:**

Supported RDBMS for external storage include Oracle, Microsoft SQL Server, MySQL, and Hypersonic SQL 2.0. The complete list of technologies that support external storage is available on Oracle Technical Network (OTN) :

<http://www.oracle.com/technetwork/middleware/data-integrator/documentation/index.html>.

These schemas are created in addition to the one that may be created with the properties specified in the JDBC driver URL.

The external storage is configured with a set of properties described in [Table B-2](#). These properties can be passed in several ways:

- [Passing the Properties in the Driver URL](#)
- [Setting the Properties in ODI Studio](#)
- [Setting the Properties in a Properties File](#)

Passing the Properties in the Driver URL

The properties can be directly set in the Properties table as detailed in [Table B-2](#) table.

Setting the Properties in ODI Studio

The properties can be specified on the Properties table below the JDBC URL of the JDBC tab in Topology Navigator. The properties can be directly set in the Properties table as detailed in [Table B-2](#) table.

Setting the Properties in a Properties File

The properties can be set in an *external database properties file*. This properties file, also called *property bundle*, is a text file with the `.properties` extension containing a set of lines with on each line a `<property>=<value>` pair. This external database properties file contains the properties of a JDBC connection to the relational database schema. The properties file is referenced using the `db_props` property in the JDBC URL. When using this method, note the following:

- The properties in the properties file are not prefixed and used as described in [Table B-2](#).
- The `db_props` property is set to the name of the properties file including the `.properties` extension. The `db_props` property indicates that the schema must be loaded in a database schema whose connection information is stored in a *external database properties file*.
- The properties files has to be deployed by the agent using the XML connection. The location of the properties file depends on the agent you are using:
 - *Local agent (Studio)*: Place the external DB properties file in the `<user.dir>/odi/oracledi/userlib` folder
 - *Standalone Agent*: Place the external DB properties file in `domain_home/lib` folder
 - *JavaEE Agent*: The external DB properties file should be packed into a JAR or ZIP file and added to the template generated by the Java EE agent. See *Deploying an Agent in a Java EE Application Server (Oracle WebLogic Server)* in *Administering Oracle Data Integrator* for more information.
- The properties file must be set in the classpath of Oracle Data Integrator that uses the XML driver. Typically, you can install it with your custom drivers.

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. You can use the Properties table below the URL to override the file. For example:

```
jdbc:snps:xml?file=/temp/  
payload.xml&dp_driver=<external_db_driver>&dp_url=<external_db_url>
```

The properties for configuring external storage are described in [Table B-2](#).

Table B-2 Properties of the External Database Properties File

Property	Default Value	Mandatory	Description
dp_driver	-	Yes	JDBC driver name. Important: The driver class file must be in the classpath of the java application.
dp_url	-	Yes	JDBC URL
dp_user	-	Yes	Login used to connect the database
dp_pass word	-	Yes	Encrypted password of the user. Note: To encrypt the password, use the <code>encode.bat (cmd sh)</code> command. See <i>Encoding a Password in Administering Oracle Data Integrator</i> for more information.
dp_sche ma	-	Yes	Database schema storing the relational schema and the XML data. Note for MS SQLServer that: <ul style="list-style-type: none"> If schema is not specified, tables will be created under the default schema of the user If schema is specified, tables will be created under this schema Limitation when using v3 mode: When using an external database, make sure that the provided or calculated schema name exists. The <code>schema</code> driver property value must match the <code>schema</code> property value of the external database. Otherwise an error is raised.
dp_catal og	-	Yes	For Microsoft SQL Server only. Database catalog storing the XML data & information.
dp_drop_ on_ conn ect	False	No	Drop the tables from the database schema if they already exist. If set to N the existing tables are preserved.
dp_creat e_ tables	auto	No	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.
dp_creat e_ indexe s	True	No	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.
dp_nume ric_ scale	-	No	Scale of the numeric columns generated during the XML to SQL mapping.
dp_trunc ate_ befor e_ load	True	No	True: truncate all data when connecting False: preserve existing data
dp_ids_in _db	True	No	True: preserve identifiers (counters) in the database for a future append connection False: do not preserve identifiers. Future append is not possible.
dp_drop_ tables_ on_ drop_ s chema	True	No	True: 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. False: DROP SCHEMA erases the reference to the database schema from the driver, but the tables are kept in the database schema.
dp_use_ prepared _stateme nts	True	No	True: use the prepared statements with the database connection to perform driver operation (load/unload files). False: 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.

