Oracle® Fusion Middleware

Administering Oracle Data Integrator

12*c* (12.2.1.4.0)
E95625-03
March 2024

Oracle Fusion Middleware Administering Oracle Data Integrator, 12*c* (12.2.1.4.0)

E95625-03

Primary Author: Oracle Corporation

Contents

3 Setting Up a Topology

4 High Availability for Oracle Data Integrator

Part II Creating and Using Data Services

5 Setting Up Data Services

6 Generating and Deploying Data Services

Part III Running and Monitoring Integration Processes

7 Running Integration Processes

8 Debugging Integration Processes

9 Monitoring Integration Processes

10 Using Oracle Data Integrator Console

Part IV Managing Security Settings

11 Managing Security in Oracle Data Integrator

Index

Preface

This guide describes how to perform configuration and user management tasks in Oracle Data Integrator (ODI). This includes configuring ODI components, performing basic administrative tasks, running and monitoring integration processes, and managing security in ODI.

This preface contains the following topics:

Audience

This document is intended for administrators who perform ongoing configuration tasks 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*
- *Developing Integration Projects with Oracle Data Integrator*
- *Installing and Configuring Oracle Data Integrator*
- *Upgrading Oracle Data Integrator*
- *Integrating Big Data with Oracle Data Integrator Guide*

- *Application Adapters Guide for Oracle Data Integrator*
- *Developing Knowledge Modules with Oracle Data Integrator*
- *Connectivity and Knowledge Modules Guide for Oracle Data Integrator Developer's Guide*
- *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.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New In Oracle Data Integrator?

This section summarizes the new features and significant product changes for Oracle Data Integrator (ODI) in the Oracle Fusion Middleware 12*c* release.

This chapter includes the following sections:

New and Changed Features for Release 12c (12.2.1.4.0)

Oracle Data Integrator 12c (12.2.1.4.0) introduces the following enhancements:

- **Oracle Sales Cloud** - Oracle Data Integrator integrates with Oracle Sales Cloud. The Oracle Sales Cloud JDBC drivers are now packaged with Oracle Data Integrator. It allows you to create an Oracle Sales Cloud Technology and a Data Server, which can be used to reverse engineer the Oracle Sales Cloud Model and create ETL mappings.

  For more information, see Oracle Sales Cloud chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* guide.

- **Oracle Service Cloud** - Oracle Data Integrator integrates with Oracle Service Cloud. The Oracle Service Cloud JDBC drivers are now packaged with Oracle Data Integrator. It allows you to create an Oracle Service Cloud Technology and a Data Server, which can be used to reverse engineer the Oracle Service Cloud Model and create ETL mappings.

  For more information, see Oracle Service Cloud chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* guide.

- **GIT Offline Support** - Oracle Data Integrator (ODI) integration with Version Control System (VCS) has been extended by Git offline mode support. Now you can perform VCS operation in local Git repository and push to the remote Git Repository on need basis or when the remote Git Repository is reachable.

  For more information, see Integrating ODI with Version Control Systems chapter in the *Developing Integration Projects with Oracle Data Integrator* guide.

- **SAP Delta Extraction** - SAP delta extraction provides advanced table join support. This can be used to join with SAP CHDR table for delta extraction.

  For more information, see Using Advanced Join Handling for Change Detection with SAP CDHDR table chapter in the *Getting Started with SAP ABAP ERP Adapter for Oracle Data Integrator* guide.

New and Changed Features for Release 12c (12.2.1.3.1)

Oracle Data Integrator 12c (12.2.1.3.1) introduces new functionality in the following areas:

- Integration with Oracle Object Storage and Oracle Object Storage Classic
- Optimized Knowledge Modules (KMs) for Oracle Autonomous Data Warehouse Cloud and Oracle Autonomous Transaction Processing
- Support for Oracle Enterprise Resource Planning (ERP) Cloud

These key investment areas ensure that Oracle Data Integrator (ODI) will continue to accompany customers throughout their technological transformation and modernization process.

Oracle Object Storage and Oracle Object Storage Classic

Oracle Object Storage and Object Storage Classic offers fast, reliable and secure cloud storage and now Oracle Data Integrator can seamlessly integrate with them on Oracle Cloud Infrastructure (OCI).

ODI comes with a set of Knowledge Modules (KMs) and ODI Tools that can used in Mappings and Packages to connect to Oracle Object Storage and Object Storage Classic for uploading, downloading and deleting files/objects onto/from local directory or the Hadoop Distributed File System (HDFS).

For more details, see Oracle Object Storage and Oracle Storage Cloud Service Documentation.

Autonomous Databases: Autonomous Data Warehouse and Autonomous Transaction Processing

Oracle Data Integrator now comes with optimized Loading and Integration Knowledge Modules (KMs) that are certified with Oracle Autonomous databases:

- Oracle Autonomous Data Warehouse Cloud (ADWC)
- Oracle Autonomous Transaction Processing (ATP)

Oracle Data Integrator seamlessly integrates with ADWC and ATP. By integrating ODI with Autonomous databases, you can get the full performance of Oracle Data Integrator and Oracle databases, in a fully-managed environment that is tuned and optimized for various workloads. The same set of Knowledge Modules are used for both ADWC and ATP and also leverage the new native integration with Oracle Object Storage and Oracle Object Storage Classic.

In addition to being able to load data directly into ADWC or ATP, Oracle Data Integrator users can benefit from the native integration between Oracle Autonomous Data Warehouse and Oracle Object Storage to enable extremely fast data transfer into ADWC or ATP. ODI can then automate the complete loading process of Oracle Autonomous Data Warehouse Cloud (ADWC) and Oracle Autonomous Transaction Processing (ATP).

For more details, see Oracle Autonomous Data Warehouse Cloud and Oracle Autonomous Transaction Processing Documentation.

Oracle Enterprise Resource Planning (ERP) Cloud

This release also adds a new Technology and Knowledge Modules for Oracle Enterprise Resource Planning (ERP) Cloud, 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 and helps organizations integrate their ERP data into their data warehouses, data marts or data lakes. This native integration also allows ODI users to load data into Oracle ERP Cloud.

For more details, see Oracle Enterprise Resource Planning Cloud Documentation.

New and Changed Features for Release 12*c* (12.2.1.3.0)

Oracle Data Integrator 12*c* (12.2.1.3.0) introduces the following enhancements:

- Big Data
- Cloud

Cloud and Big Data remain key investment areas and ensure that Oracle Data Integrator will continue to accompany customers throughout their technological transformation and modernization process.

Big Data

Big Data continues to evolve within Oracle Data Integrator with advances including:

- **Spark Knowledge Modules Improvements** — In this release, the focus has been on generating high performing and easily readable Spark code that will stand up to any handwritten scripts. In addition, the Spark Knowledge Modules (KMs) are now leveraging the latest Apache Spark 2.x features such as Dataframes. SparkSQL is also leveraged where applicable to speed up the ODI processes running on Spark.
- **Spark KMs Support in Knowledge Module Editor** — The Spark KMs are now fully supported in the Knowledge Module editor and can be tailored to your own specific requirements.
- **Hadoop Complex Types Enhancements** — ODI 12.2.1.2.6 introduced Complex Types support for Hive and HDFS (batch mode) technologies. ODI 12.2.1.3.0 further improves this functionality for Spark Streaming with Complex Types support in Apache Kafka as well as in HDFS.
- **Big Data Configuration Wizard** — The Big Data Configuration Wizard has been improved with newer templates for the latest Cloudera's Distribution including Apache Hadoop (CDH) releases.

Cloud

Cloud improvements include:

- **Certification with Salesforce.com** — Oracle Data Integrator is fully certified with Salesforce.com and now includes a JDBC driver for this technology out of the box.

New and Changed Features for Release 12*c* (12.2.1.2.6)

Oracle Data Integrator 12*c* (12.2.1.2.6) introduces the following enhancements:

- Big Data
- Cloud
- Lifecycle Management
- Developer Productivity

Big Data

It includes:

1. **Spark Streaming Support** — Oracle Data Integrator (ODI) now supports Spark Streaming to fully enable the creation of Big Data streaming jobs easily without requiring end users to write a single line of code. In addition to Spark Streaming ODI already supports Hive, Pig and batch Spark when it comes to data processing. Through its unique decoupling of the Logical and Physical design of Mappings, Oracle Data Integrator is the only Data Integration tool on the market giving developers the flexibility to design Mappings with a generic business logic and then generate code for as many data processing technologies (Hive, Spark, Spark Streaming, etc.) as they want. This unique capability also helps future-proof Data Integration processes.
2. **Support for Apache Kafka and Apache Cassandra** — Apache Kafka and Cassandra are certified with the latest version of Oracle Data Integrator as both sources and targets.
3. **Hadoop Complex Types and Storage Format**— This release further extends the market leading Hadoop support in ODI with the ability to natively access data stored in various formats such as Avro, Parquet or JSON. In addition, new features were added to leverage complex types or nested types in Mappings such as Array, Struct or Map.
4. **Enhancements to Big Data Configuration Wizard**— The Big Data Configuration Wizard introduced with ODI 12.2.1.1.0 has been improved to support new Hadoop technologies such as Kafka and Cassandra. It also now helps users configure Oracle Data Integrator with Hadoop clusters secured with Kerberos.

Cloud

It includes:

1. **RESTful Service Support**— Oracle Data Integrator can now invoke RESTful Service. A RESTful Service connectivity, resource URL, methods and parameters can be configured in Topology configurations like any other data source connectivity. There are a number of parameters supported providing maximum flexibility to support widespread RESTful services. Data chunking and pagination are also supported for uploading or downloading larger payloads.
2. **Business Intelligence Cloud Service (BICS) Knowledge Modules** — Business Intelligence Cloud Service is now supported out of the box in Oracle Data Integrator. You can define Business Intelligence Cloud Service connectivity in Topology, reverse engineer metadata and load data into it just like any other target data server.

Lifecycle Management

It includes:

1. **Git Support and Improvements to Lifecycle Management functionality** — In addition to Apache Subversion, Oracle Data Integrator now also supports Git as an external version control system. A unified user experience is provided when using either of the supported version control systems. There are many advanced operations added for improved lifecycle management needs. You can now view all the pending changes and create version for set of objects from a single place. You can regenerate scenarios while creating versions or creating deployment archives ensuring scenarios always corresponds to the current object version. You can selectively populate objects from the branch or a tag. The SDK APIs are also enhanced to support various operations needed for continuous integration.
2. **Enhanced Merge Capability** — The merge capabilities have been enhanced to auto-merge changes based upon three-way merge and object change detection. Conflict resolution is simplified with brand new support to pick and choose properties or objects from the source or target in the Merge user interface.

Developer Productivity

Enhanced Knowledge Module Framework — Exciting new features have been added to the Knowledge Modules development framework to maximize flexibility and minimize maintenance. You can now inherit steps from a Knowledge Module into another Knowledge Module and override steps like in object-oriented programming languages. Brand new template languages and syntaxes have been introduced providing greater control over the generated code. Furthermore, several other productivity enhancements were added such as syntax highlighting, auto-complete, folding code blocks and more.

New and Changed Features for Release 12*c* (12.2.1.1)

Oracle Data Integrator 12*c* (12.2.1.1) introduces the following enhancements:

- Hyperion Essbase and Hyperion Planning Knowledge Modules
- Integration Capture and Delivery support in GoldenGate Knowledge Modules
- ETL enhancements to support Cubes and Dimensions
- Big Data Configuration Wizard

Hyperion Essbase and Hyperion Planning Knowledge Modules

Hyperion Essbase and Hyperion Planning Knowledge Modules are now available out of the box with Oracle Data Integrator and support the latest version (11.1.2.4) of these Hyperion applications.

For more information, see the "Oracle Hyperion Planning" and "Oracle Hyperion Essbase" sections in *Connectivity and Knowledge Modules Guide for Oracle Data Integrator*.

Integration Capture and Delivery support in GoldenGate Knowledge Modules

The GoldenGate Journalization Knowledge Modules for Oracle databases have been updated and now support Integrated Capture and Delivery. These new capabilities can improve performance and provide better scalability and load balancing.

For more information, see the "Oracle GoldenGate" section in *Connectivity and Knowledge Modules Guide for Oracle Data Integrator*.

ETL enhancements to support Cubes and Dimensions

Oracle Data Integrator introduces support for two types of dimensional objects - Cubes and Dimensions. Users can now create and use Cubes and Dimensions objects directly in Mappings and reduce their development costs with out of the box patterns that automate the loading of dimensional objects.

For more information, see the "Creating and Using Dimensions and Cubes" section in *Developing Integration Projects with Oracle Data Integrator*.

Big Data Configuration Wizard

A brand new Big Data Configuration Wizard is now available in the ODI Studio Gallery and provides a single entry point to configure the Topology objects for Hadoop technologies such as Hadoop, Hive, or Spark.

For more information, see the "Configuring Big Data technologies using the Big Data Configuration Wizard" section in *Integrating Big Data with Oracle Data Integrator*.

New and Changed Features for Release 12*c* (12.2.1)

Oracle Data Integrator 12*c* (12.2.1) introduces the following enhancements:

- Big Data Enhancement: Oozie Execution Modes
- Lifecycle Management of ODI Objects
- ODI Exchange for Sharing Global ODI Objects
- Complex File Enhancements
- Complex File, File, LDAP, JMS Queue XML, JMS Topic XML, and XML Technology Enhancements
- Pre/Post Processing for XML and Complex JDBC Drivers
- Improved Web Service Support
- Ability to Cancel Import/Export and Reverse Engineering Operations
- Support for Analytic or Window Functions
- Oracle Connectivity Enhancements
- Enhanced Integration with Oracle Enterprise Data Quality
- Ability to View a List of Users Connected to Studio/Repository
- ODI Console Enhancements

Big Data Enhancement: Oozie Execution Modes

You can now choose between Task and Session execution modes for Oozie workflow generation. The new Session mode allows support for transactions, scripting, and loops in packages. ODI will automatically choose the correct mode based on the executed object or the mode can also be manually selected.

Please note that Big Data enhancements were made available in ODI 12.1.3.0.1 and improved upon in this release. For more information regarding Big Data enhancements, see "New and Changed Features for Release 12*c* (12.1.3.0.1) ".

Lifecycle Management of ODI Objects

- Oracle Data Integrator is now integrated with Subversion and this provides you with the ability to version control ODI Objects in Subversion. For more information on Subversion, visit the Subversion website.
- Using the Subversion integration capabilities, you can create tags to take a snapshot of ODI object versions. You can create branches for parallel development from distributed locations or for parallel development for multiple releases.
- Release management capabilities are introduced to provide a distinction between the development and deployment environments. You can create Deployment Archives (DA) from a development environment, which can be deployed in a QA environment for testing, and then delivered to the production environment. The DA can be created using ODI Studio or from a command line.

ODI Exchange for Sharing Global ODI Objects

You can now browse, download, and install global ODI objects made available to you by Oracle or other ODI users through Official or Third-Party Update Centers. This feature is available for Global Knowledge Modules, Global User Functions and Mapping Components. The Check for Updates menu item in the Help menu in ODI Studio enables you to connect to the Update Centers and obtain Global ODI Objects.

Complex File Enhancements

The Native Format Builder utility is now included with ODI Studio and allows you to create nXSD files without leaving the ODI user interface.

Complex File, File, LDAP, JMS Queue XML, JMS Topic XML, and XML Technology Enhancements

All JDBC properties for Complex File, File, LDAP, JMS Queue XML, JMS Topic XML, and XML technologies are now displayed at the Data Server level along with default values where applicable and a description of the properties, thereby enhancing usability.

Pre/Post Processing for XML and Complex JDBC Drivers

You can now customize the way data is fed into XML and Complex File drivers. This feature adds support for intermediate processing stages that may be added for processing data as it has either been retrieved from the external endpoint using Oracle Data Integrator, or it is to be written out to an external endpoint. This feature also provides support for complex configuration of intermediate processing stages as part of the configuration of data servers that use ODI XML or Complex File JDBC drivers.

Improved Web Service Support

A new SOAP Web Service technology is now available in Topology and allows the creation of data servers, physical schemas, and logical schemas for Web Services. Oracle Web Service Management (OWSM) policies can also be attached to Web Services data servers. In addition, the OdiInvokeWebService tool is enhanced to support Web Services data servers through Contexts and Logical Schemas.

Ability to Cancel Import/Export and Reverse Engineering Operations

You can now cancel import/export and reverse-engineering operations that may run for a long time.

Support for Analytic or Window Functions

Analytic or Window functions are now supported out of the box at the Mapping level. Analytic functions such as PERCENT_RANK, LAST, FIRST, or LAG can be used at the Mapping Expression level in any component.

Oracle Connectivity Enhancements

A new Knowledge Module to perform Partition Exchange Loading is now available allowing you to swap partitions when needed. In addition, improvements have been made to Loading Knowledge Modules using External Tables, which can now load more than one file at a time. Knowledge Modules using Data Pump have also been improved.

Enhanced Integration with Oracle Enterprise Data Quality

A new Oracle Enterprise Data Quality (EDQ) technology is available in Topology and allows the creation of data servers, physical schemas, and logical schemas for EDQ. Also, the OdiEnterpriseDataQuality tool is enhanced to support EDQ data servers through Contexts and Logical Schemas.

Ability to View a List of Users Connected to Studio/Repository

The Review User Activity menu item has been added to the Security menu. Using this you can view, purge, and save user activity record in the User Connections dialog. This feature is available in ODI Studio and ODI Console.

ODI Console Enhancements

The overall look and feel of ODI Console has been improved. Security tasks such as creating users or profiles can now be performed in ODI Console. Also, Release Management activities can now be performed in ODI Console and the functionality related to Topology activities has been enhanced.

New and Changed Features for Release 12*c* (12.1.3.0.1)

Oracle Data Integrator 12*c* (12.1.3.0.1) introduces the following enhancements:

- Execution of ODI Mappings using Spark and Pig
- Orchestration of ODI Jobs using Oozie
- Enhanced Hive Driver and Knowledge Modules
- Retrieval of Hadoop Audit Logs
- HDFS access in ODI File Tools
- Flatten and Jagged Components
- New Big Data Guide added to the ODI Documentation Set

Execution of ODI Mappings using Spark and Pig

ODI allows the defining of mappings through a logical design, which is independent of the implementation language. For Hadoop-based transformations, you can select between Hive, Spark, and Pig as the generated transformation language. This allows you to pick the best implementation based on the environment and use case; you can also choose different implementations simultaneously using multiple physical designs. This selection makes development for Big Data flexible and future-proof.

- **Generate Pig Latin transformations:** You can choose Pig Latin as the transformation language and execution engine for ODI mappings. Apache Pig is a platform for analyzing large data sets in Hadoop and uses the high-level language, Pig Latin for expressing data analysis programs. Any Pig transformations can be executed either in Local or MapReduce mode. Custom Pig code can be added through user-defined functions or the table function component.
- **Generate Spark transformations:** ODI mapping can also generate PySpark, which exposes the Spark programming model in the Python language. Apache Spark is a transformation engine for large-scale data processing. It provides fast in-memory processing of large data sets. Custom PySpark code can be added through user-defined functions or the table function component.

Orchestration of ODI Jobs using Oozie

You can now choose between the traditional ODI Agent or Apache Oozie as the orchestration engine for ODI jobs such as mappings, packages, scenarios, and procedures. Apache Oozie allows a fully native execution on a Hadoop infrastructure without installing an ODI environment for orchestration. You can utilize Oozie tools to schedule, manage, and monitor ODI jobs. ODI uses Oozie's native actions to execute Hadoop processes and conditional branching logic.

Enhanced Hive Driver and Knowledge Modules

ODI includes the WebLogic Hive JDBC driver that provides a number of advantages when compared to the Apache Hive driver, such as, total JDBC compliance and improved performance. All Hive Knowledge Modules have been rewritten to benefit from this new driver. Also, the Knowledge Modules whose main purpose is to load from a source are now provided as Load Knowledge Modules, enabling them to be combined in a single mapping with other Load Knowledge Modules. A new class of "direct load" Load Knowledge Modules also allows the loading of targets without intermediate staging. The table function component has been extended to support Hive constructs.

Retrieval of Hadoop Audit Logs

ODI integrates results from Hadoop Audit Logs in Operator tasks for executions of Oozie, Pig, and other tasks. The log results show MapReduce statistics and provide a link to Hadoop statistics in native web consoles.

HDFS access in ODI File Tools

The file based tools used in ODI packages and procedures have been enhanced to include Hadoop Distributed File System (HDFS) file processing. This includes copying, moving, appending, and deleting files, detecting file changes, managing folders, and transferring files using FTP directly into HDFS.

Flatten and Jagged Components

The new Flatten component for mappings allows complex sub-structures to be processed as part of a flat list of attributes. The new Jagged component converts key-value lists into named attributes for further processing.

New Big Data Guide added to the ODI Documentation Set

A new guide, *Integrating Big Data with Oracle Data Integrator*, has been added to the ODI documentation set. This guide provides information on how to integrate Big Data, deploy and execute Oozie workflows, and generate code in languages such as Pig Latin and Spark.

New and Changed Features for Release 12*c* (12.1.3)

Oracle Data Integrator 12*c* (12.1.3) introduces the following enhancements:

- ODI FIPS Compliance
- ODI XML Driver Enhancements
- JSON Support
- Hadoop SQOOP Integration
- Hadoop HBase Integration
- Hive Append Optimization
- Multi-threaded Target Table Load in ODI Engine
- Improved Control for Scenario and Load Plan Concurrent Execution
- Create New Model and Topology Objects
- Documentation Changes

ODI FIPS Compliance

ODI now uses Advanced Encryption Standard (AES) as the standard encryption algorithm for encrypting Knowledge Modules, procedures, scenarios, actions, and passwords. You can configure the encryption algorithm and key length to meet requirements. Passwords and other sensitive information included in repository exports are now encrypted and secured by a password.

For more information, see "Advanced Encryption Standard".

ODI XML Driver Enhancements

The following XML Schema support enhancements have been added:

- Recursion: ODI now supports recursion inside XML Schemas.
- `any`, `anyType`, and `anyAttribute`: Data defined by these types is stored in string type columns with XML markup from the original document.
- Metadata annotations can be added inside an XML Schema to instruct the ODI XML Driver which table name, column name, type, length, and precision should be used.

For more information, see the "Oracle Data Integrator Driver for XML Reference" section in *Connectivity and Knowledge Modules Guide for Oracle Data Integrator Developer's Guide*.

JSON Support

The ODI Complex File Driver can now read and write files in JSON format. The JSON structure is defined through an nXSD schema.

For more information, see the "JSON Support" section in *Connectivity and Knowledge Modules Guide for Oracle Data Integrator Developer's Guide*.