Table B-2 (Cont.) Properties of the External Database Properties File

Property	Default Value	Mandatory	Description
dp_use_batch_update	True	No	<p>True: use batch update with the database connection. False: do not use batch update.</p> <p>Inserting data is usually faster with batch update. Should be set to true only if the following conditions are met:</p> <ul style="list-style-type: none"> The database and driver support batch update The database supports prepared statements The use_prepared_statements parameter is set to Yes <p>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.</p>
dp_batch_update_size	30	No	<p>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 true.</p>
dp_commit_periodically	True	No	<p>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.</p> <p>Should be set to true only if the following conditions are met:</p> <ul style="list-style-type: none"> The database supports batch update The database supports prepared statements The use_prepared_statements parameter is set to Yes The use_batch_updates parameters is set to Yes <p>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.</p>
dp_num_inserts_before_commit	1000	No	<p>Interval in records between each COMMIT, if the commit_periodically property is set to true.</p>
dp_reserve_chars_for_column	3	No	<p>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.</p> <p>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.</p>
dp_reserve_chars_for_table	3	No	<p>Same as reserve_chars_for_column (rffc) property but applies to names of the table created in the RDBMS schema.</p>
dp_varchar_length	255	No	<p>Size of all the columns of the relational structure that will be used to contain string data. This property does not apply to Annotation or Documentation elements. For those elements dp_default_length_varchar property should be used instead.</p>
dp_default_type_varchar	N	No	<p>If set to Yes, the default datatype used in the relational schema for columns storing XML annotation and documentation elements is VARCHAR of size 255. The length of this column is specified using the dp_default_length_varchar property. If set to false, the LONG datatype if used. This property should be set to yes for technologies that do not support multiple LONG columns within the same table, such as Oracle.</p>

Table B-2 (Cont.) Properties of the External Database Properties File

Property	Default Value	Mandatory	Description
dp_default_length_varchar	255	No	Default length of the VARCHAR column used for storing XML annotation and documentation elements. This property is valid only if dp_default_type_varchar property is set to yes. For example: default_length_varchar=2000 where 2000 is the new desired default column size.
dp_numeric_length	10	No	Size of all the columns of the relational structure that will be used to contain numeric data.
dp_unicode	False	No	For MS SQL Server: If unicode = true, nvarchar is used. If unicode = false or not set, varchar is used.
dp_multi_user_safe	False	No	Its usage controls the way row ids are generated. If multi_user_safe is set to true, then each ID generation is tasked to the DB. If set to false at the very beginning of the data load, retrieve the IDs which are stored in the ID table and then work off that stored data in-memory. At the end of the data load this is then pushed to the DB.

The following sample is an example of a property file for using an Oracle Database as the external storage:

```
driver=oracle.jdbc.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=OWNER_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
```

Detailed Driver Commands

Note:

The notion of SCHEMA referred to in these commands refers to the string value set with the `s=...` parameter in the XML Driver Data Server URL present in the physical architecture.

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 FOREIGNKEYS](#)
- [CREATE XMLFILE](#)
- [CREATE SCHEMA](#)
- [DROP FOREIGNKEYS](#)
- [DROP SCHEMA](#)
- [LOAD FILE](#)
- [SET SCHEMA](#)
- [SYNCHRONIZE](#)

- UNLOCK FILE
- TRUNCATE SCHEMA
- VALIDATE
- WRITE MAPPING FILE
- COMMIT
- CREATE TABLE
- DELETE
- DISCONNECT
- DROP TABLE
- INSERT INTO
- ROLLBACK
- SELECT
- SET AUTOCOMMIT
- UPDATE

CREATE FILE

If the EMPTY option is specified, create an empty XML instance file containing all ELEMENTS (including optional ELEMENTS) present in the related XSD or DTD file. However, no XML ATTRIBUTES declared in these files will be referenced in the created XML instance file.

The attributes are handled differently between `compat_mode v1/v2` and `v3`. In `v1/v2` mode attributes are not written, while in `v3` mode attributes are also written out.

```
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_TAGS

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.

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](#).

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_TAGS

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.

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>`.

**Note:**

This command cannot be used on an external database.

```
CREATE SCHEMA <schema_name> [WITH DTD <dtd/xsd_name>] [REPLACE]
  [ROOTELT <root element>] [READONLY] [COMPAT_MODE <compatibility mode>]
  [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.

COMPAT_MODE

Indicates the compatibility with mapping modes. This property can take the following values:

- `v1` is equivalent to `v1_compatibility=true` which is the 1.x compatibility mode
- `v2` is the 10g/11g mode. This is the default mode.

Please note that when you use a DTD or only a XML file, you must specify `compat_mode=v2` in the JDBC URL. For example:

```
jdbc:snps:xml?d=/tmp/myDTD.dtd&compat_mode=v2
```

```
jdbc:snps:xml?f=/tmp/myfile.xml&compat_mode=v2
```

- `v3` indicates the compatibility with the XDK XSD parser. Please note that `compat_mode=v3` is not supported when you use a DTD or only a XML file. For example, the following syntaxes are not supported:

```
- jdbc:snps:xml?d=/tmp/myDTD.dtd&compat_mode=v3
```

```
- jdbc:snps:xml?f=/tmp/myfile.xml&compat_mode=v3
```

If `compat_mode=v3`, the `v1_compatibility` property will be ignored.

Note:

When using the SYNCHRONIZE command, only those DB schemas that have been created with 'v3' option will parse the DTD/XSD in the 'v3' mode. In 'v3' mode all the restrictions on schema name value corresponding with DB property for schema name etc. will apply.

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 in double quotes.
- The encoding values should be enclosed in double quotes as they may contain special characters.

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
```

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>]
```

LOAD FILE

Load the `<file_name>` XML file into the specified `<schema_name>` XML schema. If a schema name is not specified with the **ON SCHEMA** parameter, one is generated with the XML file name. If a schema with the specified or generated name is found, then the properties of that schema are inherited. If a schema with the specified or generated name does not exist at runtime, a new XML JDBC URL with only the properties specified in the **LOAD FILE** command is created. This schema does not inherit any of the properties of the current 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>]
```

If **INSERT_ONLY** is absent,

- i) Check to see if schema identifier is provided in the command via **ON SCHEMA**
- ii) If not, use XML file name to generate a schema identifier (first 5 characters)

If this schema is already present and **REPLACE** option is not present in the command, driver raises error. If the schema is already present, and **REPLACE** option is present, it drops the existing schema and recreates it, loading the data from the FILE into it.

If schema is not present, create new schema and load FILE.

If **INSERT_ONLY** is present,

- i) Use current schema
- ii) If **ON SCHEMA** is provided, check if schema of that name exists. If it does, use it, otherwise continue with the current schema. Read FILE into this schema.

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.

The **INSERT_ONLY** option just inserts data from the XML file. Existing data is not touched.

If existing data is same as the data being inserted, it may raise an error.

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.

REPLACE option causes 'drop' of the entire schema. When schema is dropped, all connections are also closed.

For 'standalone' dataserver, when all connections are closed, in-memory HSQL DB itself is destroyed.

Then an entirely new database instance is created for 'standalone' dataserver.

In all cases, an entirely new schema and tables are created and data from the XML file is inserted.

This option is to be used, only when same schema name is to be used for an entirely different XML structure.

Using this option is very non-performant as may be deduced.

It will also cause problems, if the dataserver is in 'standalone' mode and the same dataserver is being used again, after LOAD FILE command.

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

- Enclose the file 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.

SET SCHEMA

Set the current schema to <schema_name>.

```
SET SCHEMA <schema_name>
```

Remarks

It is necessary to specify a name for the schema.

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.

UNLOCK FILE

Unlocks <file_name> if it is locked by another instance of the driver.

```
UNLOCK FILE <file_name>
```

TRUNCATE SCHEMA

Clears all data from the current schema, or from <schema_name>.

```
TRUNCATE SCHEMA [<schema_name>]
```

The TRUNCATE command merely deletes all data from all tables of the schema. Nothing is dropped. Connections are not closed.

VALIDATE

Verifies that the XML file <file_name> is well-formed and validates the content of the XML file <file_name> against the XML Schema (XSD) if the schema is referenced in the XML file. This command returns an exception if the file is not valid. For a full description of the validation performed, see:

<http://xerces.apache.org/xerces2-j/features.html#validation.schema>

```
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.

WRITE MAPPING FILE

Writes out the element/attribute name to table/table.column name mapping for each element/attribute to the specified file. The mapping file helps to understand the relational structure that has been created for the XSD/DTD file. This command can be used only when the schema was created in v3 mode. Otherwise exception is thrown.

```
WRITEMAPPINGFILE FILE <file-path> [FROM SCHEMA <schema-name>]
        [JAVA_ENCODING <java_encoding> XML_ENCODING <xml-encoding>]
```

Parameters

file_path

Name of the generated mapping file

FROM_SCHEMA

If the optional FROM_SCHEMA parameter is not provided, the current schema will be used.

JAVA_ENCODING

Encoding of the generated file, for example: ISO8859_1. You will find a list of supported encoding at the following URL: <http://download.oracle.com/javase/6/docs/technotes/guides/intl/encoding.doc.html>.

Note that if the Java encoding is specified, the XML encoding should also be specified.

XML_ENCODING

Encoding in the xml tag of the generated file.

Example of generated tag: `<?xml version="1.0" encoding="ISO-8859-1"?>`

You will find a list of supported encoding at the following URL: <http://download.oracle.com/javase/6/docs/technotes/guides/intl/encoding.doc.html>.

Note that if the XML encoding is specified, the Java encoding should also be specified.

Example B-1 Mapping File

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<personnel xmlns:x2r="http://www.oracle.com/odi/xml-mapping"
x2r:tableName="PERSONNEL">
  <person x2r:tableName="PERSON" id="ID" select="SELECT_">
    <email x2r:tableName="EMAIL"></email>
    <link x2r:tableName="LINK" manager="MANAGER" subordinates="SUBORDINATES"></
link>
    <name x2r:tableName="NAME">
      <given x2r:columnName="GIVEN"></given>
      <family x2r:columnName="FAMILY"></family>
    </name>
    <url x2r:tableName="URL" href="HREF"></url>
  </person>
</personnel>
```

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.

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

COMMIT

Ends a transaction on the schema and makes the changes permanent.

```
COMMIT [WORK]
```

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

DELETE

Remove rows in a table in the relational schema. This function uses a standard SQL Syntax.

```
DELETE FROM <table_name> [ WHERE <expression> ]
```


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.

DROP TABLE

Remove a table, the data and indexes from the relational schema.

```
DROP TABLE <table_name>
```

INSERT INTO

Insert one or more new rows of data into a table.

```
INSERT INTO <table_name> [ ( <column_name> [,...] ) ]
    { VALUES (<expression> [,...]) | <SELECT Statement> }
```

ROLLBACK

Undo the changes made since the last COMMIT or ROLLBACK.

```
ROLLBACK
```

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 ]
```

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 }
```

UPDATE

Modifies data of a table in the database.