Hadoop SQOOP Integration

ODI can now load the following sources and targets using Hadoop SQOOP:

- From relational databases to HDFS, Hive, and HBase through Knowledge Module IKM File-Hive to SQL (SQOOP)
- From HDFS and Hive to relational databases through Knowledge Module IKM SQL to Hive-HBase-File (SQOOP)

SQOOP enables load and unload mechanisms using parallel JDBC connections in Hadoop Map-Reduce processes.

Hadoop HBase Integration

ODI now supports Hadoop HBase through a new technology and the following knowledge modules:

- LKM HBase to Hive (HBase-SerDe)
- IKM Hive to HBase Incremental Update (HBase-SerDe)
- RKM HBase

Hive Append Optimization

Knowledge Modules writing to Hive now support the Hive 0.8+ capability and can append data to the existing data files rather than copying existing data into a new appended file.

Multi-threaded Target Table Load in ODI Engine

ODI can now load a target table using multiple parallel connections. This capability is controlled through the Degree of Parallelism for Target property in the data server.

For more information, see "Creating a Data Server".

Improved Control for Scenario and Load Plan Concurrent Execution

You can now limit concurrent executions in a scenario or load plan and force a concurrent execution to either wait or raise an execution error.

For more information, see the "Controlling Concurrent Execution of Scenarios and Load Plans" section in *Developing Integration Projects with Oracle Data Integrator*.

Create New Model and Topology Objects

The Create New Model and Topology Objects dialog in the Designer Navigator provides the ability to create a new model and associate it with new or existing topology objects, if connected to a work repository. This dialog enables you to create topology objects without having to use Topology editors unless more advanced options are required.

For more information, see the "Creating a Model and Topology Objects" section in *Developing Integration Projects with Oracle Data Integrator*.

Documentation Changes

The information that was previously available in the Oracle Data Integrator Developer's Guide is now reorganized. The following new guides have been added to the ODI documentation library:

- *Understanding Oracle Data Integrator*
- *Administering Oracle Data Integrator*
- *Oracle Data Integrator Tools Reference*

For more information, see the "What's New In Oracle Data Integrator?" section in *Developing Integration Projects with Oracle Data Integrator*.

New and Changed Features for Release 12*c* (12.1.2)

Oracle Data Integrator 12*c* (12.1.2) introduces the following enhancements:

- Declarative Flow-Based User Interface
- Reusable Mappings
- Multiple Target Support
- Step-by-Step Debugger
- Runtime Performance Enhancements
- Oracle GoldenGate Integration Improvements
- Standalone Agent Management with WebLogic Management Framework
- Integration with OPSS Enterprise Roles
- XML Improvements
- Oracle Warehouse Builder Integration
- Unique Repository IDs

Declarative Flow-Based User Interface

The new declarative flow-based user interface combines the simplicity and ease-of-use of the declarative approach with the flexibility and extensibility of configurable flows. Mappings (the successor of the Interface concept in Oracle Data Integrator 11*g*) connect sources to targets through a flow of components such as Join, Filter, Aggregate, Set, Split, and so on.

Reusable Mappings

Reusable Mappings can be used to encapsulate flow sections that can then be reused in multiple mappings. A reusable mapping can have input and output signatures to connect to an enclosing flow; it can also contain sources and targets that are encapsulated inside the reusable mapping.

Multiple Target Support

A mapping can now load multiple targets as part of a single flow. The order of target loading can be specified, and the Split component can be optionally used to route rows into different targets, based on one or several conditions.

Step-by-Step Debugger

Mappings, Packages, Procedures, and Scenarios can now be debugged in a step-by-step debugger. You can manually traverse task execution within these objects and set breakpoints to interrupt execution at pre-defined locations. Values of variables can be introspected and changed during a debugging session, and data of underlying sources and targets can be queried, including the content of uncommitted transactions.

Runtime Performance Enhancements

The runtime execution has been improved to enhance performance. Various changes have been made to reduce overhead of session execution, including the introduction of blueprints, which are cached execution plans for sessions.

Performance is improved by loading sources in parallel into the staging area. Parallelism of loads can be customized in the physical view of a map.

You also have the option to use unique names for temporary database objects, allowing parallel execution of the same mapping.

Oracle GoldenGate Integration Improvements

The integration of Oracle GoldenGate as a source for the Change Data Capture (CDC) framework has been improved in the following areas:

- Oracle GoldenGate source and target systems are now configured as data servers in Topology. Extract and replicate processes are represented by physical and logical schemas. This representation in Topology allows separate configuration of multiple contexts, following the general context philosophy.
- Most Oracle GoldenGate parameters can now be added to extract and replicate processes in the physical schema configuration. The UI provides support for selecting parameters from lists. This minimizes the need for the modification of Oracle GoldenGate parameter files after generation.
- A single mapping can now be used for journalized CDC load and bulk load of a target. This is enabled by the Oracle GoldenGate JKM using the source model as opposed to the Oracle GoldenGate replication target, as well as configuration of journalizing in mapping as part of a deployment specification. Multiple deployment specifications can be used in a single mapping for journalized load and bulk load.
- Oracle GoldenGate parameter files can now be automatically deployed and started to source and target Oracle GoldenGate instances through the JAgent technology.

Standalone Agent Management with WebLogic Management Framework

Oracle Data Integrator Standalone agents are now managed through the WebLogic Management Framework. This has the following advantages:

- UI-driven configuration through Configuration Wizard
- Multiple configurations can be maintained in separate domains
- Node Manager can be used to control and automatically restart agents

Integration with OPSS Enterprise Roles

Oracle Data Integrator can now use the authorization model in Oracle Platform Security Services (OPSS) to control access to resources. Enterprise roles can be

mapped into Oracle Data Integrator roles to authorize enterprise users across different tools.

XML Improvements

The following XML Schema constructs are now supported:

- list and union - List or union-based elements are mapped into VARCHAR columns.
- substitutionGroup - Elements based on substitution groups create a table each for all types of the substitution group.
- Mixed content - Elements with mixed content map into a VARCHAR column that contains text and markup content of the element.
- Annotation - Content of XML Schema annotations are stored in the table metadata.

Oracle Warehouse Builder Integration

Oracle Warehouse Builder (OWB) jobs can now be executed in Oracle Data Integrator through the `OdiStartOwbJob` tool. The OWB repository is configured as a data server in Topology. All the details of the OWB job execution are displayed as a session in the Operator tree.

Unique Repository IDs

Master and work repositories now use unique IDs following the GUID convention. This avoids collisions during import of artifacts and allows for easier management and consolidation of multiple repositories in an organization.

1

Introduction to Oracle Data Integrator

Oracle Data Integrator provides a fully unified solution for building, deploying, and managing complex data warehouses or as part of data-centric architectures in an SOA or business intelligence environment. This chapter provides an introduction to Oracle Data Integrator and its various components.
This chapter includes the following sections:

What is Oracle Data Integrator?

Oracle Data Integrator (ODI) features an active integration platform that includes all styles of data integration: data-based, event-based and service-based. ODI unifies silos of integration by transforming large volumes of data efficiently, processing events in real time through its advanced Changed Data Capture (CDC) framework, and providing data services to the Oracle SOA Suite. It also provides robust data integrity control features, assuring the consistency and correctness of data. With powerful core differentiators - heterogeneous E-LT, Declarative Design and Knowledge Modules - Oracle Data Integrator meets the performance, flexibility, productivity, modularity and hot-pluggability requirements of an integration platform.

Oracle Data Integrator Components

The Oracle Data Integrator platform integrates in the broader Fusion Middleware platform and becomes a key component of this stack. Oracle Data Integrator provides its run-time components as Java EE applications, enhanced to fully leverage the WebLogic Application Server. Oracle Data Integrator components include exclusive features for Enterprise-Scale Deployments, high availability, scalability, and hardened security. The ODI architecture relies on the following components that collaborate together:

- Repositories
- ODI Studio and User Interfaces
- Design-time Projects
- Run-time Agent
- ODI Console
- Oracle Enterprise Manager Fusion Middleware Control
- Management Pack for Oracle Data Integrator

For more information regarding ODI components see the Understanding the Oracle Data Integrator Component Architecture section in *Understanding Oracle Data Integrator*.

Basic Tasks for Configuring and Managing Oracle Data Integrator

The following provides a summary of the steps you need to take to configure and manage a basic Oracle Data Integrator environment after you have installed the software:

Table 1-1 Roadmap for Configuring and Managing Oracle Data Integrator

Task	Documentation
1. Create a master and work repository schema	See the Creating the Oracle Data Integrator Master and Work Repository Schema section in *Installing and Configuring Oracle Data Integrator*.
2. Create additional work repositories	See Creating a Work Repository.
3. Configure a domain for Oracle Data Integrator agents	• Java EE agent: See the Configuring the WebLogic Domain for the Java EE Agent section in *Installing and Configuring Oracle Data Integrator*. • Standalone agent: See the Configuring the Standalone Domain for the Standalone Agent section in *Installing and Configuring Oracle Data Integrator*. • Standalone Colocated agent: See the Configuring the WebLogic Domain for the Standalone Colocated Agent section in *Installing and Configuring Oracle Data Integrator*.
4. Create an agent	• Java EE agent: See the Creating a Java EE Agent in the Master Repository Using ODI Studio section in *Installing and Configuring Oracle Data Integrator*. • Standalone agent: See the Creating a Standalone Agent in the Master Repository Using ODI Studio section in *Installing and Configuring Oracle Data Integrator*. • Standalone Colocated agent: See the Creating a Standalone Colocated Agent in the Master Repository Using ODI Studio section in *Installing and Configuring Oracle Data Integrator*.
5. Start an agent	• Java EE agent: See the Starting a Java EE Agent section in *Installing and Configuring Oracle Data Integrator*. • Standalone agent: See the Starting a Standalone Agent Using Node Manager section in *Installing and Configuring Oracle Data Integrator*. • Standalone Colocated agent: See the Starting a Standalone Colocated Agent section in *Installing and Configuring Oracle Data Integrator*.
6. Configure load balancing on a Standalone agent	See Load Balancing Agents
7. Configure high availability for Oracle Data Integrator	See Roadmap for Setting Up a High Availability Topology .
8. Configure security	See Managing Security in Oracle Data Integrator.
9. Set up a topology	See Setting Up a Topology.
10. Set up data services	See Creating and Using Data Services .

Table 1-1 (Cont.) Roadmap for Configuring and Managing Oracle Data Integrator

Task	Documentation
11. Run, debug, and monitor integration processes	See Running and Monitoring Integration Processes .
12. Use Enterprise Manager to manage ODI environment	See ODI Domain, Oracle Enterprise Manager Fusion Middleware Control, and Management Pack for Oracle Data Integrator.
13. Stop an agent	See the Stopping Your Oracle Data Integrator Agents section in *Installing and Configuring Oracle Data Integrator*.

Part I

Administering Oracle Data Integrator Architecture

This part describes the Oracle Data Integrator architecture and a description of Oracle Data Integrator components from a High Availability perspective.

Part I contains the following chapters:

- Administering Repositories
- Setting Up a Topology
- High Availability for Oracle Data Integrator

2

Administering Repositories

This chapter describes how to create and administer Oracle Data Integrator repositories. An overview of the repositories used in Oracle Data Integrator is provided.
This chapter includes the following sections:

> **See Also:**
>
> - For more information about creating ODI repositories in RCU, see *Installing and Configuring Oracle Data Integrator*.
> - For more information about upgrading ODI repositories, see *Upgrading Oracle Data Integrator*.

Introduction to Oracle Data Integrator Repositories

There are two types of repositories in Oracle Data Integrator:

- **Master Repository:** This is a data structure containing information on the topology of the company's IT resources, on security and on version management of projects and data models. This repository is stored on a relational database accessible in client/server mode from the different Oracle Data Integrator modules. In general, you need only one master repository. However, it may be necessary to create several master repositories in one of the following cases:
  - Project construction over several sites not linked by a high-speed network (off-site development, for example).
  - Necessity to clearly separate the operating environments (development, test, production), including on the database containing the master repository. This may be the case if these environments are on several sites.
- **Work Repository:** This is a data structure containing information about data models, projects, and their use. This repository is stored on a relational database accessible in

client/server mode from the different Oracle Data Integrator modules. Several work repositories can be created with several master repositories if necessary. However, a work repository can be linked with only one master repository for version management purposes.

The standard method for creating repositories is using the Repository Creation Utility (RCU). The RCU automatically manages storage space as well as repository creation. However, if you want to create repositories manually, it is possible to manually create and configure the repositories using ODI Studio.

The steps needed to create and configure repositories are detailed in the following sections:

- Creating Repository Storage Spaces
- Creating the Master Repository
- Connecting to the Master Repository
- Creating a Work Repository
- Connecting to a Work Repository

> **Note:**
>
> Oracle recommends that you regularly perform the following maintenance operations: purge the execution logs in order to reduce the work repository size, and back up the Oracle Data Integrator repositories on the database.

Advanced actions for administering repositories are detailed in Advanced Actions for Administering Repositories .

Creating Repository Storage Spaces

Oracle Data Integrator repositories can be installed on database engines supported by Oracle Fusion Middleware 12*c*. For the latest list of supported databases versions as well as the requirements for each database, see:

`http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html`

For each database that will contain a repository, a storage space must be created.

> **Caution:**
>
> For reasons of maintenance and back-up, Oracle strongly recommends that repositories be stored separately from your application data (for example, in a different schema for an Oracle database, or in a different database for Microsoft SQL Server).

Your master repository can be stored in the same schema as one of your work repositories. However, you cannot create two different work repositories in the same schema.

The examples in the following table are supplied as a guide:

Technology	Steps to follow

Oracle	Create a schema `odim` to host the master repository and a schema `odiw` to host the work repository.
	The schemas are created by the following SQL commands:
	`SQL> create user MY_SCHEMA identified by MY_PASS`
	`default tablespace MY_TBS`
	`temporary tablespace MY_TEMP;`
	`SQL> grant connect, resource to MY_SCHEMA;`
	`SQL> grant execute on dbms_lock to MY_SCHEMA;`
	Where:
	MY_SCHEMA corresponds to the name of the schema you want to create, such as *odim* and *odiw*
	MY_PASS corresponds to the password you have given it <MY_TBS> the Oracle tablespace where the data will be stored
	MY_TEMP temporary default tablespace

Microsoft SQL Server	Create a database `db_odim` to host the master repository and a database `db_odiw` to host the work repository. Create two logins `odim` and `odiw`, which have these databases by default.
	Use Oracle Enterprise Manager to create the two databases *db_odim* and *db_odiw*.
	Use Query Analyzer or I-SQL to launch the following commands:
	`CREATE LOGIN mylogin`
	`WITH PASSWORD = 'mypass',`
	`DEFAULT_DATABASE = defaultbase,`
	`DEFAULT_LANGUAGE = us_english;`
	`USE defaultbase;`
	`CREATE USER dbo FOR LOGIN mylogin;`
	`GO`
	Where:
	mylogin corresponds to odim or odiw.
	mypass corresponds to a password for these logins.
	defaultbase corresponds to db_odim and db_odiw respectively.
	Note: Oracle recommends configuring the Microsoft SQL Server databases that store the repository information with a case-sensitive collation. This enables reverse-engineering and creating multiple objects with the same name but a different case (for example: *tablename* and *TableNAme*).

Technology	Steps to follow

Sybase ASE	Create a database `db_odim` to host the master repository and a database `db_odiw` to host the work repository. Create two logins `odim` and `odiw` which have these databases by default.
	Use I-SQL to launch the following commands:
	`isql -Usa -P<sa password> -S<Server>`
	`use master`
	`go`
	`disk init`
	`name = "<Device Name>",`
	`physname = "<Actual path>",`
	`size = <Size>`
	`go`
	`create database <database name>`
	`on <Device Name>='<Size>'`
	`log on <Device Name>='<Size>' WITH OVERRIDE`
	`go`
	`use <database name>`
	`go`
	`sp_addlogin '<DB USERNAME>', '<password>', <database name>`
	`go`
	`sp_adduser '<DB USERNAME>'`
	`go`
	`grant CREATE TABLE to <DB USERNAME>`
	Note: Use a different device for log and data. Also, `<DB USERNAME` should be in uppercase.

MySQL	1. Login to MySQL terminal.
	2. Create databases for Master, Work, and Schema version registries, as shown below:
	`create database <MASTER_DB>;`
	`create database <WORK_DB>;`
	`create database <SCHEME_VERSION_REGISTRY_DB>;`
	3. Create users, as shown below:
	`create user '<MASTER_DB>'@'localhost' identified by '<password>';`
	`create user '<WORK_DB>'@'localhost' identified by '<password>';`
	`create user '<MASTER_DB>'@'%' identified by '<password>';`
	`create user '<WORK_DB>'@'%' identified by '<password>';`
	`commit;`
	Note: The user name should be in uppercase.
	4. Grant privileges to the user, as shown below:
	`grant all privileges on <MASTER_DB>.* to <MASTER_DB>;`
	`grant all privileges on <MASTER_DB>.* to <MASTER_DB>@localhost;`
	`grant all privileges on <MASTER_DB>.* to <MASTER_DB>@'%';`
	`grant all privileges on <WORK_DB>.* to <WORK_DB>;`
	`grant all privileges on <WORK_DB>.* to <WORK_DB>@localhost;`
	`grant all privileges on <WORK_DB>.* to <WORK_DB>@'%';`

Technology	Steps to follow
DB2/400	Create a library *odim* to host the master repository and a schema *odiw* to host the work repository. Create two users *odim* and *odiw* who have these libraries by default. **Note**: The libraries must be created in the form of SQL collections.

DB2/UDB	**Prerequisites**:
	• Master and work repository users must have access to tablespaces with minimum 16k pagesize
	• The database must have a temporary tablespace with minimum 16 k
	For example:
	`CREATE LARGE TABLESPACE ODI16 PAGESIZE 16 K MANAGED BY AUTOMATIC STORAGE ;`
	`GRANT USE OF TABLESPACE ODI16 TO USER ODIREPOS;`

Creating the Master Repository

Creating the master repository creates an empty repository structure and seeds metadata (for example, technology definitions, or built-in security profiles) into this repository structure.

To create the master repository:

1. In ODI Studio, open the New Gallery dialog by choosing **File** > **New**.
2. In the New Gallery dialog, in the Categories tree, select **ODI**.
3. Select from the Items list the **Master Repository Creation Wizard**.
4. Click **OK**.

   The Master Repository Creation Wizard opens.
5. Specify the **Database Connection** parameters as follows:
   - **Technology**: From the list, select the technology that will host your master repository. Default is *Oracle*.
   - **JDBC Driver**: The driver used to access the technology, that will host the repository.
   - **JDBC Url**: The URL used to establish the JDBC connection to the database.

     Note that the parameters **JDBC Driver** and **JDBC Url** are synchronized and the default values are technology dependent.
   - **User**: The user ID / login of the owner of the tables (for example, `odim`).

     Note that the user name should be in uppercase to ensure compliance with the Upgrade Assistant and avoid issues when upgrading the repository.
   - **Password**: This user's password.
   - **DBA User**: The database administrator's username.
   - **DBA Password**: This user's password.
6. Specify **Repository Configuration** parameters as needed.
7. Click **Test Connection** to test the connection to your master repository.

The Information dialog opens and informs you whether the connection has been established. If the connection fails, fix the connection to your master repository before moving to next step.

8. Click **Next**.
9. Do one of the following:
   - Select **Use ODI Authentication** to manage users using ODI's internal security system, and enter the following supervisor login information:

Properties	Description
Supervisor User	User name of the ODI supervisor. The value must be `SUPERVISOR`.
Supervisor Password	This user's password
Confirm Password	This user's password

   - Select **Use External Authentication** to use an external enterprise identity store, such as Oracle Internet Directory, to manage user authentication, and enter the following supervisor login information:

Properties	Description
Supervisor User	User name of the ODI supervisor
Supervisor Password	This user's password

> **Note:**
>
> In order to use the external authentication option, ODI Studio has to be configured for external authentication. See Configuring External Authentication for more information.

10. Click **Next**.
11. Specify the password storage details for your data servers (that is, sources and targets, which are defined in the Topology):
    - Select **Internal Password Storage** if you want to store passwords in the Oracle Data Integrator master repository
    - Select **External Password Storage** if you want use JPS Credential Store Framework (CSF) to store the data server and context passwords in a remote credential store. Indicate the **MBean Server Parameters** to access the credential store. Refer to Managing Security in Oracle Data Integrator for more information.
12. In the Master Repository Creation Wizard click **Finish** to validate your entries.

Oracle Data Integrator begins creating your master repository. You can follow the procedure on your Messages – Log. To test your master repository, refer to Connecting to the Master Repository.

> **Note:**
>
> By default, the master repository uses AES-128 for encryption. However, if you want to create a master repository with the encryption algorithm AES-256, you need to update the `odi.conf` file to include the following:
>
> AddVMOption: `Doracle.odi.encryption.algo=AES` AddVMOption: `Doracle.odi.encryption.keylen=256`
>
> For information regarding AES, see Advanced Encryption Standard.

Connecting to the Master Repository

To connect to the master repository, follow the instructions in the Connecting to the Master Repository section in *Installing and Configuring Oracle Data Integrator*.

Creating a Work Repository

A master repository can have one or more work repositories associated with it; a work repository, however, can only be associated with one master repository.

To create a new work repository:

1. In the Topology Navigator, open the **Repositories** panel.
2. Right-click the Work Repositories node and select **New Work Repository**.

   The **Work Repository Creation Wizard** opens.
3. Specify the Oracle Data Integrator work repository connection details as follows:
   - **Technology:** Choose the technology of the server to host your work repository. Default is *Oracle*.
   - **JDBC Driver**: The driver used to access the technology, that will host the repository.
   - **JDBC Url:** The complete path of the data server to host the work repository.

     Note that the parameters **JDBC Driver** and **JDBC Url** are synchronized and the default values are technology dependent.

     Oracle recommends using the full machine name instead of `localhost` in the JDBC URL to avoid connection issues. For example, for remote clients, the client (ODI Studio or SDK) is on a different machine than the work repository and `localhost` points to the current client machine instead of the one hosting the work repository.
   - **User:** User ID / login of the owner of the tables you are going to create and host of the work repository.

     Note that the user name should be in uppercase to ensure compliance with the Upgrade Assistant and avoid issues when upgrading the repository.
   - **Password:** This user's password.
4. Click **Test Connection** to verify that the connection is working.
5. Click **Next.**

Oracle Data Integrator verifies whether a work repository already exists on the connection specified in step 3:

- If an existing work repository is detected on this connection, the next steps will consist in attaching the work repository to the master repository. Refer to step 6 of Attaching and Detaching (Deleting) a Work Repository for further instructions.
- If no work repository is detected on this connection, a new work repository is created. Continue with the creation of a new work repository and provide the work repository details in step 6.

6. Specify the Oracle Data Integrator work repository properties:
   - **Name**: Give a unique name to your work repository (for example: *DEVWORKREP1*).
   - **Password**: Optional. Enter a password required for attaching this work repository to a different master. If you leave this option blank, no password is required for this operation.
   - **Type**: Select the type for the work repository:
     - **Development**: This type of repository allows management of design-time objects such as data models and projects (including mappings, procedures, and so on). A development repository also includes the run-time objects (scenarios and sessions). This type of repository is suitable for development environments.
     - **Execution**: This type of repository only includes run-time objects (scenarios, schedules and sessions). It allows launching and monitoring of data integration jobs in Operator Navigator. Such a repository cannot contain any design-time artifacts. Designer Navigator cannot be used with it. An execution repository is suitable for production environments.
7. Click **Finish**.
8. The Create Work Repository login dialog opens. If you want to create a login for the work repository, click **Yes** and you will be asked to enter the **Login Name** in a new dialog. If you do not want to create a work repository login, click **No**.
9. Click **Save** in the toolbar.

For more information, refer to Connecting to a Work Repository.

Connecting to a Work Repository

To connect to a work repository, click **Connect to repository** in ODI Studio, and enter the credentials you specified in Step 8 of Creating a Work Repository.

If you did not create a work repository Login Name in Step 8 of Creating a Work Repository, in order to connect, you must create a login by performing the following steps:

1. In ODI Studio, open the New Gallery dialog by choosing **File** > **New**.
2. In the New Gallery dialog, in the Categories tree, select **ODI**.
3. Select from the Items list **Create a New ODI Repository Login**.
4. Click **OK**.

   The Repository Connection Information dialog opens.

5. Specify the Oracle Data Integrator connection details as follows:
   - **Login Name**: A generic alias (for example: *Repository*)
   - **User**: The ODI supervisor user name you have defined when creating the master repository or an ODI user name you have defined in the Security Navigator after having created the master repository.
   - **Password**: The ODI supervisor password you have defined when creating the master repository or an ODI user password you have defined in the Security Navigator after having created the master repository.
6. Specify the Database Connection (master repository) details as follows:
   - **User**: Database user ID/login of the schema (database, library) that contains the ODI master repository.
   - **Password**: This user's password.
   - **Driver List**: Select the driver required to connect to the DBMS supporting the master repository you have just created from the drop-down list.
   - **Driver Name**: The complete driver name.
   - **URL**: The URL used to establish the JDBC connection to the database hosting the repository.
7. Click **Test** to check the connection is working.
8. Select **Work Repository** and specify the work repository details as follows:
   - **Work repository name:** The name you gave your work repository in the previous step (*WorkRep1* in the example). You can display the list of work repositories available in your master repository by clicking on the button to the right of this field.
9. Click **OK** to validate your entries.

Changing a Work Repository Password

To change a work repository password:

1. In the Repositories tree of the Topology Navigator, expand the Work Repositories node.
2. Double-click a work repository, or right-click and select **Open**. The Work Repository Editor opens.
3. On the Definition tab of the Work Repository Editor, click **Change password**.
4. Enter the current password, and enter the new password twice.
5. Click **OK**.

Advanced Actions for Administering Repositories

Advanced actions for administering repositories do not concern the creation process of repositories. The actions described in this section deal with advanced actions performed on already existing repositories. Once the repositories are created you may want to switch the password storage or you may need to recover the password storage after a credential store crash. Actions dealing with password handling are covered in Setting Up External Password Storage. For information regarding the export and import of master and work repositories, see the Exporting and Importing chapter in *Developing Integration Projects with Oracle Data Integrator*.

This section contains the following topics:

Attaching and Detaching (Deleting) a Work Repository

Attaching a work repository consists of linking an existing work repository to the current master repository. This existing work repository already exists in the database and has been previously detached from this or another master repository.

Deleting a work repository detaches it, by deleting its link to the master repository. This is an opposite operation to attaching. This operation does not destroy the work repository content.

Attaching a Work Repository

To attach a work repository to a master repository:

1. In the Topology Navigator, go to the **Repositories** panel.
2. Right-click the Work Repositories node and select **New Work Repository**.

   The **Work Repository Creation Wizard** opens.
3. Specify the Oracle Data Integrator work repository connection details as follows:
   - **Technology**: From the list, select the technology that will host your work repository. Default is *Oracle*.
   - **JDBC Driver**: The driver used to access the technology, that will host the repository.
   - **JDBC Url**: The complete path of the data server to host the work repository.

     Note that the parameters **JDBC Driver** and **JDBC Url** are synchronized and the default values are technology dependent
   - **User**: User ID / login of the owner of the tables you are going to create and host of the work repository.
   - **Password**: This user's password.
4. Click **Test Connection** to check the connection is working.
5. Click **Next**.
6. Specify the **Password** of the Oracle Data Integrator work repository to attach.
7. Click **Next**.
8. Specify the **Name** of the Oracle Data Integrator work repository to attach.
9. Click **Finish**.

Deleting a Work Repository

To detach a repository, delete the link from the master repository using the following procedure:

1. In the Topology Navigator, expand the **Repositories** panel.

2. Expand the **Work Repositories** node and right-click the work repository you want to delete.
3. Select **Delete**.
4. In the Confirmation dialog click **Yes**.
5. The work repository is detached from the master repository and is removed from the **Repositories** panel in Topology Navigator.

Erasing a Work Repository

Deleting a work repository is equivalent to detaching a work repository from the master repository. For more information, refer to Attaching and Detaching (Deleting) a Work Repository.

Erasing a work repository consists of deleting the work repository from the database.

> **WARNING:**
>
> Erasing your work repository is an irreversible operation. All information stored in the work repository will be definitively deleted, including the metadata of your models, projects and run-time information such as scenarios, schedules, and logs.

Erasing a Work Repository

To erase a work repository from the database:

1. In the Topology Navigator, expand the **Repositories** panel.
2. Expand the **Work Repositories** node and right-click the work repository you want to delete.
3. Select **Erase from Database**.
4. In the Confirmation dialog click **Yes**, if you want to definitively erase the work repository from the database.
5. The work repository is erased from the database and is deleted from the **Repositories** panel in Topology Navigator.

Configuring Repository Connections

Concurrent connections to the repository database may be controlled and limited by the database engine where the repository is stored. On Oracle the initialization parameter limiting the number of connections is *processes*. When running a large number of parallel executions, you may need to tune the database to increase the maximum number of connections allowed to the repository database.

The number of connections required depends on the number of ODI Studio instances and Load Plan steps running:

- Each ODI Studio session requires two database connections (one to the master, one to the work repository) for the duration of the session, and also a third database connection is required for a security check for a very short period when the session begins.

- For non-Oracle databases, each Load Plan step consumes an additional connection as a lock while the Load Plan is being executed.

Changing the Host Name for Oracle Data Integrator

You can change the host name or IP address of a host that contains Oracle Data Integrator, or the database that contains the Oracle Data Integrator schemas.

1. You use the copyBinary and pasteBinary scripts to copy the Oracle Fusion Middleware binaries from one host to another.

   For information about these scripts, see the Moving an Oracle Fusion Middleware Database to a New Host section of *Administering Oracle Fusion Middleware*.
2. In Target Machine, Connect to the ODI master repository, using ODI Studio.
3. From **Topology Manager,** click **Work Repositories.**
4. Open **Editor** for each **Work Repository.**
5. Click **OK** for **"ODI-23018: Unable to connect to Work Repository."**
6. Click the **Connection** icon, and then click the **JDBC** tab.
7. Adjust the work repository **JDBC** URL to the new host.
8. To test new work repository connection, click **Test Connection**.

3
Setting Up a Topology

This chapter describes how to set up the topology in Oracle Data Integrator.
This chapter includes the following sections:

- Setting Up the Topology
- Working with Big Data
- Managing Agents

> **See Also:**
>
> the Overview of Oracle Data Integrator Topology section in *Developing Integration Projects with Oracle Data Integrator*.

Setting Up the Topology

The following steps are a guideline to create the topology. You can always modify the topology after an initial setting:

1. Create the contexts corresponding to your different environments. See Creating a Context.
2. Create the data servers corresponding to the servers used by Oracle Data Integrator. See Creating a Data Server.
3. For each data server, create the physical schemas corresponding to the schemas containing data to be integrated with Oracle Data Integrator. See Creating a Physical Schema.
4. Create logical schemas and associate them with physical schemas in the contexts. See Creating a Logical Schema.
5. Create the physical agents corresponding to the Standalone, Standalone Colocated, or Java EE agents that are installed in your information systems. See Creating a Physical Agent.
6. Create logical agents and associate them with physical agents in the contexts. See Creating a Logical Agent.

> **Note:**
>
> You can use the New Model and Topology Objects wizard to create a model and associate it with topology objects, if connected to a work repository. For more information, see the Creating a Model and Topology Objects section in *Developing Integration Projects with Oracle Data Integrator*.

Creating a Context

To create a context:

1. In Topology Navigator expand the Contexts navigation tree.
2. Click **New context** in the navigation tree header.
3. Fill in the following fields:
   - **Name**: Name of the context, as it appears in the Oracle Data Integrator graphical interface.
   - **Code**: Code of the context, allowing a context to be referenced and identified among the different repositories.
   - **Password**: Password requested when the user requests switches to this context in a graphical interface. It is recommended to use a password for critical contexts (for example, contexts pointing to Production data).
   - Check **Default** if you want this context to be displayed by default in the different lists in Designer Navigator or Operator Navigator.
4. From the **File** menu, click **Save**.

Creating a Data Server

A Data Server corresponds for example to a Database, JMS server instance, a scripting engine or a file system accessed with Oracle Data Integrator in the integration flows. Under a data server, subdivisions are created in the form of Physical Schemas.

> **Note:**
>
> Frequently used technologies have their data server creation methods detailed in *Connectivity and Knowledge Modules Guide for Oracle Data Integrator*.

Pre-requisites and Guidelines

It is recommended to follow the guidelines below when creating a data server.

Review the Technology Specific Requirements

Some technologies require the installation and the configuration of elements such as:

- Installation of a JDBC Driver. See Configuring the Domain for the Standalone Agent in *Installing and Configuring Oracle Data Integrator* for more information.
- Installation of a Client Connector
- Data source configuration

Refer to the documentation of the technology you are connecting to through the data server and to Databases, Files, and XML in *Connectivity and Knowledge Modules Guide for Oracle Data Integrator*. The connection information may also change

depending on the technology. Refer to the server documentation provided, and contact the server administrator to define the connection methods.

Create an Oracle Data Integrator User

For each database engine used by Oracle Data Integrator, it is recommended to create a user dedicated for ODI on this data server (typically named `ODI_TEMP`).

Grant the user privileges to:

- Create/drop objects and perform data manipulation in his own schema.
- Manipulate data into objects of the other schemas of this data server according to the operations required for the integration processes.

This user should be used as follows:

- Use this user name/password in the data server user/password definition.
- Use this user's schema as your Work Schema for all data schemas on this server.

Creating a Data Server

To create a Data Server:

1. In Topology Navigator expand the **Technologies** node in the Physical Architecture navigation tree.

   > **Tip:**
   >
   > The list of technologies that are displayed in the Physical Architecture navigation tree may be very long. To narrow the list of displayed technologies, you can hide unused technologies by selecting **Hide Unused Technologies** from the Topology Navigator toolbar menu.

2. Select the technology you want to create a data server for.
3. Right-click and select **New Data Server**.
4. Fill in the following fields in the **Definition** tab:
   - **Name**: Name of the Data Server that will appear in Oracle Data Integrator.

     For naming data servers, it is recommended to use the following naming standard: `<TECHNOLOGY_NAME>_<SERVER_NAME>`.
   - **... (Data Server)**: This is the physical name of the data server used by other data servers to identify it. Enter this name if your data servers can be inter-connected in a native way. This parameter is not mandatory for all technologies.

     For example, for Oracle, this name corresponds to the name of the instance, used for accessing this data server from another Oracle data server through DBLinks.
   - **User/Password**: User name and password for connecting to the data server. This parameter is not mandatory for all technologies, as for example for the File technology.

     Depending on the technology, this could be a "Login", a "User", or an "account". For some connections using the JNDI protocol, the user name and its associated password can be optional (if they have been given in the LDAP directory).

5. Define the connection parameters for the data server:

   A technology can be accessed directly through JDBC or the JDBC connection to this data server can be served from a JNDI directory.

   If the technology is accessed through a JNDI directory:

> **Note:**
>
> The JNDI Providers that are supported are:
>
> - None
> - Simple
> - CRAM-MD5
>
> The Authentication methods supported for each of the supported provider are:
>
> - JNDI User
> - JNDI Password
>
> The JNDI Protocol for each of the supported provider are:
>
> - LDAP - ODI MQ
> - SMQP
> - SUNOPSIS MQ

   a. Check the **JNDI Connection** on the Definition tab.

   b. Go to the **JNDI** tab, and fill in the following fields:

Field	Description
JNDI authentication	• **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 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 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`

Field	Description
JNDI Resource	The directory element containing the connection parameters For example: `cn=sampledb`

If the technology is connected through JDBC:

a. Un-check the **JNDI Connection** box.

b. Go to the **JDBC** tab, and fill in the following fields:

Field	Description
JDBC Driver	Name of the JDBC driver used for connecting to the data server
JDBC URL	URL allowing you to connect to the data server.

You can get a list of pre-defined JDBC drivers and URLs by clicking **Display available drivers** or Display URL sample.

6. Fill in the remaining fields in the **Definition** tab.

- **Array Fetch Size:** When reading large volumes of data from a data server, Oracle Data Integrator fetches successive batches of records. This value is the number of rows (records read) requested by Oracle Data Integrator on each communication with the data server.
- **Batch Update Size:** When writing large volumes of data into a data server, Oracle Data Integrator pushes successive batches of records. This value is the number of rows (records written) in a single Oracle Data Integrator INSERT command.

> **Caution:**
>
> The Fetch Array and Batch Update parameters are accessible only with JDBC. However, not all JDBC drivers accept the same values. At times, you are advised to leave them empty.

> **Note:**
>
> The greater the number specified in the Fetch Array and Batch Update values, the fewer are the number of exchanges between the data server and Oracle Data Integrator. However, the load on the Oracle Data Integrator machine is greater, as a greater volume of data is recovered on each exchange. Batch Update management, like that of Fetch Array, falls within optimization. It is recommended that you start from a default value (30), then increase the value by 10 each time, until there is no further improvement in performance.

- **Degree of Parallelism for Target:** Indicates the number of threads allowed for a loading task. Default value is 1. Maximum number of threads allowed is 99.

> **Note:**
>
> The effect of increasing Degree of Parallelism is dependent on your target environment and whether the resources are sufficient to support a large number of target threads/connections. As per the Fetch Array and Batch Update sizes, you should perform some benchmarking to decide what is the best value for your environment. Details of the performance of the individual source and target threads can be viewed in the Execution Details section for the loading task in Operator. The Execute value is the time spent on performing the JDBC operation and the Wait value is the time the Source is waiting on the Targets to load the rows, or the time the Target is waiting on the Source to provide rows. Also, the Degree of Parallelism > 1 should not be used if you are relying on the order of loading rows, for example, if you are using sequences, timestamps, and so on. This is because the source rows are processed and loaded by one out of a number of target threads in an indeterminate manner.

7. From the **File** menu, click **Save** to validate the creation of the data server.

Creating a Data Server (Advanced Settings)

The following actions are optional:

- Adding Connection Properties
- Defining Data Sources
- Setting Up On Connect/Disconnect Commands

Adding Connection Properties

These properties are passed when creating the connection, in order to provide optional configuration parameters. Each property is a (key, value) pair.

- For JDBC: These properties depend on the driver used. Please see the driver documentation for a list of available properties. It is possible in JDBC to specify here the user and password for the connection, instead of specifying there in the **Definition** tab.
- For JNDI: These properties depend on the resource used.

To add a connection property to a data server:

1. On the **Properties** tab click **Add a Property**.
2. Specify a Key identifying this property. This key is case-sensitive.
3. Specify a value for the property.
4. From the **File** menu, click **Save**.

Defining Data Sources

On the Data Sources tab you can define JDBC data sources that will be used by Oracle Data Integrator Java EE agents deployed on application servers to connect to this data server. Note that data sources are not applicable for Standalone agents.

Defining data sources is not mandatory, but allows the Java EE agent to benefit from the data sources and connection pooling features available on the application server. Connection pooling allows reusing connections across several sessions. If a data source is not declared for a given data server in a Java EE agent, this Java EE agent always connects the data server using direct JDBC connection, that is without using any of the application server data sources.

Before defining the data sources in Oracle Data Integrator, please note the following:

- Datasources for WebLogic Server should be created with the **Statement Cache Size** parameter set to *0* in the **Connection Pool** configuration. Statement caching has a minor impact on data integration performances, and may lead to unexpected results such as data truncation with some JDBC drivers. Note that this concerns only data connections to the source and target data servers, not the repository connections.
- If using Connection Pooling with datasources, it is recommended to avoid **ALTER SESSION** statements in procedures and Knowledge Modules. If a connection requires **ALTER SESSION** statements, it is recommended to disable connection pooling in the related datasources.

To define JDBC data sources for a data server:

1. On the **DataSources** tab of the Data Server editor click **Add a DataSource**
2. Select a Physical Agent in the **Agent** field.
3. Enter the data source name in the **JNDI Name** field.

   Note that this name must match the name of the data source in your application server.
4. Check **JNDI Standard** if you want to use the environment naming context (ENC).

   When **JNDI Standard** is checked, Oracle Data Integrator automatically prefixes the data source name with the string `java:comp/env/` to identify it in the application server's JNDI directory.

   Note that the JNDI Standard is not supported by Oracle WebLogic Server and for global data sources.
5. From the **File** menu, click **Save**.

After having defined a data source for a Java EE agent, you must create it in the application server into which the Java EE agent is deployed. There are several ways to create data sources in the application server, including:

- Configure the data sources from the application server console. For more information, refer to your application server documentation.
- Deploying Datasources from Oracle Data Integrator in an application server for an Agent.
- Deploying an Agent in a Java EE Application Server.

Setting Up On Connect/Disconnect Commands

On the On Connect/Disconnect tab you can define SQL commands that will be executed when a connection to a data server defined in the physical architecture is created or closed.

The On Connect command is executed every time an ODI component, including ODI client components, connects to this data server.

The On Disconnect command is executed every time an ODI component, including ODI client components, disconnects from this data server.

These SQL commands are stored in the master repository along with the data server definition.

Before setting up commands On Connect/Disconnect, please note the following:

- The On Connect/Disconnect commands are only supported by data servers with a technology type Database (JDBC).
- The On Connect and Disconnect commands are executed even when using data sources. In this case, the commands are executed when taking and releasing the connection from the connection pool.
- Substitution APIs are supported. Note that the design time tags `<%` are not supported. Only the execution time tags `<?` and `<@` are supported.
- Only global variables in substitution mode (`#GLOBAL.<VAR_NAME>` or `#<VAR_NAME>`) are supported. See the Variable Scope section in *Developing Integration Projects with Oracle Data Integrator* for more information. Note that the Expression Editor only displays variables that are valid for the current data server.
- The variables that are used in On Connect/Disconnect commands are only replaced at runtime, when the session starts. A command using variables will fail when testing the data server connection or performing a View Data operation on this data server. Make sure that these variables are declared in the scenarios.
- Oracle Data Integrator Sequences are not supported in the On Connect and Disconnect commands.

The commands On Connect/Disconnect have the following usage:

- When a session runs, it opens connections to data servers. every time a connection is opened on a data server that has a command On Connect defined, a task is created under a specific step called *Command on Connect*. This task is named after the data server to which the connection is established, the step and task that create the connection to this data server. It contains the code of the On Connect command.
- When the session completes, it closes all connections to the data servers. Every time a connection is closed on a data server that has a command On Disunite defined, a task is created under a specific step called *Command on Disconnect*. This task is named after the data server that is disconnected, the step and task that dropped the connection to this data server. It contains the code of the On Disconnect command.
- When an operation is made in ODI Studio or ODI Console that requires a connection to the data server (such as View Data or Test Connection), the commands On Connect/Disconnect are also executed if the Client Transaction is selected for this command.

> **Note:**
>
> You can specify whether or not to show On Connect and Disconnect steps in Operator Navigator. If the user parameter Hide On Connect and Disconnect Steps is set to `Yes`, On Connect and Disconnect steps are *not* shown.

To set up On Connect/Disconnect commands:

1. On the **On Connect/Disconnect** tab of the Data Server editor, click **Launch the Expression Editor** in the On Connect section or in the On Disconnect section.
2. In the Expression Editor, enter the SQL command.

> **Note:**
>
> The Expression Editor displays only the substitution methods and keywords that are available for the technology of the data server. Note that global variables are only displayed if the connection to the work repository is available.

3. Click **OK**. The SQL command is displayed in the Command field.
4. Optionally, select **Commit**, if you want to commit the connection after executing the command. Note that if **AutoCommit** or **Client Transaction** is selected in the Execute On list, this value will be ignored.
5. Optionally, select **Ignore Errors**, if you want to ignore the exceptions encountered during the command execution. Note that if **Ignore Errors** is not selected, the calling operation will end in error status. A command with **Ignore Error** selected that fails during a session will appear as a task in a Warning state.
6. From the Log Level list, select the logging level (from 1 to 6) of the connect or disconnect command. At execution time, commands can be kept in the session log based on their log level. Default is `3`.
7. From the Execute On list, select the transaction(s) on which you want to execute the command.

> **Note:**
>
> Transactions from 0 to 9 and the **Autocommit** transaction correspond to connection created by sessions (by procedures or knowledge modules). The **Client Transaction** corresponds to the client components (ODI Console and Studio).

   You can select **Select All** or **Unselect All** to select or unselect all transactions.
8. From the **File** menu, click **Save**.

You can now test the connection, see Testing a Data Server Connectionfor more information.

Testing a Data Server Connection

It is recommended to test the data server connection before proceeding in the topology definition.

To test a connection to a data server:

1. In Topology Navigator expand the **Technologies** node in the **Physical Architecture** navigation tree and then expand the technology containing your data server.
2. Double-click the data server you want to test. The Data Server Editor opens.
3. Click **Test Connection**.

The Test Connection dialog is displayed.

4. Select the agent that will carry out the test. **Local (No Agent)** indicates that the local station will attempt to connect.
5. Click **Detail** to obtain the characteristics and capacities of the database and JDBC driver.
6. Click **Test** to launch the test.

A window showing "connection successful!" is displayed if the test has worked; if not, an error window appears. Use the detail button in this error window to obtain more information about the cause of the connection failure.

Creating a Physical Schema

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 mappings.
- A **Work Schema**, into which Oracle Data Integrator can create and manipulate temporary work data structures associated to the sources and targets contained in the Data Schema.

Frequently used technologies have their physical schema creation methods detailed in *Connectivity and Knowledge Modules Guide for Oracle Data Integrator*.