```
UPDATE table SET column = <expression> [, ...] [WHERE <expression>]
```

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).

SQL FUNCTIONS

[Table B-3](#) lists the numerical functions.

Table B-3 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-4](#) lists the string functions.

Table B-4 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
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-5](#) 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-5 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)

Table B-5 (Cont.) Date/Time Functions

Function	Description
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-6 lists the system functions.

Table B-6 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

JDBC API Implemented Features

Table B-7 lists the JDBC API features that are implemented in the Oracle Data Integrator Driver for XML:

Table B-7 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-8 lists JDBC Java classes.

Table B-8 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
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

Rich Metadata

When creating RDB structures based on XML schema, there must be flexibility to supply the driver with metadata. For example, in situations where RDB table/column names can conflict if element/attributes have same local names.

The ODI XML driver attaches an attribute in the x2r namespace (<http://www.oracle.com/odi/xml-mapping>) to the elements/attribute namely: x2r:tableName/x2r:columnName. If conflicting names do not have the metadata attribute, then they are appended with an incrementing number until a non-conflicting table/column name is obtained.

The new object model maintains a map between xpath and table/table.column names for each element/attribute.

If two elements with same name and same type exist in two different locations, same table is used for storing the data but FK reference to parent element is used to differentiate the data. The new implementation creates new tables. [Table B-9](#) lists the table attributes.

Table B-9 Table Attributes

Attribute	Type	Description
x2r:tableName	String	To be attached to elements that resolve to RDB tables/ attributes that are lists or enumerations whose local names match.
x2r:columnName	String	To be attached to attributes whose local names match or for elements that map to columns, but whose local names match with each other or with an attribute of the containing type.
x2r:columnDataType	String	Lets you provide the datatype information as a string from a mapping table that we will provide. May only be attached to elements that the driver will map to columns or to attributes. If this parameter is provided user must also supply x2r:columnLength and/or x2r:columnPrecision as required for the datatype.
x2r:columnLength	integer	Length of the column. By default the values hard-coded in the driver are used. VARCHAR and NUMERIC have global override option in JDBC URL. This attribute, if provided, overrides both the default value and the global override. May only be attached to elements that the driver will map to columns or to attributes.
x2r:columnPrecision	integer	Precision of the column. Used by driver only for those datatypes that allow it. Same logic as for columnLength is used when determining the value to be applied. May only be attached to elements that the driver will map to columns or to attributes.

The following sample is an example of an XSD enriched with metadata.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:x2r="http://
www.oracle.com/odi/xml-mapping">
  <xs:element name="root">
    <xs:complexType>
      <xs:sequence>
        <!-- Example for redefining table name -->
        <xs:element name="person" maxOccurs="unbounded" x2r:tableName="CUSTOMER">
          <xs:complexType>
            <xs:sequence>
              <!-- Example for redefining column name -->
              <xs:element name="given" type="xs:string" x2r:columnName="FIRST"/>
              <xs:element name="last" type="xs:string"/>
              <!-- Example for redefining column length -->
              <xs:element name="address" type="xs:string" x2r:columnLength="400"/>
              <!-- Example for redefining column type -->
              <xs:element name="notes" type="xs:string" x2r:columnDataType="CLOB"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Supported user-specified types for different databases

Table B-10 provides the details of the supported user-specified types for different databases. Using any other type name will raise exception.

Table B-10 Supported user-specified types for databases

Type	HSQL	Oracle	MySQL	MS SQL Server
SMALLINT	X		X	X
INTEGER	X		X	
REAL	X			X
NUMERIC	X		X	
NUMBER		X		
FLOAT	X		X	X
DOUBLE	X		X	
DECIMAL	X		X	
CHAR	X	X	X	X
NCHAR		X	X	X
VARCHAR	X	X	X	X
VARCHAR2		X		
NVARCHAR2		X		
BLOB	X	X	X	
CLOB	X	X		
NCLOB		X		
TEXT			X	X
DATE	X	X	X	
TIME	X	X	X	
TIMESTAMP	X	X	X	X

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](#)

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

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
- Any
- AnyAttribute
- AnyType
- 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.

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>
```

Any

This element enables you to extend the XML document with elements not specified by the schema.

```
<any
  id=ID
  maxOccurs=(nonNegativeInteger|unbounded):1
  minOccurs=nonNegativeInteger:1
  namespace=((##any|##other)|List of (anyURI(##targetNamespace|##local))):##any
  processContents=(lax|skip|strict):strict
  any attributes
>
(annotation?)
</any>
```

AnyAttribute

This element enables you to extend the XML document with attributes not specified by the schema.

```
<anyAttribute
  id=ID
  namespace=((##any|##other)|List of (anyURI(##targetNamespace|##local))):##any
  processContents=(lax|skip|strict):strict
  any attributes
>
(annotation?)
</anyAttribute>
```

AnyType

This XML Schema type is the root type for all XML Schema types.

```
<xsd:element name="something" type="xsd:anyType"/>
```

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.

AttributeGroup

This element defines a set of attributes.

```
<attributeGroup
  id=ID
  name=NCName
  ref=QName
  any attributes
>
(annotation?),((attribute|attributeGroup)*,anyAttribute?)
</attributeGroup>
```

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.

ComplexContent

This element defines extensions or restrictions on a complex type.

```

<complexContent
  id=ID
  mixed=true|false
  any attributes
>
(annotation?, (restriction|extension))
</complexContent>

```

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)?, ((attribute|
attributeGroup)*, anyAttribute?))))
</complexType>

```

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 considered 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 element 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.

Extension

This element extends an existing `simpleType` or `complexType` element

```
<extension
  id=ID
  base=QName
  any attributes
>
(annotation?, ((group|all|choice|sequence)?, ((attribute|
attributeGroup)*, anyAttribute?)))
</extension>
```

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>
```

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>

```

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>

```

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>

```

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>
```

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>
```

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.

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>
```

SimpleType

This element defines a simple type element.

```
<simpleType
  name=NCName
  id=ID
  any attributes
>
(annotation?,(restriction|list|union))
</simpleType>
```

Unsupported Features

The following elements and features are not supported or implemented by the XML driver.