Before creating a Physical Schema, note the following:

- Not all technologies support multiple schemas. In some technologies, you do not specify the work and data schemas since one data server has only one schema.
- Some technologies do not support the creation of temporary structures. The work schema is useless for these technologies.
- The user specified in the data server to which the Physical Schema is attached must have appropriate privileges on the schemas attached to this data server.
- In a Physical Schema for the OWB technology, only OWB workspaces are displayed and can be selected.

To create a Physical Schema:

1. Select the data server, Right-click and select **New Physical Schema**. The Physical Schema Editor appears.
2. If the technology supports multiple schemas:
   a. Select or type the Data Schema for this Data Integrator physical schema in **... (Schema)**. A list of the schemas appears if the technologies supports schema listing.
   b. Select or type the Work Schema for this Data Integrator physical schema in **... (Work Schema)**. A list of the schemas appears if the technologies supports schema listing.
3. 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).
4. Go to the **Context** tab.
5. Click **Add**.

6. Select a Context and an existing Logical Schema for this new Physical Schema.

   If no Logical Schema for this technology exists yet, you can create it from this Editor.

   To create a Logical Schema:

   a. Select an existing Context in the left column.

   b. Type the name of a Logical Schema in the right column.

      This Logical Schema is automatically created and associated to this physical schema in this context when saving this Editor.

7. From the **File** menu, click **Save**.

Creating a Logical Schema

To create a logical schema:

1. In Topology Navigator expand the **Technologies** node in the Logical Architecture navigation tree.
2. Select the technology you want to attach your logical schema to.
3. Right-click and select **New Logical Schema**.
4. Fill in the **schema name**.
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.
6. From the **File** menu, click **Save**.

Creating a Physical Agent

To create a Physical Agent:

1. In Topology Navigator right-click the Agents node in the Physical Architecture navigation tree.
2. Select **New Agent**.
3. Fill in the following fields:
   - **Name**: Name of the agent used in the Java graphical interface.

     > **Note:**
     >
     > Avoid using *Internal* as agent name. Oracle Data Integrator uses the *Internal* agent when running sessions using the internal agent and reserves the *Internal* agent name.

   - **Host**: Network name or IP address of the machine the agent will be launched on.
   - **Port**: Listening port used by the agent. By default, this port is the 20910.
   - **Web Application Context**: Name of the web application corresponding to the Java EE agent deployed on an application server. For Standalone and Standalone Colocated agents, this field should be set to **oraclediagent**.

> **Note:**
>
> The Web Application Context should be unique for each Java EE agent to be deployed on the WebLogic domain.

- **Protocol**: Protocol to use for the agent connection. Possible values are `http` or `https`. Default is `http`.
- **Maximum number of sessions supported** by this agent.
- **Maximum number of threads**: Controls the number of maximum threads an ODI agent can use at any given time. Tune this as per your system resources and CPU capacity.
- **Maximum threads per session**: ODI supports executing sessions with multiple threads. This limits maximum parallelism for a single session execution.
- **Session Blueprint cache Management**:
  - **Maximum cache entries**: For performance, session blueprints are cached. Tune this parameter to control the JVM memory consumption due to the Blueprint cache.
  - **Unused Blueprint Lifetime (sec)**: Idle time interval for flushing a blueprint from the cache.

4. If you want to setup load balancing, go to the Load balancing tab and select a set of linked physical agent to which the current agent can delegate executions. See Setting Up Load Balancingfor more information.
5. If the agent is launched, click **Test**. The successful connection dialog is displayed.
6. Click **Yes**.

Creating a Logical Agent

To create a logical agent:

1. In Topology Navigator right-click the Agents node in the Logical Architecture navigation tree.
2. Select **New Logical Agent.**
3. Fill in the **Agent Name**.
4. For each Context in the left column, select an existing Physical Agent in the right column. This Physical Agent is automatically associated to the logical agent in this context. Repeat this operation for all necessary contexts.
5. From the **File** menu, click **Save**.

Working with Big Data

Oracle Data Integrator lets you integrate Big Data, deploy and execute Oozie workflows, and generate code in languages such as Pig Latin and Spark.

The following steps are a guideline to set up a topology to work with Big Data:

Table 3-1 Working with Big Data

Task	Documentation
Set up the environment to integrate Hadoop data	See the Setting Up the Environment for Integrating Hadoop Data chapter in *Integrating Big Data with Oracle Data Integrator Guide*.
Set up the data servers for Big Data technologies, such as Hive, HDFS, and HBase	See the following sections in *Integrating Big Data with Oracle Data Integrator Guide*: Setting Up File Data Sources Setting Up Hive Data Sources Setting Up HBase Data Sources
Set up an Oozie Engine if you want to execute Oozie workflows from within Oracle Data Integrator	See the Setting Up and Initializing the Oozie Runtime Engine section in *Integrating Big Data with Oracle Data Integrator Guide*.
Set up Hive, Pig, and Spark topology objects if you want to generate Pig Latin and Spark code	See the following sections in *Integrating Big Data with Oracle Data Integrator Guide*: Setting Up Hive Data Server Creating a Hive Physical Schema Setting Up Pig Data Server Creating a Pig Physical Schema Setting Up Spark Data Server Creating a Spark Physical Schema

Managing Agents

This section describes how to work with a Standalone agent, a Standalone Colocated agent, a Java EE agent and how to handle load balancing. For information on Oracle Data Integrator agents, see the Run-Time Agent section in *Understanding Oracle Data Integrator*.

Standalone Agent

Managing the Standalone agent involves the actions discussed in these sections:

- Configuring a Standalone Agent
- Launching a Standalone Agent
- Stopping an Agent

> **Note:**
>
> The agent command line scripts, which are required for performing the tasks described in this section, are only available if you have installed the Oracle Data Integrator Standalone agent. See *Installing and Configuring Oracle Data Integrator* for information about how to install the Standalone agent.

Configuring a Standalone Agent

Configuring a Standalone agent is described in *Installing and Configuring Oracle Data Integrator*. See:

- Installing the PRODUCT Software
- Creating the Oracle Data Integrator Master and Work Repository Schema
- Configuring the Domain for the Standalone Agent

Launching a Standalone Agent

The Standalone agent is able to execute scenarios on predefined schedules or on demand. The instructions for launching the Standalone agent are provided in the Starting a Standalone Agent Using Node Manager section in *Installing and Configuring Oracle Data Integrator*.

Stopping an Agent

The procedure for stopping a Standalone agent is described in the Stopping Your Oracle Data Integrator Agents section in *Installing and Configuring Oracle Data Integrator*.

Standalone Colocated Agent

Managing a Standalone Colocated agent involves the actions discussed in these sections:

- Configuring a Standalone Colocated Agent
- Launching a Standalone Colocated Agent
- Stopping an Agent

> **Note:**
>
> A Standalone Colocated agent is a Standalone agent that is configured in a WebLogic domain and is managed by an Administration Server. The WebLogic domain makes Oracle Fusion Middleware Infrastructure services available for managing the agent. See the Understanding the Standard Installation Topology for the Standalone Colocated Agent section in *Installing and Configuring Oracle Data Integrator* for more information.

Configuring a Standalone Colocated Agent

Configuring a Standalone Colocated agent is described in *Installing and Configuring Oracle Data Integrator*. See:

- Installing Oracle Data Integrator
- Creating the Oracle Data Integrator Master and Work Repository Schema
- Configuring the Domain for the Standalone Colocated Agent

Launching a Standalone Colocated Agent

The instructions for launching the Standalone Colocated agent are provided in the Starting a Standalone Colocated Agent with Node Manager section in *Installing and Configuring Oracle Data Integrator*.

Stopping an Agent

The procedure for stopping a Standalone Colocated agent is described in the Stopping Your Oracle Data Integrator Agents, section in *Installing and Configuring Oracle Data Integrator*.

Java EE Agent

Managing a Java EE agent involves the actions discussed in the sections:

- Deploying an Agent in a Java EE Application Server
- Creating a Server Template for a Java EE Agent
- Deploying Datasources from Oracle Data Integrator in an application server for an Agent

Deploying an Agent in a Java EE Application Server

Configuring a Java EE agent is described in *Installing and Configuring Oracle Data Integrator*. See:

- Installing the PRODUCT Software
- Creating the Oracle Data Integrator Master and Work Repository Schema
- Configuring the Domain for the Java EE Agent

Creating a Server Template for a Java EE Agent

Oracle Data Integrator provides a Server Template Generation wizard to help you create a server template for a run-time agent.

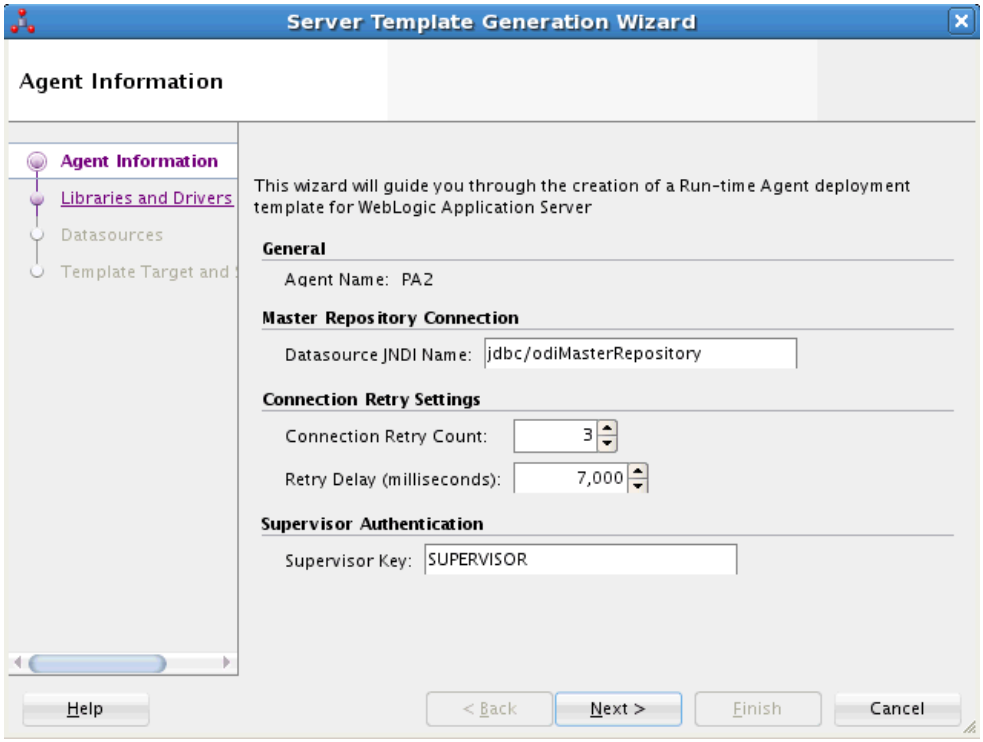
> **Note:**
>
> To use the generate server template feature, ODI Studio must be installed using the Enterprise Installation option. For more information on ODI Studio and installation types, see the Installing the PRODUCT Software in Installing and Configuring Oracle Data Integrator guide.

To open the Server Template Generation wizard:

1. From the Physical Agent Editor toolbar menu, select **Generate Server Template**. This starts the Template Generation wizard, shown in Figure 3-1.

> **Note:**
>
> As mentioned in Creating a Physical Agent, the Web Application Context should be unique for each Java EE agent to be deployed on the WebLogic domain.

Figure 3-1 Server Template Generation Wizard - Agent Information Tab



2. In the **Agent Information** tab, review the agent information and modify the default configuration if needed.

   The Agent Information includes the following parameters:

   - **General**

     Agent Name: Displays the name of the agent that you want to deploy.

   - **Master Repository Connection**

     Datasource JNDI Name: The name of the datasource used by the Java EE agent to connect to the master repository. The template can contain a definition of this datasource. Default is `jdbc/odiMasterRepository`.

   - **Connection Retry Settings**

     Connection Retry Count: Number of retry attempts done if the agent loses the connection to the repository. Note that setting this parameter to a non-zero value, enables a high availability connection retry feature if the ODI repository resides on an Oracle RAC database. If this feature is enabled, the agent can continue to execute sessions without interruptions even if one or more Oracle RAC nodes become unavailable.

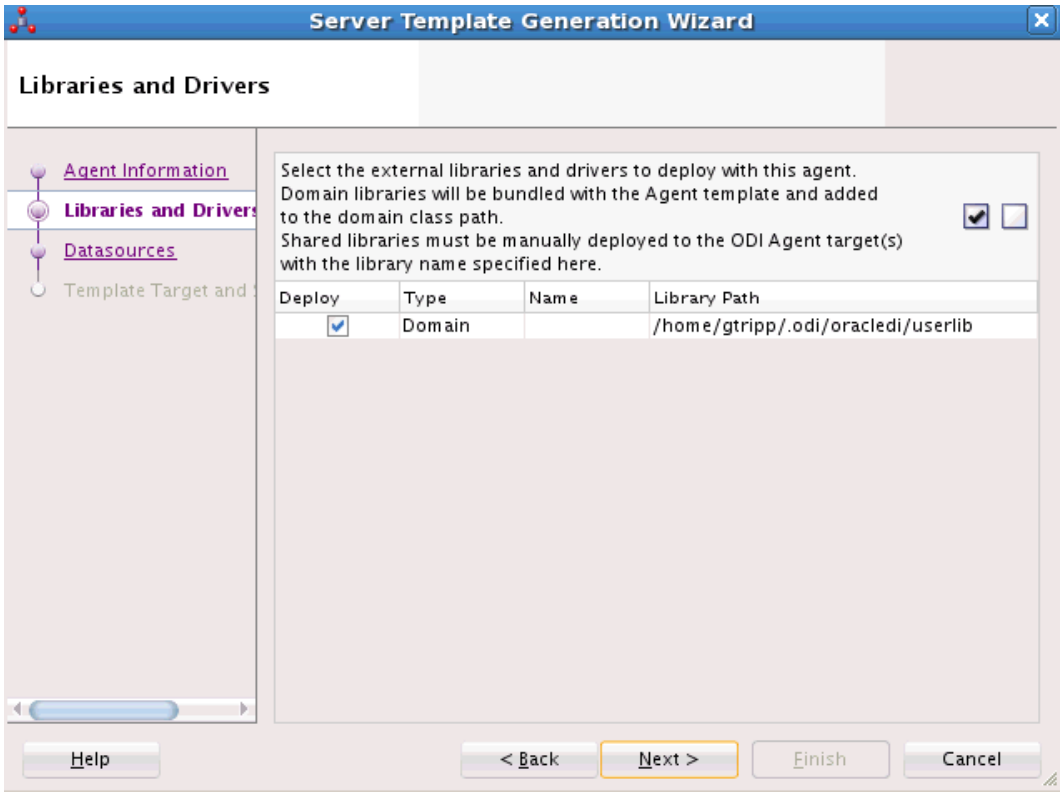
Retry Delay (milliseconds): Interval (in milliseconds) between each connection retry attempt.

- **Supervisor Authentication**

  Supervisor Key: Name of the key in the application server credential store that contains the login and the password of an ODI user with Supervisor privileges. This agent will use this user credentials to connect to the repository.

3. Click **Next**. The Libraries and Drivers tab is displayed, shown in Figure 3-2.

Figure 3-2 Server Template Generation Wizard - Libraries and Drivers Tab



4. In the **Libraries and Drivers** tab, select from the list the external libraries and drivers to deploy with this agent. Only libraries added by the user appear here.

   Note that the libraries can be any JAR or ZIP file that is required for this agent. Additional JDBC drivers or libraries for accessing the source and target data servers must be selected here.

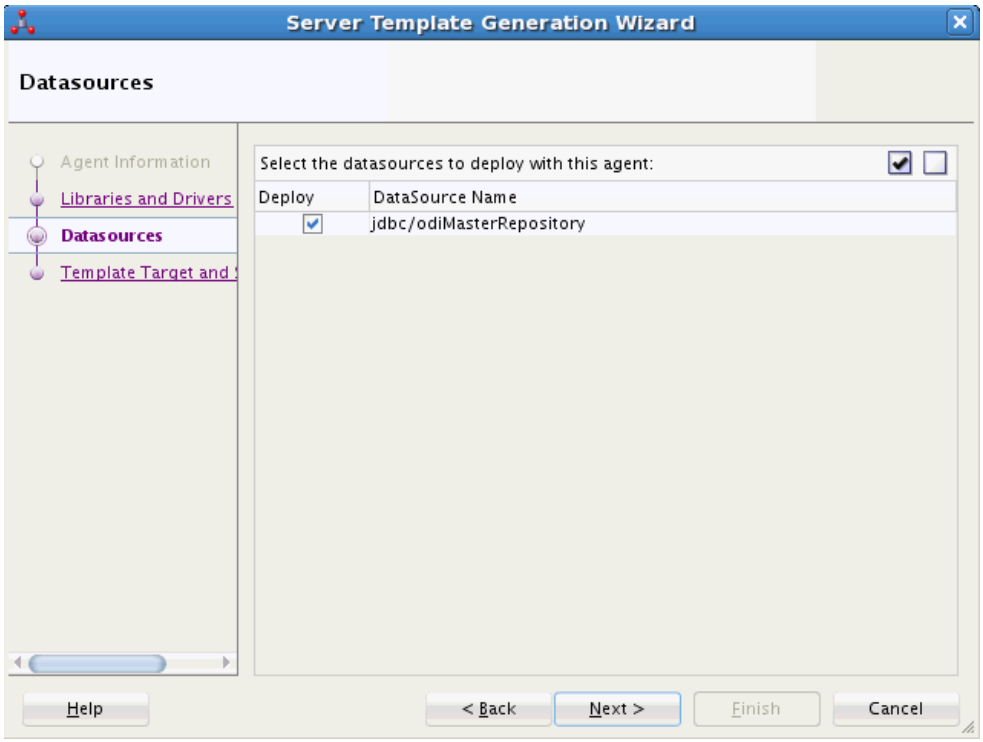
   You can also select a **Domain** library or a **Shared** library in the Type column. On selecting **Domain**, the respective JAR is added as part of the system class path and is visible to all the applications in that domain. On selecting **Shared**, you can select a JAR and make it a shared library, when it is deployed. If the JAR is selected as a shared library, the Name column is enabled and the JAR file's reference name, which is the name provided in the MANIFEST file against Extension-Name attribute, is displayed. If the JAR file does not have a reference name, you must add a unique name in the Name column as selected shared libraries are deployed into the WebLogic server with the name provided in ODI Studio.

> **Note:**
>
> OpenTool JAR should always be selected as Shared type. Also, JARs which are selected as shared, should be deployed as a shared library manually into the Weblogic server. Now these JARs will be added as part of the application class path.

You can use the corresponding buttons in the toolbar to select or deselect all libraries and/or drivers in the list.

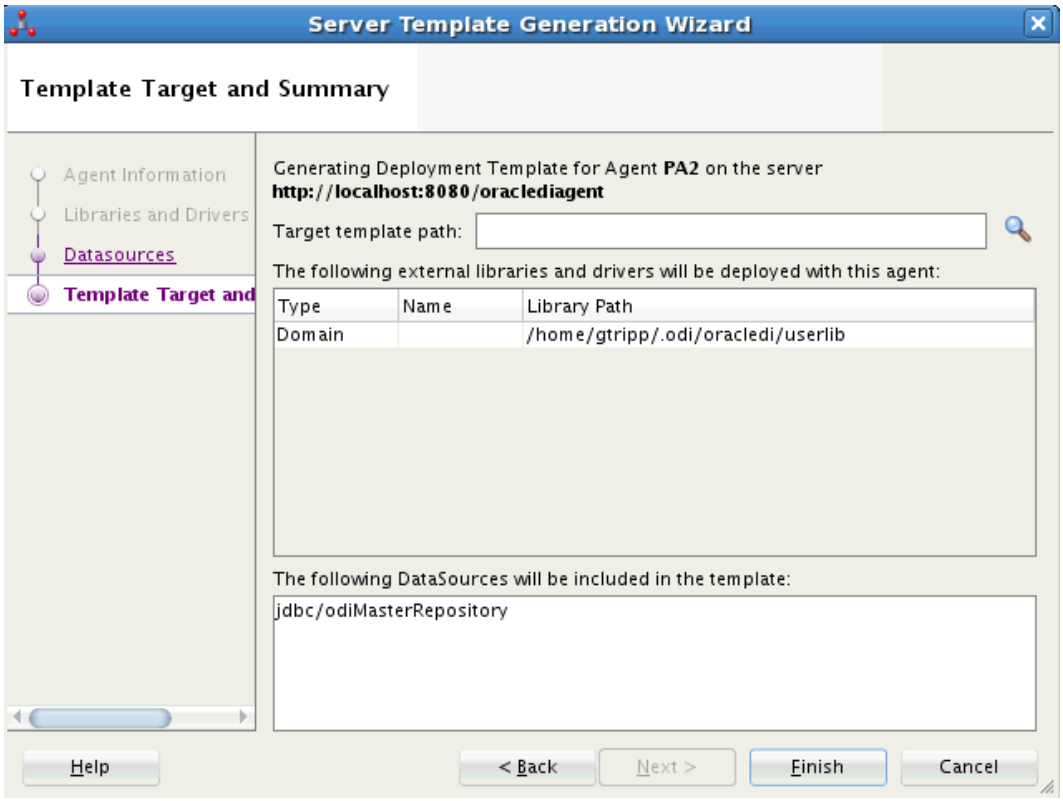
5. Click **Next**. The Datasources tab is displayed, shown in Figure 3-3.

Figure 3-3 Server Template Generation Wizard - Datasources Tab



6. In the **Datasources** tab, select the datasources definitions that you want to include in this agent template. You can only select datasources from the wizard. Naming and adding these datasources is done in the **Data Sources** tab of the **Physical Agent** editor.
7. Click **Next**. The Template Target and Summary tab is displayed, shown in Figure 3-4.

Figure 3-4 Server Template Generation Wizard - Template Target and Summary Tab



8. In **Template Target and Summary** tab, enter the **Target Template Path** where the server template will be generated.
9. Click **Finish** to close the wizard and generate the server template.

   The Template generation information dialog appears.
10. Click **OK** to close the dialog.

The generated template can be used to deploy the agent in WLS or WAS using the respective configuration wizard. Refer to *Installing and Configuring Oracle Data Integrator* for more information.

Example 3-1 Declare the Supervisor in the Credential Store

After deploying the template, it is necessary to declare the Supervisor into the WLS or WAS Credential Store. Refer to *Installing and Configuring Oracle Data Integrator* for more information.

Deploying Datasources from Oracle Data Integrator in an application server for an Agent

You can deploy datasources from the Topology Navigator into an application server for which a Java EE agent is configured. Note that the datasources can only be deployed on the Oracle WebLogic Server.