Unsupported Elements

The following schema elements are not supported by the XML driver.

- **Key/keyRef/Unique:** These elements allow the definition of constraints in the schema. These elements and their child elements (selector, field) are ignored.
- **Redefine:** The redefine element redefines simple and complex types, groups, and attribute groups from an external schema. This element is not supported.

In v3 mode an error is raised, if any unsupported XSD element is encountered.

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.

Unsupported Features

Multipass parsing is supported in v3 mode. The other modes do not support multipass parsing.

Unsupported Datatypes

The following datatypes are not supported:

- gYear
- gYearMonth
- gMonth
- gMonthDay
- gDay
- language
- ENTITY

- ENTITIES
- NOTATION
- IDREFS

C

Oracle Data Integrator Driver for Complex Files Reference

The Oracle Data Integrator Driver for Complex Files (Complex File driver) allows Oracle Data Integrator to use complex files as data servers.

Note:

The use of the In-Memory Engine is not a best practice, and may cause issues, when using the Complex File Technology.

Cause of the error: In-memory HSQL is not recommended for production use. It has memory leaks, beyond the scope of ODI that will eventually bring the JVM down. It is only meant for development use. It is recommended to switch to external DB such as Oracle, MySQL, or MS SQLServer.

This appendix includes the following sections:

- [Introduction to Oracle Data Integrator Driver for Complex Files](#)
- [Complex Files Processing Overview](#)
- [Driver Configuration](#)
- [Detailed Driver Commands](#)
- [JDBC API and XML Schema Supported Features](#)

Introduction to Oracle Data Integrator Driver for Complex Files

The *Oracle Data Integrator Driver for Complex Files (Complex File driver)* handles files in a Complex (or Native) Format as a JDBC data source. This allows Oracle Data Integrator to use complex files as data servers.

With the Complex File driver, Oracle Data Integrator can query complex files using standard SQL syntax and perform changes in the complex files. These operations occur within transactions and can be committed or rolled back.

The Oracle Data Integrator driver for Complex Files 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 into the complex files

Complex Files Processing Overview

The Complex File driver uses a *Native Schema* file. This file, written in the nXSD format describes the structure of the Native File and how to translate it to an XML file.

The Complex File driver translates internally the native file into an XML structure, as defined in the Native Schema (nXSD) description and from this XML file it generates a relational schema that is consumed by Oracle Data Integrator. The overall mechanism is shown in [Figure C-1](#).

Figure C-1 Complex File Driver Process



The second part of the process, starting from the XML structure, corresponds precisely to the capabilities of the *Oracle Data Integrator Driver for XML*.

The Complex Files driver works in the following way:

1. The complex file is translated to an intermediate XML file using the Native Schema (nXSD) file. Note that no physical file is created for the intermediate XML file but a streaming XML structure.
2. The driver *loads* the XML structure and data into a relational schema, using a [XML to SQL Mapping](#).
3. The user works on the relational schema, manipulating data through regular SQL statements or specific driver commands for driver operations.
4. Upon disconnection or user request, the Complex Files driver *synchronizes* the data and structure stored in the schema back to the complex file.

Generating the Native Schema

The Native Schema can be created manually, or generated using the Native Format Builder Wizard available as part of Fusion Middleware Technology Adapters. See Native Format Builder Wizard in the *User's Guide for Technology Adapters* for more information on the Native Schema format and the Native Format Builder Wizard.

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. See [XML to SQL Mapping](#) for more information.

JSON Support

Flat files in JSON format are supported through the nXSD format. The nXSD file can be created manually or through the Native Format Builder Wizard (See [Generating the Native Schema](#) for details). If an XSD file with no nXSD annotation is used, you need to provide additional JDBC property: `tt=json` or `translator_type=json`, which will enable the driver to use the JSON translator for parsing the input file.

Supported Features

The Complex File driver supports the same features as the XML driver:

- Schema Storage in a *built-in engine* or *external database* is supported in the same way as the XML Driver. See [Schema Storage](#) and [Using an External Database to Store the Data](#) for more information.
- Multiple Schemas are supported, with the following differences:
 - Only a single schema can be created at connection time, based on the Native Schema file.
 - Parameters allowing creating multiple schemas at connection time as indicated in [Automatically Create Multiple Schemas](#) are not supported. This includes `add_schema_bundle`, `add_schema_path`, and `addschema_X`.
 - Additional schemas can be created after the connection using the CREATE SCHEMA and LOAD FILE commands.
- Case-sensitivity is managed similarly to the XML driver. See [Case Sensitivity](#) for more information.
- Loading/Synchronizing with the Complex File driver works the same way as the XML Driver. Loading/Synchronizing operations automatically propagate to the Native file. See [Loading/Synchronizing](#) for more information.
- Locking is supported. When connected, the complex file is locked and when disconnected, it is unlocked. The UNLOCK FILE command is supported.

Driver Configuration

The Oracle Data Integrator driver for Complex Files is automatically installed with Oracle Data Integrator. The following topics cover advanced configuration topics and reference information.