To deploy datasources in an application server:

1. Open the Physical Agent Editor configured for the application server into which you want to deploy the datasources.
2. Go to the **Datasources** tab.
3. Drag and drop the source/target data servers from the Physical Architecture tree in the Topology Navigator into the **DataSources** tab.
4. Provide a **JNDI Name** for these datasources.
5. Right-click any of the datasource, then select **Deploy Datasource on Server**.
6. On the Datasources Deployment dialog, select the server on which you want to deploy the data sources. Possible values are WLS or WAS server.
7. In the **Deployment Server Details** section, fill in the following fields:
   - **Host**: Host name or IP address of the application server.
   - **Port**: Bootstrap port of the deployment manager
   - **User**: Server user name.
   - **Password**: This user's password
8. In the **Datasource Deployment** section, provide the name of the server on which the datasource should be deployed, for example `odi_server1`.
9. Click **OK**.

> **Note:**
>
> This operation only creates the Datasources definition in the Oracle WebLogic Server. It does not install drivers or library files needed for these datasources to work. Additional drivers added to the Studio classpath can be included into the Agent Template. See Creating a Server Template for a Java EE Agentfor more information.

Example 3-2 WLS Datasource Configuration and Usage

When setting up datasources in WebLogic Server for Oracle Data Integrator, please note the following:

- Datasources should be created with the **Statement Cache Size** parameter set to *0* in the **Connection Pool** configuration. Statement caching has a minor impact on data integration performances, and may lead to unexpected results such as data truncation with some JDBC drivers.
- If using Connection Pooling with datasources, it is recommended to avoid **ALTER SESSION** statements in procedures and Knowledge Modules. If a connection requires **ALTER SESSION** statements, it is recommended to disable connection pooling in the related datasources, as an altered connection returns to the connection pool after usage.

Load Balancing Agents

Oracle Data Integrator allows you to load balance parallel session execution between physical agents.

Each physical agent is defined with:

- A maximum number of sessions it can execute simultaneously from a work repository.

  The maximum number of sessions is a value that must be set depending on the capabilities of the machine running the agent. It can be also set depending on the amount of processing power you want to give to the Oracle Data Integrator agent.
- Optionally, a number of linked physical agents to which it can delegate sessions' executions.

An agent's load is determined at a given time by the ratio `(Number of running sessions / Maximum number of sessions)` for this agent.

Delegating Sessions

When a session is started on an agent with linked agents, Oracle Data Integrator determines which one of the linked agents is less loaded, and the session is delegated to this linked agent.

An agent can be linked to itself, in order to execute some of the incoming sessions, instead of delegating them all to other agents. Note that an agent not linked to itself is only able to delegate sessions to its linked agents, and will never execute a session.

Delegation cascades in the hierarchy of linked agents. If agent A has agent B1 and B2 linked to it, and agent B1 has agent C1 linked to it, then sessions started on agent A will be executed by agent B2 or agent C1. Note that it is not recommended to make loops in agents links.

If the user parameter "Use new Load Balancing" is set to `Yes`, sessions are also re-balanced each time a session finishes. This means that if an agent runs out of sessions, it will possibly be reallocated sessions already allocated to another agent.

Agent Unavailable

When for a given agent the number of running sessions reaches its maximum number of sessions, the agent will put incoming sessions in a "queued" status until the number of running sessions falls below the maximum of sessions.

If an agent is unavailable (because it crashed for example), all its sessions in queue will be re-assigned to another load balanced agent that is neither running any session nor having sessions in queue if the user parameter *Use the new load balancing* is set to Yes.

Setting Up Load Balancing

To setup load balancing:

1. Define a set of physical agents, and link them in a hierarchy of agents (see Creating a Physical Agentfor more information).
2. Start all the physical agents corresponding to the agents defined in the topology.
3. Run the executions on the root agent of your hierarchy. Oracle Data Integrator will balance the load of the executions between its linked agents.

4

High Availability for Oracle Data Integrator

This chapter provides a description of Oracle Data Integrator components from a high availability perspective and a roadmap for setting up a high availability topology. The sections in this chapter outline the single instance concepts that are important for designing high availability deployment.
This chapter includes the following topics:

- Oracle Data Integrator Single Instance Characteristics
- Oracle Data Integrator High Availability and Failover Considerations
- Roadmap for Setting Up a High Availability Topology

> **Note:**
>
> For more information regarding High Availability concepts and procedures, see Introduction to High Availability in *High Availability Guide*.

Oracle Data Integrator Single Instance Characteristics

Oracle Data Integrator run-time agents manage integration processes. Oracle Data Integrator agents are components that run the integration jobs deployed in a production configuration as scenarios stored in a repository.

Oracle Data Integrator agents process each scenario execution instance as a session. Each session exists in the agent as a separate thread of the agent Java process.

Agents store very basic information about the session they run. Most of the session data is stored in the repository. When a scenario is executed on an agent, the agent creates a session in the repository that corresponds to this scenario's instance. The agent reads each task of this session from the repository, processes it, and writes the result - the return code, message and tasks metrics such as the duration or number of rows processed - into the repository.

The repository consists of two database schemas, one containing the master repository, and one containing the work repository. The master repository contains all topology and security related information (such as the source data server definition, target data server definition, and user credential). The work repository contains development and run-time data (such as sessions and scenarios). The master repository also contains the connection information to the work repository. To connect to a work repository, an agent first connects to the master repository, checks the Oracle Data Integrator user's credentials, reads the work repository connection information, and then connects to the work repository. A typical topology includes one master repository and possibly several work repositories (for example, for test and production).

Sessions can be initiated on the agent:

- From another Oracle Data Integrator component (such as the agent or Oracle Data Integrator Studio) over HTTP.
- Via the agent's web service interface.
- From an external scheduler or from a command line.
- From a Java program using the Agent Invocation SDK

The agent is always attached to a master repository. It connects to this master repository at startup and is able to start sessions on any of the work repositories attached to this master. It also acts as a scheduler. On startup, the agent reads from the different work repositories the schedules defined for the agent, and stores this scheduling information. The agent is able to initiate sessions from this in-memory schedule on the appropriate work repositories.

Agents can interact with one another through remote scenario startup (over HTTP) or via the load balancing feature. Load balancing allows defining hierarchies of parent/child agents. In this hierarchy, parent agents can delegate the processing of their sessions to their child agents.

The agent is a Java program that is provided as a Java EE agent and as a standalone agent. The Java EE agent is a web application that can be deployed in a Java EE application server, along with other web applications within the same JVM. This agent can use this server's data sources to connect the source, target and repository databases.

The standalone agent is provided as a standalone Java process started from a command line interface. This standalone agent is similar to the Java EE agent, but is embedded in a lightweight container. The main difference is that unlike the Java EE agent, the standalone agent can connect the source and target data servers using only direct JDBC connection.

Oracle Data Integrator Sessions Lifecycle and Recovery

When an execution request arrives to a run-time agent, the agent connects the master repository to check the user credentials and then the work repository to create the session and all its tasks, and marks them as "waiting." Then it creates the connections to all the data servers that will be used during this session.

When execution starts, the agent reads the first task in the work repository to be executed, and marks both the session and this task as "running." This task can start an operation on the data servers or on the operating system. When the task is complete, the agent writes into the work repository the execution result for this task, moves it to a finished state ("Done", "Warning" or "Error") and proceeds to the next task in the session. Note that errors cases can be handled in the ODI packages, and an error does not necessarily halt a session. When the session completes (either because of an unmanaged error, or by reaching a final step), the agent moves the session to a finished state ("Done", "Warning" or "Error") and releases all the connections. At this point, the session is finished.

Sessions Interruption

Sessions can be interrupted when:

- A user requests the agent to stop the session.

- An agent is stopped by the administrator. All sessions for this agent are stopped, depending on the agent stop mode selected.
- A critical event occurs on the agent or the repository.

Any session that is stopped due to user or administrator action is moved to an error state and marked as "Stopped."

In the case of an agent or repository crash, a session that cannot be stopped properly still appears in a running state in the repository. These sessions are called stale sessions, because they are marked as running, but are no longer handled by any agent. Stale sessions are automatically moved to an error state when an agent restarts and detects that these sessions are incorrectly marked in the repository as being executed by this agent.

Recovering Sessions

Oracle Data Integrator uses JDBC transactions when interacting with source and target data servers, and any open transaction state is not persisted when a session finishes in error state. The appropriate restart point is the task that started the unfinished transaction(s). If such a restart point is not identifiable, it is recommended that you start a fresh session by executing the scenario instead of restarting existing sessions that are in error state.

By default, a session restarts from the last task that failed to execute (typically a task in error or in waiting state). A session may need to be restarted in order to proceed with existing staging tables and avoid re-running long loading phases. In that case the user should take into consideration transaction management, which is KM specific. A general guideline is: If a crash occurs during a loading task, you can restart from the loading task that failed. If a crash occurs during an integration phase, restart from the first integration task, because integration into the target is within a transaction. This guideline applies only to one interface at a time. If several interfaces are chained and only the last one performs the commit, then they should all be restarted because the transaction runs over several interfaces.

To restart from a specific task or step:

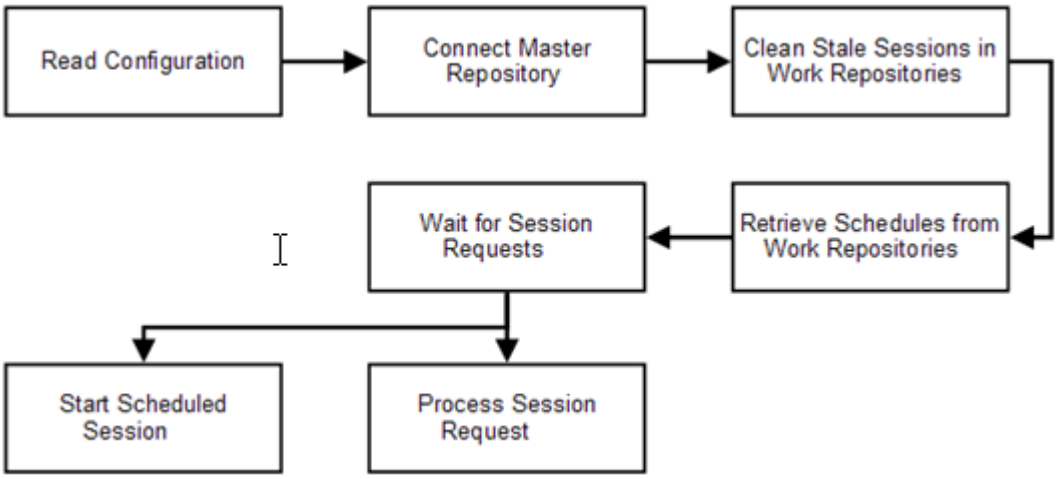
1. In Operator Navigator, navigate to this task or step, edit it and switch it to Waiting state.
2. Set all tasks and steps after this one in the Operator tree view to Waiting state.
3. Right-click the session and click **Restart**.

The session restarts from the first task in waiting state.

Agent Startup and Shutdown Cycle

Figure 4-1 shows the agent startup cycle.

Figure 4-1 Oracle Data Integrator Agent Startup Cycle



When the Oracle Data Integrator agent starts, it first reads its configuration, which includes master repository connection information. Then the agent connects to each of the work repositories attached to this master repository and removes stale sessions. Stale sessions are sessions that are incorrectly indicated in the work repository as running on this given agent. Stale sessions may result from an agent being stopped without being able to stop these sessions gracefully. As the agent restarts, it identifies the stale sessions and moves them to an error state.

From that point, the agent can retrieve and compute the schedules available for it in each work repository. Once this phase is complete, the agent waits for incoming sessions requests to start their processing, and is also able to start sessions based on its schedules.

Oracle Data Integrator External Dependencies

Oracle Data Integrator depends on the Oracle Data Integrator master repository and work repository database schemas.

If advanced features are being used, these other dependencies may exist:

- Other Oracle Data Integrator agents: If the load balancing feature is configured and the agent needs to delegate the execution of sessions to its child agents.
- If External Password Storage is enabled for this agent's master repository, the agent depends on the credential store for retrieving the source and target data servers' passwords to connect these data servers during session execution.
- If External Authentication is enabled for this agent's master repository, the run-time agents as well as Oracle Data Integrator Console depend on the Identity Store service that stores the Oracle Data Integrator user accounts.

These components must be available for the Oracle Data Integrator system to start and run properly.

Oracle Data Integrator Startup and Shutdown Process

For information on the startup and shutdown process, see the following sections in *Installing and Configuring Oracle Data Integrator*.

- Java EE Agent: Configuring the Domain for the Java EE Agent
- Standalone Agent: Configuring the Domain for the Standalone Agent
- Standalone Colocated Agent: Configuring the Domain for the Standalone Colocated Agent

Oracle Data Integrator Configuration Artifacts

This section describes Oracle Data Integrator configuration artifacts.

Agent Configuration

For information on configuring the Java EE Agent, Standalone Agent, Standalone Colocated Agent, see *Installing and Configuring Oracle Data Integrator*.

Oracle Data Integrator Console Configuration

Oracle Data Integrator Console configuration consists of connection definitions to the master and work repositories that can be browsed using this web application.

The list of connections is stored in the repositories.xml file in the following directory:

`user_projects/domains/domainName/config/oracledi`

Connections can be added, edited, or deleted from the Oracle Data Integrator Console management pages.

> **Note:**
>
> Oracle Data Integrator Console is used as the entry point for Enterprise Manager to discover Oracle Data Integrator targets in a domain. The discovery process works in the following way: Enterprise Manager identifies Oracle Data Integrator Console. Using the Oracle Data Integrator Console configuration, Enterprise Manager identifies the master and work repositories as well as the run-time agents in the domain. For more information, see Using Oracle Data Integrator Console .

Oracle Data Integrator Log Locations and Configuration

This section provides information about Oracle Data Integrator log locations and configuration.

Oracle Data Integrator Session Logs

Oracle Data Integrator session execution logs are stored in the work repositories against which the sessions are started. This session shows Oracle Data Integrator session details, such as the executed code and the number of processed rows. This log can be displayed

from the Oracle Data Integrator Studio's Operator Navigator, in the **Session List** accordion, or from Oracle Data Integrator Console's **Browse** tab, under **Run-Time > Sessions**.

Java EE Agent Log Files

The operations performed by the Oracle Data Integrator Java EE agent are logged by Oracle WebLogic Managed Server where the agent application is running. You can find these logs at the following location:

DOMAIN_HOME/servers/*WLS_ServerName*/logs/oracledi/odiagent.log

The log files for the different Oracle WebLogic Server Managed Servers are also available from Oracle WebLogic Server Administration Console. To verify the logs, access Oracle WebLogic Server Administration Console using the following URL: admin_server_host:port/console. Click **Diagnostics-Log Files**.

It is also important to verify the output of the Oracle WebLogic Managed Server where Oracle Data Integrator is running. This information is stored at the following location:

DOMAIN_HOME/servers/*WLS_ServerName*/logs/*WLS_ServerName*.out

Additionally, a diagnostic log is produced in the log directory for the managed server. This log's granularity and logging properties can be changed through the following file:

DOMAIN_HOME/config/fmwconfig/logging/oraclediagent-logging.xml

Standalone and Standalone Colocated Agent Log Files

The operations performed by the Oracle Data Integrator standalone and standalone colocated agent are logged by the lightweight container running the agent. By default, logs are traced on the console and in the <DOMAIN_HOME>/system_components/ODI/<AgentName>/logs/ folder.

The logging method and the logging level can be configured by editing the <DOMAIN_HOME>/config/fmwconfig/components/ODI/<AgentName>/ODI-logging-config.xml file.

Oracle Data Integrator Console Log Files

Oracle Data Integrator Console logging operations are logged by Oracle WebLogic Managed Server where the agent application is running, like the Java EE agent log files described in Java EE Agent Log Files.

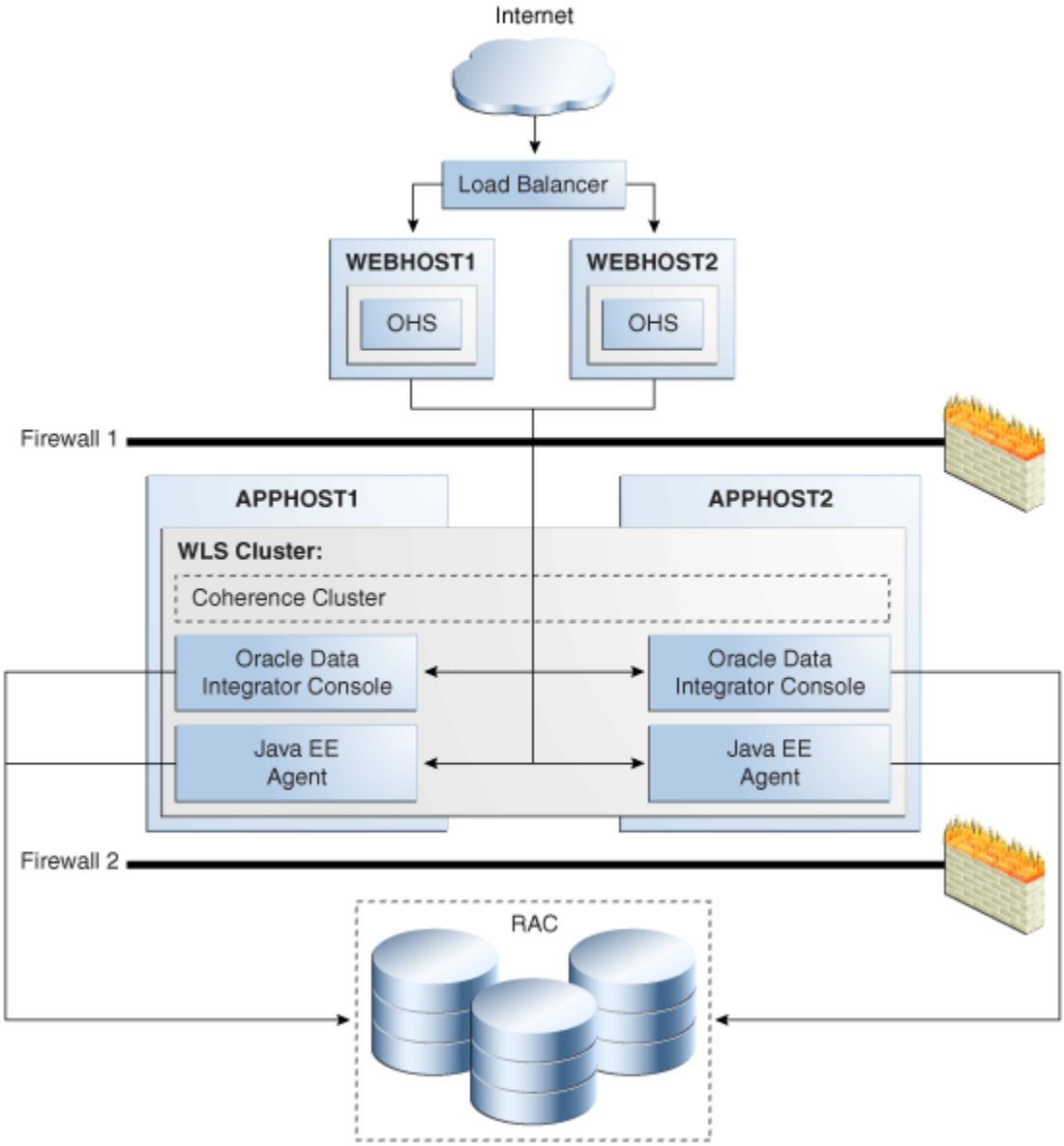
Oracle Data Integrator High Availability and Failover Considerations

This section describes Oracle Data Integrator high availability and failover considerations.

Oracle Data Integrator Clustered Deployment

Figure 4-2 shows a two-node Oracle Data Integrator cluster running on two Oracle WebLogic servers. Oracle WebLogic Servers are front ended by Oracle HTTP Servers, which load balance incoming requests to them.

Figure 4-2 Oracle Data Integrator High Availability Architecture



The main characteristics of this configuration are:

- Oracle Data Integrator applications run on two clustered WebLogic Server managed servers. The WebLogic Server cluster synchronizes configuration for common artifacts of WebLogic Server used by Oracle Data Integrator, such as data sources.
- To avoid duplicate schedule processing, only one of these agents behaves like a scheduler. A Coherence cache is used to handle scheduler service uniqueness and migration.

  The agent provides failover scheduling capabilities. For example, if a schedule is supposed to start at 9 AM, and the cycle is to run job X every hour for four hours, and the agent fails at 9:55 AM, it should compute where it was in the cycle and continue. However, if a single job is scheduled to start at 9 AM, and the agent fails at 8:59 AM, and then recovers at 9:01 AM, then it will not run the job that was scheduled at 9 AM.

- Requests to the Oracle Data Integrator agent in a cluster must be routed via a load balancer or via an HTTP proxy server. The address of this fronting server is used by clients to connect transparently to any of the Oracle Data Integrator servers in the cluster. This address must be specified in the agent definition in the master repository. The scheduler singleton also routes all scheduled sessions startup requests to this address so that they are load balanced over the cluster.
- Oracle Data Integrator's master and work repositories database is configured with Oracle Real Application Clusters (Oracle RAC) to protect from database failures. Oracle Data Integrator components perform the appropriate reconnection and operations retries if database instance failure occurs.

Creating High Availability when Java EE and Colocated agents are installed in Node 1

The `startscen.sh` binary is not part of the Java EE agent. This binary is part of the Standalone or Standalone Colocated agent, so the Standalone or Standalone Colocated agent needs to be installed in order to use it. Installing the Colocated agent in the same domain as the Java EE agent will create the `startscen.sh` binary. Once installed, the `startscen.sh` binary can be used with the Java EE agent.

> **Note:**
>
> Note that using the pack command with the parameter '`managed`' set to **true** does not copy the `startscen.sh` binaries to Node 2. If this parameter is set to **false**, the `startscen.sh` binaries are copied to Node 2.

Oracle Data Integrator Protection from Failure and Expected Behavior

This section describes how an Oracle Data Integrator high availability cluster deployment and Node Manager protects components from failure. This section also describes expected behavior in the event of component failure.

WebLogic Server or Standalone Agent Crash

If a WebLogic Server crashes, Node Manager attempts to restart it locally. If repeated restarts fail, the WebLogic Server infrastructure attempts to perform a server migration of the server to the other node in the cluster. While the failover takes place, the other WebLogic instance becomes the scheduler and is able to read, compute, and execute the schedule for all work repositories. A Coherence cache is used to handle the scheduler lifecycle. Locking guarantees the uniqueness of the scheduler, and event notification provides scheduler migration. Note that when an agent restarts and computes its schedule, it takes into account schedules in progress (those in the middle of an execution cycle). These are automatically continued in their execution cycle beyond the server startup time. New sessions will be triggered as if the scheduler was never stopped.

Stale sessions are moved to an error state and are treated as such when restarted. This session recovery/restart is described in Sessions Interruption and Recovering Sessions.

Oracle Data Integrator agents may be down due to failure in accessing resources, or other issues unrelated to whether the managed server is running. Therefore, Oracle recommends that administrators monitor the managed server logs for cluster errors caused by the application. For information about log file locations, see Oracle Data Integrator Log Locations and Configuration.

The Oracle Data Integrator Console does not support HTTP session failover. The user must log into the Oracle Data Integrator Console again after a failure.

Repository Database Failure

The Oracle Data Integrator repositories are protected against failures in the database by using multi data sources. These multi data sources are typically configured during the initial set up of the system (Oracle Fusion Middleware Configuration Wizard allows you to define these multi-pools directly at installation time) and guarantee that when an Oracle RAC database instance that hosts a repository fails, the connections are re-established with available database instances. The multi data source allows you to configure connections to multiple instances in an Oracle RAC database.

The Java EE agent uses WebLogic multi data sources that are configured during initial setup. The Standalone and Standalone Colocated agents use the Oracle RAC JDBC connection string specified when deploying the ODI Agent templates.

For additional information about multi data source configuration with Oracle RAC, see the Using Multi Data Sources with Oracle RAC appendix in *Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server*.

Oracle Data Integrator implements a retry logic that allows in-flight sessions to proceed if a repository instance becomes unavailable and is restored at a later time. In an Oracle RAC enabled configuration, both in-flight and incoming session execution requests are served as long as an Oracle RAC node is available. This is supported in both the standalone and Java EE agents using the Retry Connection Count *number* and Connection Retry Delay *time* parameters. Users can configure these parameters when generating the WebLogic Server template for the Java EE agent. For the Standalone and Standalone Colocated agents, the retry parameters can be configured in `<DOMAIN_HOME>/config/fmwconfig/components/ODI/<AGENT_NAME>/instance.properties`.

Attribute	Description
PROTOCOL	Listening protocol for the agent. Specify http or https. Default is http.
ODI_SECU_WORK_REP	Work repository name. Default is WORKREP
JMXPORT	JMX agent port number. The agent listens on this port for JMX request to provide its metrics. Default value is the listening port + 1000. For example, if -PORT=20910 then JMXPORT=21910 **Note:** If port conflicts occur, you must change the default JMX port.

Attribute	Description
JMXPROTOCOL	JMX agent protocol. The default is rmi
ODI_MASTER_TIMESTAMP	Master database time stamp. **Note:** If there is a need to reconfigure master repository connection, update this timestamp from the master repository configuration.
ODI_KEYSTORE_ENCODED_PASS	The encrypted keystore password. It is used, only if the protocol is https.
ODI_KEY_ENCODED_PASS	The encrypted key password. This value is optional.
ODI_TRUST_STORE_ENCODED_PASS	The encrypted truststore password. This value is optional.
ODI_CONNECTION_RETRY_COUNT	The number of retries to establish the connection in the event that a repository connection fails. If set to 0, no retry will be performed. Default is 0. **Note:** The RETRY parameters allow the agent to continue sessions, if the repository fails and is temporarily unavailable. This scenario applies primarily to Oracle RAC configurations.
ODI_CONNECTION_RETRY_DELAY	Time in milliseconds between repository connection retries. Default is 7000.
ProductHome	Product home path
DomainHome	Domain home path
ODI_EXCLUDED_CIPHERS	Attribute to disable less secure SSL ciphers. The names of the ciphers to be excluded are to be provided as a comma-separated list.

If Oracle Data Integrator Studio loses its connection to an Oracle RAC database, you will lose any Oracle Data Integrator Studio work performed since the last save operation. As a general practice, save your work on a regular basis when you use Oracle Data Integrator Studio.

Close ODI Studio.

Navigate to `C:\Users\<username>\AppData\Roaming\odi directory`

Delete the “system <ODI version>” folder, with <ODI version> corresponding to your version of ODI.

Start ODI Studio.

Scheduler Node Failure

In an Oracle Data Integrator agent cluster, when the agent node that is the scheduler node crashes, another node in the WebLogic Server cluster takes over as the

scheduler node. The new scheduler node reinitializes all the schedules from that point and continues executing the scheduled scenarios from that point forward.

If a scheduled scenario with a repeatable execution cycle was running on the first scheduler node when that node crashed, the new scheduler node takes over and the scheduler scenario that was running on the first scheduler node will continue its iterations on the new scheduler node from the point at which the first scheduler node crashed.

For example, if the scheduled scenario is configured to repeat the execution ten times after an interval of two minutes, and the first scheduler node crashes in the middle of the third execution, the new scheduler node will continue the execution of the scenario for the next eight executions.

Roadmap for Setting Up a High Availability Topology

This section provides the high level steps you need to perform to set up a high availability topology.

Table 4-1 Roadmap for Setting Up a High Availability Topology

Task	Documentation
1. Install Real Application Clusters	See *Oracle Real Application Clusters Administration and Deployment Guide*.
2. Install middleware components	See *Installing and Configuring the Oracle Fusion Middleware Infrastructure*.
3. Configure Repository Connections to Oracle RAC	See the Configuring ODI Repository Connections for Oracle RAC section in *Administering Oracle Data Integrator*.
4. Install Oracle HTTP Server	See *Installing and Configuring Oracle HTTP Server*.
5. Configure a load balancer	See the Configuring the Load Balancer and Server Load Balancing in a High Availability Environment sections in *High Availability Guide*.
6. Scale out the topology (machine scale out)	See the Scaling Out a Topology (Machine Scale Out) chapter in *High Availability Guide*.
7. Create the high availability domain	See the Configuring the Domain for a Java EE Agent section in *Installing and Configuring Oracle Data Integrator*.
8. Configure high availability for the Administration Server	See the Administration Server High Availability chapter in *High Availability Guide*.
9. Reconfigure agents	See Reconfigure Agents.

Reconfigure Agents

Agent definitions should point to the load balancer address instead of the individual server addresses. Connect to Oracle Data Integrator Studio and edit the Load Balancer Host and Port properties, as shown below:

1. Start Oracle Data Integrator Studio at *ODI_HOME*`/oracledi/client/odi.sh`.
2. After Oracle Data Integrator Studio comes up, click on **Connect to Repository** on the left-hand pane.
3. When the Login window appears, click **OK** to log on.

4. When you are connected, open the Physical Architecture section on the left-hand pane.
5. Select **Agents** and then select **OracleDIAgent**.
6. Edit the following properties:

   - Host: The load balancer virtual server address.

   - Port: The load balancer virtual address listening port.
7. Click **Test** to test the agent connection.
8. Save and exit Oracle Data Integrator Studio.

Part II
Creating and Using Data Services

This part describes how to configure and generate data services with Oracle Data Integrator.

Part II contains the following chapters:

- Setting Up Data Services
- Generating and Deploying Data Services

5

Setting Up Data Services

This chapter describes how to configure data services with Oracle Data Integrator. Data services enable access to your data via a web service interface. Data services also allow access to the changes captured using Oracle Data Integrator's Changed Data Capture feature. See the Using Journalizing chapter in *Developing Integration Projects with Oracle Data Integrator* for more information on Changed Data Capture.
This chapter includes the following section:

- Setting Up Data Services

> **See Also:**
>
> the Using Web Services chapter in *Developing Integration Projects with Oracle Data Integrator*.

Setting Up Data Services

Data services are deployed in a web service container (an application server into which the web service stack is installed). This web service container must be declared in the topology in the form of a data server, attached to the **JAX-WS** technology.

As data services are deployed in an application server, data sources must also be defined in the topology for accessing the data from this application server, and deployed or created in the application server.

Setting up data services involves steps covered in the following sections:

- Configuring the Web Services Container
- Setting up the Data Sources
- Configuring the Model

Configuring the Web Services Container

You must declare the web service container as a data server in the topology, in order to let Oracle Data Integrator deploy Data Services into it.

> **Note:**
>
> Be careful not to mistake the web service containers and the servers containing the data. While both are declared as data servers in Oracle Data Integrator, the former do not contain any data. They are only used to publish Data Services.

Web service containers declared in Oracle Data Integrator have one of three modes of deploying Web Services:

- Copying files directly onto the server, if you have file access to the server.
- Uploading onto the server by FTP.

The next steps in the configuration of the Web Services container depend on the type of web service container and the deployment mode you choose to use.

To configure a web service container:

1. In Topology Navigator expand the **Technologies** node in the Physical Architecture panel.
2. Right-click the **JAX-WS** technology and select **New Data Server**.
3. Fill in the following fields in the **Definition** tab:
   - **Name**: Name of the Data Server that will appear in Oracle Data Integrator.

     For naming data servers, it is recommended to use the following naming standard: <`TECHNOLOGY_NAME>_<SERVER_NAME>`.
   - **Base URL for published services**: Enter the base URL from which the web services will be available. For Oracle WebLogic Server, it is `http://<host>:<HTTP Port>/`.
4. Select one of the following Deployment options:
   - **Save web services into directory**: directory into which the web service will be created. It can be a network directory on the application server or a local directory if you plan to deploy the web services separately into the container.
   - **Upload web services by FTP**: select this option to upload the generated web service to the container. You must provide a **FTP URL** as well as a **User name** and **Password** for performing the upload operation.
5. From the **File** menu, click **Save**. The data server appears in the physical architecture.
6. Select this data server, right-click and select **New Physical Schema**. A new Physical Schema Editor appears. In the **Context** tab, use a context to associate the new physical schema to an existing logical schema. It may be necessary to first create a new logical schema of the **JAX-WS** technology. The process for creating logical schemas is detailed in Setting Up a Topology
7. From the **File** menu, click **Save**.

You only need to configure one physical schema for the web container. Note that the logical schema/context/physical schema association is important here as the context will condition the container into which deployment will take place.

Setting up the Data Sources

The Data Services generated by Oracle Data Integrator do not contain connection information for sources and targets. Instead, they make use of data sources defined within the Web Services container or on the application server. These data sources contain connection properties required to access data, and must correspond to data servers already defined within the Oracle Data Integrator topology.

To set up a data source, you can either:

- Configure the data sources from the application server console. For more information, refer to your application server documentation.
- Deploy the data source from Oracle Data Integrator if the container is an Oracle WebLogic Server. See Setting Up a Topology for more information on data source deployment.

Configuring the Model

To configure Data Services, you must first create and populate a model. See the Creating and Using Data Models and Datastores chapter in *Developing Integration Projects with Oracle Data Integrator* for more information.

You should also have imported the appropriate Service Knowledge Module (SKM) into one of your projects. The SKM contains the code template from which the Data Services will be generated. For more information on importing KMs, see the Creating an Integration Project chapter in *Developing Integration Projects with Oracle Data Integrator*.

To configure a model for data services:

1. In the **Models** tree in the Designer Navigator, select the model.
2. Double-click this model to edit it.
3. Fill in the following fields in the **Services** tab:
   - **Application server:** Select the logical schema corresponding to the container you have previously defined.
   - **Namespace**: Type in the namespace that will be used in the web services WSDL.
   - **Package name**: Name of the generated Java package that contains your Web Service. Generally, this is of the form `com.<company name>.<project name>`.
   - **Name of the data source**, as defined in your container. Depending on the Application server you are using, the data source might be local or global:
     - If your data source is global, you only need to enter the data source name in the Datasource name field.
     - If your data source is local, the data source name should be prefixed by `java:/comp/env/`.

     Note that OC4J uses per default a global data source, Tomcat a local data source. Refer to the documentation of your application server for more information.
   - **Name of data service**: This name is used for the data services operating at the model level. You can also define a data service name for each datastore later.
4. Select a **Service Knowledge Module** (SKM) from the list, and set its options. See *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* for more information about this KM and its options. Only SKMs imported into projects appear in this list.
5. Go to the **Deployed Datastores** tab.
6. Select every datastore that you wish to expose with a data service. For each of those, specify a **Data Service Name** and the name of the **Published Entity**.
7. From the **File** menu, click **Save**.

Although not required, you can also fine-tune the configuration of the generated data services at the datastore and attribute level.

For example, you can specify the operations that will be permitted for each attribute. One important use of this is to lock an attribute against being written to via data services.

To configure data services options at the datastore level:

1. In the **Models** tree in the Designer Navigator, select the datastore.
2. Double-click this datastore to edit it.
3. Select the **Services** tab.
4. Check **Deploy as Data Service** if you want the datastore to be deployed.
5. Enter the **Data Service Name** and the name of the **Published Entity** for the datastore.
6. From the **File** menu, click **Save**.

To configure data service options at the attribute level:

1. In the **Models** tree in the Designer Navigator, select the attribute.
2. Double-click this attribute to edit it.
3. Select the **Services** tab.
4. Check the operation that you want to allow: **SELECT, INSERT, DELETE**. The INSERT action includes the UPDATE action.
5. From the **File** menu, click **Save**.

6
Generating and Deploying Data Services

This chapter describes how to generate and deploy data services with Oracle Data Integrator. This chapter includes the following section:

Generating and Deploying Data Services

Once the model, data sources and container have been configured, it is possible to generate and deploy the data services. Generating data services for a model generates model-level data services as well as the data services for the selected datastores in this model.

To generate Data Services for a model:

1. In the **Models** tree in the Designer Navigator, select the model.
2. Right-click, and select **Generate Service**. The **Generating Data Service** window opens.
3. In the **Generating Data Service** window, fill in the following fields:
   - **Store generated Data Services in:** Oracle Data Integrator places the generated source code and the compiled Web Service here. This directory is a temporary location that can be deleted after generation. You can review the generated source code for the data services here.
   - **Context**: Context into which the data services are generated and deployed. This context choice has three effects:
     - Determining the JDBC/Java datatype bindings at generation time.
     - Determining which physical schemas are used to serve the data.
     - Determining which physical Web Services container is deployed to
   - **Generation Phases**: Choose one or more generation phases. For normal deployment, all three phases should be selected. However, it may be useful to only perform the generation phase when testing new SKMs, for instance. See below for the meaning of these phases.
4. Click **OK** to start data service generation and deployment.

Phase	Description
Generate code	This phase performs the following operation. • Deletes the content of the generation directory. • Generates the Java source code for the data services using the code template from the SKM.
Compilation	This phase performs the following operations: • Extracts web service framework. • Compiles the Java source code.

Phase	Description
Deployment	This phase performs the following operations: • Packages the compiled code. • Deploys the package to the deployment target, using the deployment method selected for the container.

Overview of Generated Services

The data services generated by Oracle Data Integrator include model-level services and datastore level services. These services are described below.

Model-level services

Data services are generated at model-level when the model is enabled for consistent set CDC.

The following services are available at model-level:

- extend Window (no parameters): Carries out an extend window operation.
- lock (Subscriber Name): Locks the consistent set for the named subscriber. To lock the consistent set for several subscribers, call the service several times, using several OdiInvokeWebService steps for example.
- unlock (Subscriber Name): Unlocks the consistent set for the named subscriber.
- purge (no parameters): Purges consumed changes.

See the Using Journalizing chapter in *Developing Integration Projects with Oracle Data Integrator* for more information on these operations.

Datastore-level services

The range of operations offered by each generated data service depends on the SKM used to generate it. There are several common properties shared by the SKMs available with Oracle Data Integrator. In almost every case the name of the published entity forms part of the name of each operation. In the following examples, the published entity "Customer" is used.

The following operations are available at datastore-level:

- Operations on a **single** entity. These operations allow a single record to be manipulated, by specifying a value for its primary key. Other fields may have to be supplied to describe the new row, if any. Examples: addcustomer, getcustomer, deletecustomer, updatecustomer.
- Operations on a group of entities specified by **filter**. These operations involve specifying values for one or several fields to define a filter, then optionally supplying other values for the changes to made to those rows. In general, a maximum number of rows to return can also be specified. Examples: getcustomerfilter, deletecustomerfilter, updatecustomerfilter.
- Operations on a **list** of entities. This list is constructed by supplying several individual entities, as described in the "single entity" case above. Examples: addcustomerlist, deletecustomerlist, getcustomerlist, updatecustomerlist.

Testing Data Services

The easiest way to test generated data services is to use the graphical interface for the OdiInvokeWebService Oracle Data Integrator tool. See the Using Web Services chapter in *Developing Integration Projects with Oracle Data Integrator* for more information on this subject.

Part III

Running and Monitoring Integration Processes

This part describes how to run and monitor integration processes.

Part III contains the following chapters:

7

Running Integration Processes

This chapter describes how to run and schedule integration processes.
This chapter includes the following sections:

> **Note:**
>
> For information on how to run and schedule integration processes within a Hadoop cluster, see *Integrating Big Data with Oracle Data Integrator*.

Understanding ODI Executions

An *execution* takes place when an integration task needs to be performed by Oracle Data Integrator. This integration task may be one of the following:

- An operation on a model, sub-model or a datastore, such as a customized reverse-engineering, a journalizing operation or a static check started from the Oracle Data Integrator Studio
- The execution of a design-time object, such as a mapping, a package or a procedure, typically started from the Oracle Data Integrator Studio
- The execution of a run-time scenario or a Load Plan that was launched from the Oracle Data Integrator Studio, from a command line, via a schedule or a web service interface

Oracle Data Integrator generates the code for an execution in the form of a *session* or in the form of a *Load Plan run* if a Load Plan is executed.

A run-time *Agent* processes this code and connects to the sources and targets to perform the data integration. These sources and targets are located by the agent using a given execution *context*.

When an execution is started from Oracle Data Integrator Studio, the Execution Dialog is displayed. This dialog contains the execution parameters listed in Table 7-1.

Table 7-1 Execution Parameters

Properties	Description
Context	The context into which the session is started.
Agent	The agent which will execute the mapping. The object can also be executed using the agent that is built into Oracle Data Integrator Studio, by selecting **Local (No Agent)**.
Log Level	Level of logging information to retain. All session tasks with a defined log level lower than or equal to this value will be kept in the Session log when the session completes. However, if the object execution ends abnormally, all tasks will be kept, regardless of this setting. Note that log level 6 has the same behavior as log level 5, but with in addition of variable tracking. See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator*.
Simulation	Check Simulation if you want to simulate the execution and create an execution report. Refer to Simulating an Execution for more information.

Session Lifecycle

This section describes the session lifecycle. See the Introduction to Load Plans section in *Developing Integration Projects with Oracle Data Integrator* for more information on Load Plan runs and the Load Plan life cycle.

The lifecycle of a session is as follows:

1. An execution request is sent to the agent, or the agent triggers an execution from a schedule.

   Note that if the execution is triggered from Oracle Data Integrator Studio on a design-time object (mapping, package, etc.), Studio pre-generates in the work repository the code for the session before sending the request. If the execution is started from a scenario, this phase is not necessary as the scenario already contains pre-generated code.
2. The agent completes code generation for the session: It uses the context provided to resolve the physical information such as data server connections and fully qualified tables names. This resulting code is written into the work repository as a session in *Waiting* status.
3. The agent initializes the connections to the source and target data servers that are required for the execution of the session.
4. The agent acknowledges the execution request. If the execution was started from the Studio, the Session Started Dialog is displayed.
5. The agent executes each of the tasks contained in this session, using the capabilities of the database servers, operating systems, or scripting engines to run the code contained in the session's tasks.
6. While processing the session, the agent updates the execution log in the repository, reports execution statistics and error messages.

Once the session is started, you can monitor it in the log, using for example Operator Navigator. Refer to Monitoring Integration Processes for more information on session monitoring.

7. When the session completes, tasks are preserved or removed from the log according to the *log level* value provided when starting for this session.

> **Note:**
>
> A Session is always identified by a unique *Session Number* (or *Session ID*). This number can be viewed when monitoring the session, and is also returned by the command line or web service interfaces when starting a session.

When starting an execution from other locations such as a command line or a web service, you provide similar execution parameters, and receive a similar *Session Started* feedback. If the session is started synchronously from a command line or web service interface, the command line or web service will wait until the session completes, and provide the session return code and an error message, if any.

Executing Mappings, Procedures, Packages and Model Operations

Mappings, procedures, and packages are design-time objects that can be executed from the Designer Navigator of Oracle Data Integrator Studio:

- For more information on mappings execution, refer to the Running Mappings section in *Developing Integration Projects with Oracle Data Integrator*.
- For more information on procedures execution, refer to the Using Procedures section in *Developing Integration Projects with Oracle Data Integrator*.
- For more information on packages execution, refer to the Running a Package section in *Developing Integration Projects with Oracle Data Integrator*.
- For more information on model operations, refer to the Creating and Reverse-Engineering a Model, Checking Data Quality in a Model, and Setting up Journalizing sections in *Developing Integration Projects with Oracle Data Integrator*.

Executing a Scenario

Scenarios can be executed in several ways:

- Executing a Scenario from ODI Studio
- Executing a Scenario from a Command Line.
- From a Web Service. See Executing a Scenario Using a Web Servicefor more information.
- From ODI Console. See Managing Scenarios and Sessions.

> **Note:**
>
> Before running a scenario, you need to have the scenario generated from Designer Navigator or imported from a file. Refer to the Using Scenarios chapter in *Developing Integration Projects with Oracle Data Integrator* for more information.

Executing a Scenario from ODI Studio

You can start a scenario from Oracle Data Integrator Studio from Designer or Operator Navigator.

To start a scenario from Oracle Data Integrator Studio:

1. Select the scenario in the **Projects** navigation tree (in Designer Navigator) or the **Scenarios** navigation tree (in Operator Navigator).
2. Right-click, then select **Run**.
3. In the **Run** dialog, set the execution parameters. Refer to Table 7-1 for more information. To execute the scenario with the agent that is built into Oracle Data Integrator Studio, select **Local (No Agent)**.
4. Click **OK**.
5. If the scenario uses variables as parameters, the Variable values dialog is displayed. Select the values for the session variables. Selecting **Latest value** for a variable uses its current value, or default value if none is available.

When the agent has started to process the session, the **Session Started** dialog appears.

> **Note:**
>
> You can edit a scenario to specify if concurrent executions of the scenario should be limited. You can also specify if concurrent scenarios should end in error immediately, or specify a polling frequency (in seconds) for the wait behavior to check for its turn to run. See the Controlling Concurrent Execution of Scenarios and Load Plans section in *Developing Integration Projects with Oracle Data Integrator* for more information.

Executing a Scenario from a Command Line

You can start a scenario from a command line.

Before executing a scenario from a command line, read carefully the following requirements:

- The command line scripts, which are required for performing the tasks described in this section, are only available if you have installed the Oracle Data Integrator Standalone agent or the Standalone Colocated agent. See *Installing and Configuring Oracle Data Integrator* for information about how to install the Standalone agent or the Standalone Colocated agent.

- To use this command the connection to your repository must be configured in the domain. See *Installing and Configuring Oracle Data Integrator* for more information.
- When starting a scenario from a command line, the session is not started by default against a remote run-time agent, but is executed by a local Java process started from the command line. This process can be aborted locally, but cannot receive a session stop signal as it is not a real run-time agent. As a consequence, sessions started this way cannot be stopped remotely.