This section details the driver configuration.

- The driver name is: `oracle.odi.jdbc.driver.file.complex.ComplexFileDriver`
- The URL Syntax is: `jdbc:snps:complexfile?f=<native file location>&d=<native schema>&re=<root element name>[&s=<schema name>&<property>=<value>...]`

The properties for the URL are detailed in [Oracle Data Integrator Driver for Complex Files Reference](#).

Table C-1 Driver Properties

Property	Mandatory	Type	Default	Description
file or f	Yes	string (file location)	-	Native file location. 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. This parameter is mandatory.
dtd or d	Yes	string (file location)	-	Native Schema (nXSD) file location. This parameter is mandatory.
root_elt or re	Yes	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 section of the Native Schema. This parameter is mandatory.
read_only or ro	No	boolean (true false)	false	Open the native file in read only mode.
schema or s	No	string	-	Name of the relational schema where the complex file will be loaded. This parameter is mandatory. This schema will be selected when creating the physical schema under the Complex File data server. 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, and cannot be named like an existing XML element.
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. Note: This option is not applicable when an external database is used.
translator_type or tt	No	string (json)	-	If this option is set to json, the xsd does not require nXSD annotations and will automatically use the JSON translator for parsing the input file.
db_props or dp	No	string	-	This property is used to use an external database instead of the memory engine to store the schema. See Using an External Database to Store the Data for more information.
load_data_on_connect or ldoc	No	boolean (true false)	true	Automatically load 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.

Table C-1 (Cont.) Driver Properties

Property	Mandatory	Type	Default	Description
drop_on_disconnect or dod	No	boolean (true false)	false	Automatically drop 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. See Using an External Database to Store the Data for more information.
useimplicitmaxvalue	No	boolean (true false)	false	When this property is set to true, elements for which maxOccurs is not specified in the schema are considered as maxOccurs="unbounded". Otherwise, the driver assumes that maxOccurs=1 when maxOccurs is not specified.
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: https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html .
numeric_ids 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 in the relational schema.
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.
transform_nonascii or tna	No	boolean (true false)	true	Transform Non Ascii. Set to false to keep non-ascii characters.

The following example illustrates these properties:

Connects to the PROD20100125_001.csv file described by products.nxsd and expose this file as a relational structure in the PRODUCT Schema.

```
jdbc:snps:complexfile?f=/infiles/PROD20100125_001.csv&d=/infiles/products.nxsd&re=root&s=PRODUCTS
```

Detailed Driver Commands

The Complex File driver supports the same driver commands as the XML driver. See [Detailed Driver Commands](#) for the driver commands supported by the XML Driver.

The exceptions to this rule are the following:

- In the Complex File driver syntax, the commands that are related to the XML file such as CREATE FILE or LOAD FILE, are applied to the Native File. For example, the command CREATE FILE creates a native format file from the schema content.
- VALIDATE is not supported.
- CREATE FILE is supported but the NO_CLOSING_TAGS and NO_DEFAULT_NS parameters are ignored.
- CREATE SCHEMA requires the WITH DTD parameter.
- LOAD FILE requires the WITH DTD parameter.

JDBC API and XML Schema Supported Features

The Complex File driver supports the same JDBC features as the XML driver. See [SQL Syntax](#) for more information.

D

Pre/Post Processing Support for XML and Complex File Drivers

It is possible to customize the way in which data is fed to the XML and Complex File drivers. You can set up intermediate processing stages to process the data that is retrieved from an external endpoint using Oracle Data Integrator, or to write the data out to an external endpoint.

This appendix includes the following sections:

- [Overview](#)
- [Configuring the processing stages](#)
- [Implementing the processing stages](#)
- [Example: Groovy Script for Reading XML Data From Within a ZIP File](#)
- [Example: Groovy Script for Transforming XML Data and Writing to a Different Format](#)
- [Example: Java Class for Reading Data From HTTP Source Requiring Authentication](#)
- [Example: Groovy Code Embedded in Configuration XML File](#)

Overview

You can now customize the way data is fed to the XML and Complex File drivers. You can set up intermediate processing stages to process the data that is retrieved from an external endpoint using Oracle Data Integrator, or to write the data out to an external endpoint.

You can configure one Terminal stage and zero or multiple Junction stages. The terminal stage can read data from external endpoints and write data to external endpoints. The terminal stage reads the source data from an external endpoint and passes it to the junction stage for processing. The junction stages can be configured to process the data passed by the terminal stage.

The source data can be in any format, not necessarily XML or Complex File, until it reaches the XML driver or the Complex File driver. However, when the data is finally handed off to the XML driver or the Complex File driver, the data must be in the required format. That is, when the data is handed off to the XML driver, it must be a valid XML that adheres to the XSD that has been configured for the data server. Similarly, when the data is handed off to the Complex File driver, the data must exactly match the pattern as defined by the nXSD file.

Configuring the processing stages

The complete configuration of the intermediate processing stages to the ODI JDBC driver in the form an XML file. The XSD for the configuration XML file must also be included.

For an input pipeline configuration, the first stage would be the one that first processes the input. The last stage would be the one that feeds data to the driver. This last stage must provide an output that adheres to the format expected by the XML or the Complex File driver.

For an output pipeline configuration, the last stage would be the one that writes out the output. The first stage would be the one that accepts the data from the driver. This data would have the same shape as the XSD of the dataserver.