  This process will be identified under *Local (No Agent)* in the Oracle Data Integrator Operator logs. You can change this name using the `NAME` parameter.

  If you want to start the session against a run-time agent, you must use the `AGENT_URL` parameter.

To start a scenario from a command line:

1. Create the Oracle Data Integrator domain and configure the agent template:
   - If using the ODI Standalone install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Agent** template to create an ODI Instance configuration.
   - If using the ODI Enterprise install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Colocated Agent** template to create an ODI Instance configuration.
2. Change directory to the `<DOMAIN_HOME>/bin/` directory of the Oracle Data Integrator installation.
3. Enter the following command to start a scenario.

   On UNIX systems:

   ```
   ./startscen.sh -INSTANCE=<ODIInstanceName> <scenario_name>
   <scenario_version> <context_code> [<log_level>] [-
   AGENT_URL=<remote_agent_url>] [-ASYNC=yes|no] [-NAME=<local_agent_name>] [-
   SESSION_NAME=<session_name>] [-KEYWORDS=<keywords>] [<variable>=<value>]*
   ```

   The `ODIInstanceName` is the instance that was configured in step 1 of this procedure.

   On Windows systems:

   ```
   startscen.cmd "-INSTANCE=<ODIInstanceName>" <scenario_name>
   <scenario_version> <context_code> [<log_level>] ["-
   AGENT_URL=<remote_agent_url>"]["-ASYNC=yes|no"] ["-NAME=<local_agent_name>"]
   ["-SESSION_NAME=<session_name>"] ["-KEYWORDS=<keywords>"]
   ["<variable>=<value>"]*
   ```

> **Note:**
>
> On Windows platforms, it is necessary to "delimit" the command arguments containing "=" signs or spaces, by using double quotes. The command call may differ from the Unix command call. For example:
>
> On Unix
>
> ```
> ./startscen.sh -INSTANCE=OracleDIAgent1 PKG001 001 GLOBAL -
> SESSION_NAME=RUN1 -AGENT_URL=http://localhost:20910/oracledіagent
> ```
>
> On Windows
>
> ```
> startscen.cmd "-INSTANCE=OracleDIAgent1" PKG001 001 GLOBAL "-
> SESSION_NAME=RUN1" "-AGENT_URL=http://localhost:20910/
> oracledіagent"
> ```

Table 7-2 lists the different parameters, both mandatory and optional. The parameters are preceded by the "-" character and the possible values are preceded by the "=" character. You must follow the character protection syntax specific to the operating system on which you enter the command.

Table 7-2 Startscen command Parameters

Parameters	Description
-INSTANCE=<ODIInstanceName>	Name of an ODI physical agent which has been defined in the ODI topology and has been configured in the domain from where `startscen.cmd/sh` is executed (mandatory). It indicates to use the configuration files and repository connections from the configured agent, but not the agent itself. It inherits the privileges from the shell (OS command prompt on Windows). If the agent must be used for executing the `startscen` command, set the `AGENT_URL` parameter in addition to `INSTANCE`.
<scenario_name>	Name of the scenario (mandatory).
<scenario_version>	Version of the scenario (mandatory). If the version specified is -1, the latest version of the scenario is executed.
<context_code>	Code of the execution context (mandatory).
[<log_level>]	Level of logging information to retain. This parameter is in the format <n> where <n> is the expected logging level, between 0 and 6. The default log level is 5. Note that log level 6 has the same behavior as log level 5, but with in addition of variable tracking. See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator* for more information. Example: `startscen.cmd SCENAR 1 GLOBAL 5`

Table 7-2 (Cont.) Startscen command Parameters

Parameters	Description
[-AGENT_URL=<remote_agent_url>	URL of the run-time agent that will run this session. If this parameter is set, then the NAME parameter is ignored. The typical Agent URL format is: <protocol>://<AgentHost>:<AgentPort>/<agentWebContext> Example: http://myhost:8001/oracledіagent, https://mySSLHost:8002/oracledіagent
[-ASYNC=yes|no]	Set to yes, for an asynchronous execution on the remote agent. If ASYNC is used, AGENT_URL is mandatory. Note that when the asynchronous execution is used, the session ID of the scenario is returned.
[-NAME=<local_agent_name>]	Agent name that will appear in the execution log for this session, instead of Local (No Agent). This parameter is ignored if AGENT_URL is used. Note that using an existing physical agent name in the NAME parameter is not recommended. The run-time agent whose name is used does not have all the information about this session and will not be able to manage it correctly. The following features will not work correctly for this session: • Clean stale session: This session will be considered as stale by this agent if this agent is started. The session will be pushed to error when the agent will detect this session • Kill Sessions: This agent cannot kill the session when requested. • Agent Session Count: This session is counted in this agent's sessions, even if it is not executed by it. It is recommended to use a NAME that does not match any existing physical agent name. If you want to start a session on a given physical agent, you must use the AGENT_URL parameter instead.
[-SESSION_NAME=<session_name>]	Name of the session that will appear in the execution log. If not specified, the scenario name is used as the session name.
[-KEYWORDS=<keywords>]	List of keywords attached to this session. These keywords make session identification easier. The list is a comma-separated list of keywords.
[<variable>=<value>]	Allows to assign a <value> to a <variable> for the execution of the scenario. <variable> is either a project or global variable. Project variables should be named <Project Code>.<Variable Name>. Global variables should be called GLOBAL.<variable Name>. This parameter can be repeated to assign several variables. Do not use a hash sign (#) to prefix the variable name on the startscen command line.

Restarting a Session

Any session that has encountered an error, or has been stopped by the user can be restarted.

Oracle Data Integrator uses JDBC transactions when interacting with source and target data servers, and any open transaction state is not persisted when a session finishes in error state. The appropriate restart point is the task that started the unfinished transaction(s). If such a restart point is not identifiable, it is recommended that you start a fresh session by executing the scenario instead of restarting existing sessions that are in error state.

Only sessions in status **Error** or **Waiting** can be restarted. By default, a session restarts from the last task that failed to execute (typically a task in error or in waiting state). A session may need to be restarted in order to proceed with existing staging tables and avoid re-running long loading phases. In that case the user should take into consideration transaction management, which is KM specific. A general guideline is: If a crash occurs during a loading task, you can restart from the loading task that failed. If a crash occurs during an integration phase, restart from the first integration task, because integration into the target is within a transaction. This guideline applies only to one mapping at a time. If several mappings are chained and only the last one performs the commit, then they should all be restarted because the transaction runs over several mappings.

To restart from a specific task or step:

1. In Operator Navigator, navigate to this task or step, edit it and switch it to Waiting state.
2. Set all tasks and steps after this one in the Operator tree view to Waiting state.
3. Restart the session using one of the following methods:
   - Restarting a Session from ODI Studio
   - Restarting a Session from a Command Line.
   - From a Web Service. See Restarting a Session Using a Web Servicefor more information.
   - From ODI Console. See Managing Scenarios and Sessions.

> **WARNING:**
>
> When restarting a session, all connections and transactions to the source and target systems are re-created, and not recovered from the previous session run. As a consequence, uncommitted operations on transactions from the previous run are not applied, and data required for successfully continuing the session may not be present.

Restarting a Session from ODI Studio

To restart a session from Oracle Data Integrator Studio:

1. In Operator Navigator, select the session that you want to restart.

2. Right-click and select **Restart**.
3. In the **Restart Session** dialog, specify the agent you want to use for running the new session.

   To select the agent to execute the session, do one of the following:

   - Select **Use the previous agent: <agent name>** to use the agent that was used for the previous session execution.
   - Select **Choose another agent** to select from the list the agent that you want to use for the session execution.

   > **Note:**
   >
   > Select **Internal** to use the ODI Studio built-in agent.

4. Select the Log Level. Note that log level 6 has the same behavior as log level 5, but with in addition of variable tracking. See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator* for more information.
5. Click **OK** to restart the indicated session and to close the dialog. Click **Cancel** if you do not want to restart session.

When Oracle Data Integrator has restarted the session, the **Session Started** dialog appears.

Restarting a Session from a Command Line

Before restarting a session from a command line, read carefully the following requirements:

- The command line scripts, which are required for performing the tasks described in this section, are only available if you have installed the Oracle Data Integrator Standalone agent or the Standalone Colocated agent. See *Installing and Configuring Oracle Data Integrator* for information about how to install the Standalone agent or the Standalone Colocated agent.
- To use this command the connection to your repository must be configured in the domain. See *Installing and Configuring Oracle Data Integrator* for more information.
- When restarting a session from a command line, the session is not started by default against a remote run-time agent, but is executed by a local Java process started from the command line. This process can be aborted locally, but cannot receive a session stop signal as it is not a real run-time agent. As a consequence, sessions started this way cannot be stopped remotely.

  If you want to start the session against a run-time agent, you must use the `AGENT_URL` parameter.

To restart a session from a command line:

1. Create the Oracle Data Integrator domain and configure the agent template:
   - If using the ODI Standalone install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Agent** template to create an ODI Instance configuration.
   - If using the ODI Enterprise install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Colocated Agent** template to create an ODI Instance configuration.

2. Change directory to the `<DOMAIN_HOME>/bin/` directory of the Oracle Data Integrator installation.
3. Enter the following command to restart a scenario.

   On UNIX systems:

   ```
   ./restartsession.sh -INSTANCE=<ODIInstanceName> <session_number> [-log_level][-AGENT_URL=<remote_agent_url>]
   ```

   The `ODIInstanceName` is the instance that was configured in step 1 of this procedure.

   On Windows systems:

   ```
   restartsession.cmd "-INSTANCE=<ODIInstanceName>" <session_number> [-log_level]["-AGENT_URL=<remote_agent_url>"]
   ```

Table 7-3 lists the different parameters of this command, both mandatory and optional. The parameters are preceded by the "-" character and the possible values are preceded by the "=" character. You must follow the character protection syntax specific to the operating system on which you enter the command.

Table 7-3 restartsess command Parameters

Parameters	Description
`-INSTANCE=<ODIInstanceName>`	Name of the ODI Instance configured in the domain. The ODI Instance defines the repository configuration to be used for restarting the session (mandatory).
`<session_number>`	Number (ID) of the session to be restarted.
`[-log_level]`	Level of logging information to retain. Note that log level 6 has the same behavior as log level 5, but with in addition of variable tracking. Note that if this log_level parameter is not provided when restarting a session, the previous log level used for executing the session will be reused. See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator* for more information.
`[-AGENT_URL=<remote_agent_url>]`	URL of the run-time agent that will restart this session. By default the session is executed by a local Java process started from the command line.

Stopping a Session

Any running or waiting session can be stopped. You may want to stop a session when you realize that for example your mapping contains errors or when the execution takes a long time. Certain privileges are required for you to stop sessions. See Sessionsfor more information.

Note that there are two ways to stop a session:

- **Normal**: The session is stopped once the current task is finished.
- **Immediate**: The current task is immediately interrupted and the session is stopped. This mode allows to stop long-running tasks, as for example long SQL statements before they complete.

> **Note:**
>
> The immediate stop works only with technologies and drivers that support task interruption. It is supported if the `statement.cancel` method is implemented in the JDBC driver.

> **Note:**
>
> Only sessions that are running within a Java EE, Standalone, or Standalone Colocated agent can be stopped. Sessions running in the Studio built-in agent or started with the `startscen.sh` or `startscen.cmd` script without the `AGENT_URL` parameter, cannot be stopped. See Executing a Scenariofor more information.

Session can be stopped in several ways:

- Stopping a Session From ODI Studio
- Stopping a Session From a Command Line
- From ODI Console. See Managing Scenarios and Sessions.

Stopping a Session From ODI Studio

To stop a session from Oracle Data Integrator Studio:

1. In Operator Navigator, select the running or waiting session to stop from the tree.
2. Right-click then select **Stop Normal** or **Stop Immediate**.
3. In the Stop Session Dialog, click **OK**.

The session is stopped and changed to *Error* status.

Stopping a Session From a Command Line

Before stopping a session from a command line, read carefully the following requirements:

- The command line scripts, which are required for performing the tasks described in this section, are only available if you have installed the Oracle Data Integrator Standalone agent or the Standalone Colocated agent. See *Installing and Configuring Oracle Data Integrator* for information about how to install the Standalone agent or the Standalone Colocated agent.
- To use this command the connection to your repository must be configured in the domain. See *Installing and Configuring Oracle Data Integrator* for more information.

To stop a session from a command line:

1. Create the Oracle Data Integrator domain and configure the agent template:
   - If using the ODI Standalone install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Agent** template to create an ODI Instance configuration.

- If using the ODI Enterprise install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Colocated Agent** template to create an ODI Instance configuration.

2. Change directory to the `<DOMAIN_HOME>/bin/` directory of the Oracle Data Integrator installation.
3. Enter the following command to stop a scenario.

   On UNIX systems:

   ```
   ./stopsession.sh -INSTANCE=<ODIInstanceName> <session_id> [-
   AGENT_URL=<remote_agent_url>] [-STOP_LEVEL=<normal (default) |
   immediate>]
   ```

   The `ODIInstanceName` is the instance that was configured in step 1 of this procedure.

   On Windows systems:

   ```
   stopsession.cmd "-INSTANCE=<ODIInstanceName>" <session_id> ["-
   AGENT_URL=<remote_agent_url>"] ["-STOP_LEVEL=<normal (default) |
   immediate>"]
   ```

Table 7-4 lists the different parameters of this command, both mandatory and optional. The parameters are preceded by the "-" character and the possible values are preceded by the "=" character. You must follow the character protection syntax specific to the operating system on which you enter the command.

Table 7-4 StopSession command Parameters

Parameters	Description
`-INSTANCE=<ODIInstanceName>`	Name of the ODI Instance configured in the domain. The ODI Instance defines the repository configuration to be used for stopping the session (mandatory).
`<session_id>`	Number (ID) of the session to be stopped.
`[-AGENT_URL=<remote_agent_url>`	URL of the run-time agent that stops this session. By default the session is executed by a local Java process started from the command line.
`[-STOP_LEVEL=<normal (default) | immediate>]`	The level used to stop a running session. If it is omitted, `normal` will be used as the default stop level.

Executing a Load Plan

Load Plans can be executed in several ways:

- Executing a Load Plan from ODI Studio
- Executing a Load Plan from a Command Line
- From a Web Service. See Executing a Load Plan Using a Web Servicefor more information.
- From ODI Console. See Managing Load Plans.

> **Note:**
>
> A Load Plan cannot be executed using the ODI Studio built-in agent called *Local (No Agent)*.

Executing a Load Plan from ODI Studio

In ODI Studio, you can run a Load Plan in Designer Navigator or in Operator Navigator.

To run a Load Plan in Designer Navigator or Operator Navigator:

1. In the Load Plans and Scenarios navigation tree, select the Load Plan you want to execute.
2. Right-click and select **Run**.
3. In the Start Load Plan dialog, select the execution parameters:
   - Select the **Context** into which the Load Plan will be executed.
   - Select the **Logical Agent** that will run the step.
   - Select the **Log Level.** All sessions with a defined log level lower than or equal to this value will be kept in the Session log when the session completes. However, if the object execution ends abnormally, all tasks will be kept, regardless of this setting.

     Note that log level 6 has the same behavior as log level 5, but with in addition of variable tracking. See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator* for more information.

     Select **Use Session Task Log Level** (default) to use the Session Tasks Log Level value defined in the Load Plan.
   - In the Variables table, enter the **Startup values** for the variables used in this Load Plan.
4. Click **OK**.
5. The **Load Plan Started Window** appears.
6. Click **OK**.

A new execution of the Load Plan is started: a Load Plan instance is created and also the first Load Plan run. You can review the Load Plan execution in the Operator Navigator.

> **Note:**
>
> You can create or edit a load plan to specify if concurrent executions of the load plan should be limited. You can also specify if concurrent load plans should end in error immediately or specify a polling frequency (in seconds) for the wait behavior to check for its turn to run. See the Creating a Load Plan section in *Developing Integration Projects with Oracle Data Integrator* for more information.

Executing a Load Plan from a Command Line

You can start a Load Plan from a command line.

Before executing a Load Plan from a command line, read carefully the following requirements:

- The command line scripts, which are required for performing the tasks described in this section, are only available if you have installed the Oracle Data Integrator Standalone agent or the Standalone Colocated agent. See *Installing and Configuring Oracle Data Integrator* for information about how to install the Standalone agent or the Standalone Colocated agent.
- To use this command the connection to your repository must be configured in the domain. See *Installing and Configuring Oracle Data Integrator* for more information.
- A Load Plan Run is started against a run-time agent identified by the `AGENT_URL` parameter.

To start a Load Plan from a command line:

1. Create the Oracle Data Integrator domain and configure the agent template:
   - If using the ODI Standalone install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Agent** template to create an ODI Instance configuration.
   - If using the ODI Enterprise install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Colocated Agent** template to create an ODI Instance configuration.
2. Change directory to the `<DOMAIN_HOME>/bin/` directory of the Oracle Data Integrator installation.
3. Enter the following command to start a Load Plan.

   On UNIX systems:

   ```
   ./startloadplan.sh -INSTANCE=<ODIInstanceName> <load_plan_name>
   <context_code> [log_level] -AGENT_URL=<agent_url> [-
   KEYWORDS=<keywords>] [<variable>=<value>]["-SYNC=(no|yes)"]["-
   POLLINT=<msec>"]*
   ```

   The `ODIInstanceName` is the instance that was configured in step 1 of this procedure.

   On WINDOWS systems:

   ```
   startloadplan.cmd "-INSTANCE=<ODIInstanceName>" <load_plan_name>
   <context_code> [log_level]"-AGENT_URL=<agent_url>" ["-
   KEYWORDS=<keywords>"] ["<variable>=<value>"]["-SYNC=(no|yes)"]["-
   POLLINT=<msec>"]*
   ```

> **Note:**
>
> On Windows platforms, it is necessary to "delimit" the command arguments containing "=" signs or spaces, by using double quotes. The command call may differ from the Unix command call. For example:
>
> On UNIX systems:
>
> ```
> ./startloadplan.sh -INSTANCE=OracleDIAgent1 DWLoadPlan DEV -
> AGENT_URL=http://localhost:20910/oracledIagent
> ```
>
> On WINDOWS systems:
>
> ```
> startloadplan.cmd "-INSTANCE=OracleDIAgent1" DWLoadPlan DEV "-
> AGENT_URL=http://localhost:20910/oracledIagent"
> ```

Table 7-5 lists the different parameters, both mandatory and optional. The parameters are preceded by the "-" character and the possible values are preceded by the "=" character. You must follow the character protection syntax specific to the operating system on which you enter the command.

Table 7-5 Startloadplan Command Parameters

Parameters	Description
-INSTANCE=<ODIInstanceName>	Name of the ODI Instance configured in the domain. The ODI Instance defines the repository configuration to be used for starting the Load Plan (mandatory).
<load_plan_name>	Name of the Load Plan to be started (mandatory).
<context_code>	Code of the context used for starting the Load Plan. Note that if this value is not provided, the Load Plan uses the context of the session that calls it (mandatory).
[log_level]	Level of logging information to retain. All sessions with a defined log level lower than or equal to this value will be kept in the Session log when the session completes. However, if the object execution ends abnormally, all tasks will be kept, regardless of this setting. Note that log level 6 has the same behavior as log level 5, but with in addition of variable tracking. Default is the Load Plan's Session Tasks Log Level that has been used for starting the Load Plan. See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator* for more information.
["-AGENT_URL=<agent_url>"]	URL of the Physical Agent that starts the Load Plan (mandatory).
["-KEYWORDS=<Keywords>"]	Keywords to improve the organization of ODI logs by session folders and automatic classification. Enter a comma separated list of keywords that will be attached to this Load Plan.

Table 7-5 (Cont.) Startloadplan Command Parameters

Parameters	Description
["variable>=<value> "]	Startup values for the Load Plan variables (optional). Note that project variables should be named <project_code>.<variable_name> and global variables should be named GLOBAL.<variable_name>. This list is of the form <variable>=<value>. The format for Date and Number variables is as follows: • **Date**: yyyy-MM-dd'T'HH:mm:ssZ For example: 2009-12-06T15:59:34+0100 • **Number**: Integer value For example: 29833 For example: "A_PROJ.A_REFRESH_VAR=bb" "A_PROJ.A_CROSS_PROJ_VAR=aa" "A_PROJ.A_VAR=cc"
[-SYNC=(no|yes)]	Synchronous invocation of loadplan: **Yes -** Synchronous. Start the loadplan and wait, until the loadplan run has completed in either **Done** or **Error** status. **No -** Asynchronous (default). Start the loadplan and return, without waiting for the loadplan run to complete.
[-POLLINT=<msec>]	This parameter is applicable, only if -SYNC is Yes. The period of time in milliseconds to wait between polling loadplan run status for completion state. Valid value must be > 0. Default value is 1000 (1 second).

Restarting a Load Plan Run

Restarting a Load Plan, starts a new run for the selected Load Plan instance. Note that when a Load Plan restarts the Restart Type parameter for the steps in error defines how the Load Plan and child sessions will be restarted. See the Defining the Restart Behavior section in *Developing Integration Projects with Oracle Data Integrator* and Restarting a Session for more information.

> **Note:**
>
> Restarting a Load Plan instance depends on the status of its most-recent (highest-numbered) run. Restart is only enabled for the most-recent run, if its status is Error.

Load Plans can be restarted in several ways:

- Restarting a Load Plan from ODI Studio
- Restarting a Load Plan from a Command Line

- From a Web Service. See Restarting a Load Plan Instance Using a Web Servicefor more information.
- From ODI Console. See Managing Load Plans.

Restarting a Load Plan from ODI Studio

To restart a Load Plan from ODI Studio:

1. In Operator Navigator, select the Load Plan Run to restart from the Load Plan Executions navigation tree.
2. Right-click then select **Restart**.
3. In the Restart Load Plan Dialog, select the agent that restarts the Load Plan. Optionally, select a different log level.
4. Click **OK**.

The Load Plan is restarted and a new Load Plan run is created.

Restarting a Load Plan from a Command Line

Before restarting a Load Plan from a command line, read carefully the following requirements:

- The command line scripts, which are required for performing the tasks described in this section, are only available if you have installed the Oracle Data Integrator Standalone agent or the Standalone Colocated agent. See *Installing and Configuring Oracle Data Integrator* for information about how to install the Standalone agent or the Standalone Colocated agent.
- To use this command the connection to your repository must be configured in the domain. See *Installing and Configuring Oracle Data Integrator* for more information.
- A Load Plan Run is restarted against a remote run-time agent identified by the `AGENT_URL` parameter.