After you create the XML file that contains the configuration, ensure that the `pipeline_config_file` or `pcf` property of the XML driver or the Complex File driver points to the absolute file location of the XML file.

[Example D-1](#) shows a sample configuration XML file.

Example D-1 Sample Configuration XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<pipeline xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="pre-post.xsd">

  <input-stages>
    <io-stage name="restInput">
      <codeDefinition>
        <javaClass>com.company.org.InputProcessor</javaClass>
      </codeDefinition>
      <debugOutput>http://tempuri.org</debugOutput>
    </io-stage>
    <stage name="BufferInputStage">
      <codeDefinition>
        <javaClass>com.company.org.BufferingClass</javaClass>
      </codeDefinition>
      <props>
        <property name="bufferSizeBytes">2340</property>
      </props>
    </stage>
    <stage name="UnzipStage">
      <codeDefinition>
        <code>[Groovy text in Base64 encoded form]</code>
      </codeDefinition>
    </stage>
  </input-stages>
  <output-stages>
    <io-stage name="restOut">
      <codeDefinition>
        <javaClass>com.company.org.OutputProcessor</javaClass>
      </codeDefinition>
      <debugOutput>http://tempuri.org</debugOutput>
    </io-stage>
    <stage name="SevenZipOutputStage">
      <codeDefinition>
        <code>[Groovy text in Base64 encoded form]</code>
      </codeDefinition>
    </stage>
    <stage name="BufferOutputStage">
      <codeDefinition>
        <javaClass>com.company.org.PushOutput</javaClass>
      </codeDefinition>
      <debugOutput>/scratch/jsmith/view_storage/tmp/bufferout.txt</debugOutput>
    </stage>
  </output-stages>
</pipeline>
```

```
</output-stages>  
</pipeline>
```

Implementing the processing stages

Pre or post data processing support for XML driver and Complex File driver may be implemented in three different ways.

- **Groovy Code**

By supplying the Groovy code directly into the configuration XML file. This Groovy code is a part of the dataserver configuration and cannot be re-used. You can supply the Groovy code as a Base64 encoded string or as a plain text string within a CDATA section.

For an example, see [Example: Groovy Code Embedded in Configuration XML File](#).

- **Java Class**

By providing the fully qualified name of a Java class. This Java class must be available on the ODI Agent classpath at runtime.

For ODI Studio it might be made into a JAR and placed in `USER_HOME/odi/oracledi/userlib` directory.

For Standalone or Collocated agents this JAR must either be placed in `DOMAIN_HOME/lib` directory or should be coded into the classpath using one of the scripts.

For JEE Agents it must be deployed as a shared library and ODI Agent application must depend on this shared library.

For an example, see [Example: Java Class for Reading Data From HTTP Source Requiring Authentication](#).

- **Groovy Script**

By providing the name of a Groovy script. All the requirements of Java class apply to this Groovy script as well. As an exception you may provide either the name of the script, for example, `MyGroovySource.groovy` or an absolute path to the script, for example, `/home/groupuser/name/MyCustomGroovy.groovy`.

In the former case, the script is looked up as a Java Class resource using the `ClassLoader`. The usual locator pattern for class resources applies for this. For example, if the file is not in a JAR, the file name must be provided as `/MyGroovySource.groovy`. If it is in a subdirectory of a JAR, then the locator will be `/com/foo/MyGroovySource.groovy`. If using absolute path, the Groovy script is accessed as a plain Java File.

For examples, see the following sections:

- [Example: Groovy Script for Reading XML Data From Within a ZIP File](#)
- [Example: Groovy Script for Transforming XML Data and Writing to a Different Format](#)

 **Note:**

Take a note of the following:

- The changes in the embedded Groovy code or Groovy script file located via absolute path will not be picked up unless the XML driver schema is dropped. In the case of Java class or Groovy script file located via classpath, you must restart the JVM to pick up the changes.
- The inline Groovy code, Groovy script, or Java class must all conform to the Java interfaces as provided in the Public APIs. ODI driver will apply chaining to the resultant code with the ordering as set up in the configuration and the data will flow through the multiple stages as configured.

Example: Groovy Script for Reading XML Data From Within a ZIP File

Following is an example of a Groovy script to read XML data from within a ZIP file.

Example D-2 Groovy Script: Read XML Data from within a ZIP file

```
import java.io.IOException
import java.io.InputStream;
import java.util.Properties;
import java.util.logging.Logger;

import oracle.odi.jdbc.drivers.common.pipeline.api.Stage;
import oracle.odi.jdbc.drivers.common.pipeline.api.TerminalStreamInputStage;

class FileFromZip extends TerminalStreamInputStage {

    public FileFromZip(Properties pStageProperties, String pDataseverUrl,
        Properties pDataseverProperties, String pJavaEncoding,
        Logger pLogger, String pDebugLocation, String pDebugEncoding, String
pStageName) {

        super(pStageProperties, pDataseverUrl, pDataseverProperties,
            pJavaEncoding, pLogger, pDebugLocation, pDebugEncoding, pStageName);
    }

    @Override
    public InputStream readSource() throws IOException {
        def zipFile = new java.util.zip.ZipFile(new
File(getStageProperties().get("ZIP_FILE")))
        def zipEntry = zipFile.entries().find { !it.directory &&
getStageProperties().get("XML_FILE").equalsIgnoreCase(it.name)}
        return zipFile.getInputStream(zipEntry)
    }

    @Override
    public void close() throws IOException {
        // TODO Auto-generated method stub
    }
}
```

```

    }
}

```

Example: Groovy Script for Transforming XML Data and Writing to a Different Format

Following is an example of a Groovy script to transform XML data and write it out to a different format.

Example D-3 Groovy Script: Transform XML data and write it to a different format

```

package oracle.odi.jdbc.driver

import groovy.xml.MarkupBuilder;

import java.io.IOException;
import java.io.OutputStream
import java.util.Properties;
import java.util.logging.Logger;

import oracle.odi.jdbc.drivers.common.pipeline.api.JunctionStreamOutputStage;
import oracle.odi.jdbc.drivers.common.pipeline.api.Stage;

class TransformXmlOutput extends JunctionStreamOutputStage {

    private OutputStream output

    public TransformXmlOutput(Properties pStageProperties, String pDataserverUrl,
        Properties pDataserverProperties, String pJavaEncoding, Logger pLogger, String
pDebugLocation,
        String pDebugEncoding, String pStageName) {
        super(pStageProperties, pDataserverUrl, pDataserverProperties,
pJavaEncoding, pLogger,
        pDebugLocation, pDebugEncoding, pStageName);
    }

    @Override
    public OutputStream writeOutput(OutputStream out) {
        System.out.println("In TransformXmlOutput writeOutput")
        def Writer w = new BufferedWriter(new OutputStreamWriter(out))
        System.out.println("Created writer")
        output = pipeInput { input ->
            // Perform transformation
            System.out.println("Piping")
            def builder = new MarkupBuilder (w);
            def cars = new XmlSlurper().parse(input)

            System.out.println("Parsed XML")

            builder.mkp.xmlDeclaration(version: "1.0", encoding: "utf-8")

            builder.html(xmlns:"http://www.w3.org/1999/xhtml") {
                head {
                    title "Cars collection"
                }
                body {
                    h1("Cars")
                }
            }
        }
    }
}

```

```

        ul(){
            cars.car.each{car ->
                li(car.@name.toString() + "," + car.country +
                    "," + car.description + ", Age: " + (2012 - car.@year.toInteger()) + " years")
            }
        }
    }
}
w.flush()
System.out.println("Closing connectedStage")
closeConnectedStage();
}
}

@Override
public void close() throws IOException {
    System.out.println("Closing TransformXmlOutput")
    if(output!= null) {
        output.flush();
        output.close()
    }
}

public static OutputStream pipeInput(Closure read) {

    PipedInputStream input = new PipedInputStream()
    PipedOutputStream output = new PipedOutputStream(input)
    getThreadsSource.submit {
        try{
            read(input)
        } catch (Exception e) {
            System.out.println("Exception in thread")
            e.printStackTrace();
            throw e;
        } finally {
            output.flush()
        }
    }
    return output
}
}
}

```

Example: Java Class for Reading Data From HTTP Source Requiring Authentication

Following is an example of a Java class to read data from an HTTP source that requires authentication.

Example D-4 Java Class: Read Data From HTTP Source Requiring Authentication

```

/**
 *
 */
package oracle.odi.jdbc.driver.xml;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;

```

```

import java.net.URL;
import java.net.URLConnection;
import java.util.Properties;
import java.util.logging.Logger;

import oracle.odi.jdbc.drivers.common.pipeline.api.TerminalStreamInputStage;

/**
 * @author jsmith
 *
 */
public class FromHttpBasicAuthJava extends TerminalStreamInputStage {

    /**
     * @param pStageProperties
     * @param pDataseverUrl
     * @param pDataseverProperties
     * @param pJavaEncoding
     * @param pLogger
     * @param pDebugLocation
     * @param pDebugEncoding
     * @param pStageName
     */
    public FromHttpBasicAuthJava(Properties pStageProperties, String pDataseverUrl,
        Properties pDataseverProperties, String pJavaEncoding,
        Logger pLogger, String pDebugLocation, String pDebugEncoding,
        String pStageName) {
        super(pStageProperties, pDataseverUrl, pDataseverProperties,
            pJavaEncoding, pLogger, pDebugLocation, pDebugEncoding,
            pStageName);
    }

    /* (non-Javadoc)
     * @see
     oracle.odi.jdbc.drivers.common.pipeline.api.TerminalStreamInputStage#readSource()
     */
    @Override
    public InputStream readSource() throws IOException {
        String username = (String)(getStageProperties().get("username"));
        String password = (String)(getStageProperties().get("password"));
        byte[] credential = org.apache.commons.codec.binary.Base64.encodeBase64(
            (username + ":" + password).getBytes());

        //pass encoded user name and password as header
        URL url = new URL ("http://localhost:18000/get");
        URLConnection conn = url.openConnection();
        conn.setRequestProperty ("Authorization", "Basic " + new String(credential));
        InputStream urlStream = conn.getInputStream();
        StringBuilder result = new StringBuilder();
        byte[] read;
        int bytesRead;
        while(true) {
            read = new byte[1024];
            if((bytesRead = urlStream.read(read)) == -1) {
                break;
            } else
                result.append(new String(read, 0, bytesRead));
        }

        return new ByteArrayInputStream(result.toString().getBytes());
    }
}

```