To restart a Load Plan from a command line:

1. Create the Oracle Data Integrator domain and configure the agent template:
   - If using the ODI Standalone install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Agent** template to create an ODI Instance configuration.
   - If using the ODI Enterprise install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Colocated Agent** template to create an ODI Instance configuration.
2. Change directory to the `<DOMAIN_HOME>/bin/` directory of the Oracle Data Integrator installation.
3. Enter the following command to restart a Load Plan.

   On UNIX systems:

   ```
   ./restartloadplan.sh -INSTANCE=<ODIInstanceName> <load_plan_instance_id>
   [log_level] -AGENT_URL=<agent_url>["-SYNC=(no|yes)"]["-POLLINT=<msec>"]
   ```

   The `ODIInstanceName` is the instance that was configured in step 1 of this procedure.

   On WINDOWS systems:

```
restartloadplan.cmd "-INSTANCE=<ODIInstanceName>"
<load_plan_instance_id> [log_level] "-AGENT_URL=<agent_url>"["-
SYNC=(no|yes)"]["-POLLINT=<msec>"]
```

> **Note:**
>
> On Windows platforms, it is necessary to "delimit" the command arguments containing "=" signs or spaces, by using double quotes. The command call may differ from the Unix command call.

Table 7-6 lists the different parameters, both mandatory and optional. The parameters are preceded by the "-" character and the possible values are preceded by the "=" character. You must follow the character protection syntax specific to the operating system on which you enter the command.

Table 7-6 Restartloadplan Command Parameters

Parameters	Description
-INSTANCE=<ODIInstanceName>	Name of the ODI Instance configured in the domain. The ODI Instance defines the repository configuration to be used for restarting the Load Plan (mandatory).
<load_plan_instance_id>	ID of the stopped or failed Load Plan instance that is to be restarted (mandatory).
[log_level]	Level of logging information to retain. All sessions with a defined log level lower than or equal to this value will be kept in the Session log when the session completes. However, if the object execution ends abnormally, all tasks will be kept, regardless of this setting. Note that log level 6 has the same behavior as log level 5, but with in addition of variable tracking. Default is the log level value used for the Load Plan's previous run. See the Tracking Variables and Sequences section in*Developing Integration Projects with Oracle Data Integrator* for more information.
["-AGENT_URL=<agent_url>"]	URL of the Physical Agent that starts the Load Plan (optional).
[-SYNC=(no|yes)]	Synchronous invocation of loadplan: **Yes -** Synchronous. Start the loadplan and wait, until the loadplan run has completed in either **Done** or **Error** status. **No -** Asynchronous (default). Start the loadplan and return, without waiting for the loadplan run to complete.
[-POLLINT=<msec>]	This parameter is applicable, only if -SYNC is Yes. The period of time in milliseconds to wait between polling loadplan run status for completion state. Valid value must be > 0. Default value is 1000 (1 second).

Stopping a Load Plan Run

Any running or waiting Load Plan Run can be stopped. You may want to stop a Load Plan Run when you realize that for example your Load Plan contains errors or when the execution takes a long time.

Note that there are two ways to stop a Load Plan Run:

- **Stop Normal**: In normal stop mode, the agent in charge of stopping the Load Plan sends a Stop Normal signal to each agent running a session for this Load Plan. Each agent will wait for the completion of the current task of the session and then end the session in error. Exception steps will not be executed by the Load Plan and once all exceptions are finished the load plan is moved to an error state.
- **Stop Immediate**: In immediate stop mode, the agent in charge of stopping the Load Plan sends a Stop immediate signal to each agent running a session for this Load Plan. Each agent will immediately end the session in error and *not* wait for the completion of the current task of the session. Exception steps will not be executed by the Load Plan and once all exceptions are finished the load plan is moved to an error state.

Load Plans can be stopped in several ways:

- Stopping a Load Plan from ODI Studio
- Stopping a Load Plan Run from a Command Line
- From a Web Service. See Stopping a Load Plan Run Using a Web Servicefor more information.
- From ODI Console. See Managing Load Plans.

Stopping a Load Plan from ODI Studio

To stop a Load Plan Run from ODI Studio:

1. In Operator Navigator, select the running or waiting Load Plan Run to stop from the Load Plan Executions navigation tree.
2. Right-click then select **Stop Normal** or **Stop Immediate**.
3. In the Stop Load Plan Dialog, select the agent that stops the Load Plan.
4. Click **OK**.

The Load Plan run is stopped and changed to *Error* status.

Stopping a Load Plan Run from a Command Line

Before stopping a Load Plan from a command line, read carefully the following requirements:

- The command line scripts, which are required for performing the tasks described in this section, are only available if you have installed the Oracle Data Integrator Standalone agent or the Standalone Colocated agent. See *Installing and Configuring Oracle Data Integrator* for information about how to install the Standalone agent or the Standalone Colocated agent.
- To use this command the connection to your repository must be configured in the domain. See *Installing and Configuring Oracle Data Integrator* for more information.

- A Load Plan Run signal is sent by a remote run-time agent identified by the `AGENT_URL` parameter.

To stop a Load Plan run from a command line:

1. Create the Oracle Data Integrator domain and configure the agent template:
   - If using the ODI Standalone install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Agent** template to create an ODI Instance configuration.
   - If using the ODI Enterprise install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Colocated Agent** template to create an ODI Instance configuration.
2. Change directory to the `<DOMAIN_HOME>/bin/` directory of the Oracle Data Integrator installation.
3. Enter the following command to stop a Load Plan.

   On UNIX systems:

   ```
   ./stoploadplan.sh -INSTANCE=<ODIInstanceName> <load_plan_instance_id>
   [<load_plan_run_count>] -AGENT_URL=<agent_url> [-STOP_LEVEL=<normal
   (default) | immediate>]
   ```

   The `ODIInstanceName` is the instance that was configured in step 1 of this procedure.

   On WINDOWS systems:

   ```
   stoploadplan.cmd "-INSTANCE=<ODIInstanceName>" <load_plan_instance_id>
   [<load_plan_run_count>] "-AGENT_URL=<agent_url>" ["-STOP_LEVEL=<normal
   (default) | immediate>"]
   ```

Table 7-7 lists the different parameters, both mandatory and optional. The parameters are preceded by the "-" character and the possible values are preceded by the "=" character. You must follow the character protection syntax specific to the operating system on which you enter the command.

Table 7-7 Stoploadplan Command Parameters

Parameters	Description
-INSTANCE=<ODIInstanceName>	Name of the ODI Instance configured in the domain. The ODI Instance defines the repository configuration to be used for stopping the Load Plan (mandatory).
<load_plan_instance_id>	ID of the running Load Plan run that is to be stopped (mandatory).
[<load_plan_run_count>]	Load Plan run count of the load plan instance. It prevents unintentional stopping of a load plan run that happens to be the latest one. If it is omitted, the last Load Plan run count will be used (optional).
["-AGENT_URL=<agent_url>"]	URL of the Physical Agent that starts the Load Plan (optional).
[-STOP_LEVEL=<normal (default) | immediate>]	Level used to stop the Load Plan run. Default is `normal`.

> **Note:**
>
> On Windows platforms, it is necessary to "delimit" the command arguments containing "=" signs or spaces, by using double quotes. The command call may differ from the Unix command call.

Scheduling Scenarios and Load Plans

You can schedule the executions of your scenarios and Load Plans using the Oracle Data Integrator built-in scheduler or an external scheduler. Both methods are detailed in this section:

- Scheduling a Scenario or a Load Plan with the Built-in Scheduler
- Scheduling a Scenario or a Load Plan with an External Scheduler

Scheduling a Scenario or a Load Plan with the Built-in Scheduler

You can attach schedules to scenarios and also to Load Plans. Such schedules are managed by the scheduler built-in run-time agent.

It is important to understand that a schedule concerns only one scenario or one Load Plan, while a scenario or a Load Plan can have several schedules and can be scheduled in several ways. The different schedules appear under the **Scheduling** node of the scenario or Load Plan. Each schedule allows a start date and a repetition cycle to be specified.

For example:

- **Schedule 1**: Every Thursday at 9 PM, once only.
- **Schedule 2**: Every day from 8 am to 12 noon, repeated every 5 seconds.
- **Schedule 3**: Every day from 2 PM to 6 PM, repeated every 5 seconds, with a maximum cycle duration of 5 hours.

> **Note:**
>
> - Ensure to create schedule plans in the configured timezone of the repository, to prevent from skipping schedules.
> - Configured Timezone is available as a part of Master Repository Information and by default it is set to `PST8PDT`.

If you are creating a schedule plan through ODI Studio, set the timezone to match the configured timezone from master repository information. For this, add the following line to the file `<MW_HOME>/odi/studio/bin/odi.conf`:

```
AddVMOption -Duser.timezone=<repository configured timezone>
```

For example, for configuring ODI Studio to match the configured timezone in the master repository which is `PST8PDT`, set the timezone in the file `odi.conf` using the following command:

```
AddVMOption -Duser.timezone=PST8PDT
```

Scheduling a Scenario or a Load Plan

To schedule a scenario or a Load Plan from Oracle Data Integrator Studio.

1. Right-click the **Scheduling** node under a scenario or a Load Plan in the Designer or Operator Navigator.
2. Select **New Scheduling**. The Scheduling editor is displayed.
3. On the **Definition** tab of the Scheduling editor specify the parameters as follows:

Properties	Description
Context	Context into which the scenario or Load Plan is started.
Agent	Agent executing the scenario or Load Plan.
Log Level	Level of logging information to retain.

The **Status** parameters define the activation of the schedule.

Properties	Description
Active	The scheduling will be active when the agent is restarted or when the scheduling of the physical agent is updated.
Inactive	The schedule is not active and will not run.
Active for the period	Activity range of the schedule. A schedule active for a period of time will only run within this given period.

The **Execution** parameters define the frequency of execution for each execution cycle.

Properties	Description
Execution	Frequency of execution option (annual, monthly,... simple). This option is completed by a set of options that depend on this main option.

4. On the **Execution Cycle** tab, specify the parameters for the repeat mode of the scenario as follows:

Properties	Description
None (Execute once)	The scenario or Load Plan is executed only one time.

Properties	Description
Many times	The scenario or Load Plan is repeated several times. • **Maximum number of repetitions**: The maximum number of times the scenario is repeated during the cycle. • **Maximum Cycle Duration:** As soon as the maximum time is reached, the scenario is no longer restarted, and the cycle stops. • **Interval between repetitions:** The downtime between each scenario execution.
Constraints	Allows limitations to be placed on one cycle iteration, in the event of a problem during execution. • **Number of Attempts on Failure**: Maximum number of consecutive execution attempts for one iteration. • **Stop Execution After**: Maximum execution time for one iteration. If this time is reached, the scenario or Load Plan is automatically stopped.

5. On the **Variables** tab, unselect **Latest Value** for variables for which you want to provide a **Value**. Only variables used in the scenario or Load Plan and flagged as parameters for this scenario or Load Plan appear in this tab.
6. From the **File** menu, click **Save**.

The new schedule appears under the **Scheduling** node of the scenario or Load Plan.

The schedule changes are taken into account by the run-time agent when it starts or when it receives a schedule update request.

Updating an Agent's Schedule

An agent reads schedules when starting on all the repositories attached to the master repository it connects to. It is possible, if a schedule was added for this agent in a given repository, to refresh the agent schedule.

To update an agent's schedule:

1. In Topology Navigator expand the **Agents** node in the **Physical Architecture** navigation tree.
2. Select the Physical Agent you want to update the schedule.
3. Right-click and select **Update Scheduling...**
4. In the **Select Repositories** dialog, select the repositories from which you want to read scheduling information. Check **Select All Work Repositories** to read scheduling information from all these repositories.
5. Click **OK**.

The agent refreshes and re-computes its in-memory schedule from the schedules defined in these repositories.

You can also use the OdiUpdateAgentSchedule tool (see: the OdiUpdateAgentSchedule section in *Oracle Data Integrator Tools Reference*) to update an agent's schedule.

Displaying the Schedule

You can view the scheduled tasks of all your agents or you can view the scheduled tasks of one particular agent.

> **Note:**
>
> The Scheduling Information is retrieved from the agent's in-memory schedule. The agent must be started and its schedule refreshed in order to display accurate schedule information.

Displaying the Schedule for All Agents

To display the schedule for all agents:

1. Select **Connect Navigator >Scheduling...** from the Operator Navigator toolbar menu.

The **View Schedule** dialog appears, displaying the schedule for all agents.

Displaying the Schedule for One Agent

To display the schedule for one agent:

1. In Topology Navigator expand the **Agents** node in the **Physical Architecture** navigation tree.
2. Select the Physical Agent you want to update the schedule.
3. Right-click and select **View Schedule**.

The **Schedule** Editor appears, displaying the schedule for this agent.

> **Note:**
>
> The Scheduling Information is retrieved from the agent's schedule. The agent must be started and its schedule refreshed in order to display accurate schedule information.

Example 7-1 Using the View Schedule Dialog

The schedule is displayed in form of a Gantt diagram. Table 7-8 lists the details of the **Schedule** dialog.

Table 7-8 Scheduling Details

Parameters	Description
Selected Agent	Agent for which the Schedule is displayed. You can display also the schedule of all agents by selecting **All Agents**.
Selected Work Repository	Only the scenarios executed in the selected work repository are displayed in the schedule. Default is **All Work Repositories**.
Scheduling from... to...	Time range for which the scheduling is displayed. Click **Refresh** to refresh this schedule.
Update	Click **Update** to update the schedule for the selected agent(s).

Table 7-8 (Cont.) Scheduling Details

Parameters	Description
Time Range	The time range specified (1 hour, 2 hours, and so forth) allows you to center the diagram on the current time plus this duration. This feature provides a vision of the sessions in progress plus the incoming sessions. You can use the arrows to move the range forward or backward.
Scenarios details	This panel displays the details and execution statistics for each scheduled scenario.

If you select a zone in the diagram (keep the mouse button pressed), you automatically zoom on the select zone.

By right-clicking in the diagram, you open a context menu for zooming, saving the diagram as an image file, printing or editing the display properties.

Scheduling a Scenario or a Load Plan with an External Scheduler

To start a scenario or a Load Plan with an external scheduler, do one of the following:

- Use the *startscen* or *startloadplan* command from the external scheduler
- Use the web service interface for triggering the scenario or Load Plan execution

For more information, see:

- Executing a Scenario from a Command Line
- Executing a Load Plan from a Command Line
- Executing a Scenario Using a Web Service
- Executing a Load Plan Using a Web Service

If a scenario or a Load Plan completes successfully, the return code will be 0. If not, the return code will be different than 0. This code will be available in:

- The return code of the command line call. The error message, if any, is available on the standard error output.
- The SOAP response of the web service call. The web service response includes also the session error message, if any.

Simulating an Execution

In Oracle Data Integrator you have the possibility at design-time to simulate an execution. Simulating an execution generates and displays the code corresponding to the execution without running this code. Execution simulation provides reports suitable for code review.

> **Note:**
>
> No session is created in the log when the execution is started in simulation mode.

To simulate an execution:

1. In the **Project** view of the Designer Navigator, select the object you want to execute.
2. Right-click and select **Run**.
3. In the **Run** dialog, set the execution parameters and select **Simulation**. See Table 7-1 for more information.
4. Click **OK**.

The Simulation report is displayed.

You can click **Save** to save the report as `.xml` or `.html` file.

Managing Executions Using Web Services

This section explains how to use a web service to perform run-time operations. It contains the following sections.

Introduction to Run-Time Web Services

Oracle Data Integrator includes web services for performing run-time operations. These web services are located in:

- The run-time agent, a web service allows starting a scenario or a Load Plan, monitoring a session status or a Load Plan run status, and restarting a session or a Load Plan instance, as well as stopping a Load Plan run. To use operations from this web service, you must first install and configure a Standalone or a Java EE agent.

The following applies to the SOAP request used against the agent and public web services

- The web services operations accept password in a plaintext in the SOAP request. Consequently, it is strongly recommended to use secured protocols (HTTPS) to invoke web services over a non-secured network. You can alternately use external

authentication. See Using the Run-Time Web Services with External Authenticationfor more information.

- Repository connection information is not necessary in the SOAP request as the agent or public web service component is configured to connect to a master repository. Only an ODI user and the name of a work repository are required to run most of the operations.

Executing a Scenario Using a Web Service

The `invokeStartScen` operation of the agent web service starts a scenario in synchronous or asynchronous mode; in a given work repository. The session is executed by the agent providing the web service.

```
<OdiStartScenRequest>
   <Credentials>
      <OdiUser>odi_user</OdiUser>
      <OdiPassword>odi_password</OdiPassword>
      <WorkRepository>work_repository</WorkRepository>
   </Credentials>
   <Request>
      <ScenarioName>scenario_name</ScenarioName>
      <ScenarioVersion>scenario_version</ScenarioVersion>
      <Context>context</Context>
      <LogLevel>log_level</LogLevel>
      <Synchronous>synchronous</Synchronous>
      <SessionName>session_name</SessionName>
      <Keywords>session_name</Keywords>
      <Variables>
      <Name>variable_name</name>
      <Value>variable_value</Value>
      </Variables>
   </Request>
</OdiStartScenRequest>
```

Note:

If you have multiple variables to be added to the above xml, use the following commands:

```
<Variables>
<Name>variable_name1</name>
<Value>variable_value1</Value>
</Variables>
<Variables>
<Name>variable_name2</name>
<Value>variable_value2</Value>
</Variables>
```

The scenario execution returns the session ID in a response that depends on the value of the `synchronous` element in the request.

- In synchronous mode (`Synchronous`=1), the response is returned once the session has completed, and reflects the execution result.

- In asynchronous mode (`Synchronous`=0), the response is returned once the session is started, and only indicates the fact whether the session was correctly started or not.

This operation returns a response in the following format:

```
<?xml version = '1.0' encoding = 'ISO-8859-1'?><ns2:OdiStartScenResponse
xmlns:ns2="xmlns.oracle.com/odi/OdiInvoke/">   <Session>543001</Session></
ns2:OdiStartScenResponse>
```

Monitoring a Session Status Using a Web Service

The `getSessionStatus` operation of the agent web service returns the status of one or more sessions in a given repository, identified by their Session Numbers provided in the `SessionIds` element. It manages both running and completed sessions.

```
<OdiGetSessionsStatusRequest>
   <Credentials>
      <OdiUser>odi_user</OdiUser>
      <OdiPassword>odi_password</OdiPassword>
      <WorkRepository>work_repository</WorkRepository
   </Credentials>
   <SessionIds>session_number</SessionIds>
</OdiGetSessionsStatusRequest>
```

This operation returns a response in the following format:

```
<SessionStatusResponse>
   <SessionId>session_id</SessionId>
   <SessionStatus>status_code</SessionStatus>
   <SessionReturnCode>return_code</SessionReturnCode>
</SessionStatusResponse>
```

The Return Code value is zero for successful sessions and possible status codes are:

- D: Done
- E: Error
- M: Warning
- Q: Queued
- R: Running
- W: Waiting

Restarting a Session Using a Web Service

The `invokeRestartSess` operation of the agent web service restarts a session identified by its session number (provided in the `SessionID` element) in a given work repository. The session is executed by the agent providing the web service.

Only sessions in status **Error** or **Waiting** can be restarted. The session will resume from the last non-completed task (typically, the one in error).

Note that you can change the value of the variables or use the `KeepVariables` boolean element to reuse variables values from the previous session run.

```
<invokeRestartSessRequest>
   <Credentials>
```

```
            <OdiUser>odi_user</OdiUser>
            <OdiPassword>odi_password</OdiPassword>
            <WorkRepository>work_repository</WorkRepository>
         </Credentials>
         <Request>
            <SessionID>session_number</SessionID>
            <Synchronous>synchronous</Synchronous>
            <KeepVariables>0|1</KeepVariables>
            <LogLevel>log_level</LogLevel>
            <Variables>
            <Name>variable_name</name>
            <Value>variable_value</Value>
            </Variables>
         </Request>
      </invokeRestartSessRequest>
```

This operation returns a response similar to `InvokeStartScen`, depending on the `Synchronous` element's value.

Executing a Load Plan Using a Web Service

The `invokeStartLoadPlan` operation of the agent web service starts a Load Plan in a given work repository. The Load Plan is executed by the agent providing the web service. Note the following concerning the parameters of the `invokeStartLoadPlan` operation:

- `OdiPassword`: Use a password in cleartext.
- `Context`: Use the context code.
- `Keywords`: If you use several keywords, enter a comma separated list of keywords.
- `Name`: Use the fully qualified name for variables: `GLOBAL.variable_name` or `PROJECT_CODE.variable_name`

The following shows the format of the OdiStartLoadPlanRequest.

```
<OdiStartLoadPlanRequest>
   <Credentials>
      <OdiUser>odi_user</OdiUser>
      <OdiPassword>odi_password</OdiPassword>
      <WorkRepository>work_repository</WorkRepository>
   </Credentials>
   <StartLoadPlanRequest>
      <LoadPlanName>load_plan_name</LoadPlanName>
      <Context>context</Context>
      <Keywords>keywords</Keywords>
      <LogLevel>log_level</LogLevel>
      <LoadPlanStartupParameters>
         <Name>variable_name</Name>
         <Value>variable_value</Value>
      </LoadPlanStartupParameters>
    </StartLoadPlanRequest>
</OdiStartLoadPlanRequest>
```

The `invokeStartLoadPlan` operation returns the following values in the response:

- Load Plan Run ID
- Load Plan Run Count
- Master Repository ID

- Master Repository timestamp

The following is an example of an `OdiStartLoadPlan` response:

```
<?xml version = '1.0' encoding = 'UTF8'?>
<ns2:OdiStartLoadPlanResponse xmlns:ns2="xmlns.oracle.com/odi/OdiInvoke/">
   <executionInfo>
      <StartedRunInformation>
         <OdiLoadPlanInstanceId>2001</OdiLoadPlanInstanceId>
         <RunCount>1</RunCount>
         <MasterRepositoryId>0</MasterRepositoryId>
         <MasterRepositoryTimestamp>1290196542926</MasterRepositoryTimestamp>
      </StartedRunInformation>
   </executionInfo>
</ns2:OdiStartLoadPlanResponse>
```

Stopping a Load Plan Run Using a Web Service

The `invokeStopLoadPlan` operation of the agent web service stops a running Load Plan run identified by the Instance ID and Run Number in a given work repository. The Load Plan instance is stopped by the agent providing the web service. Note that the `StopLevel` parameter can take the following values:

- `NORMAL`: Waits until the current task finishes and then stops the session.
- `IMMEDIATE`: Stops the session immediately, cancels all open statements and then rolls back the transactions.

See Stopping a Load Plan Runfor more information on how to stop a Load Plan run and Executing a Load Plan Using a Web Servicefor more information on the other parameters used by the `invokeStopLoadPlan` operation.

```
<OdiStopLoadPlanRequest>
   <Credentials>
      <OdiUser>odi_user</OdiUser>
      <OdiPassword>odi_password</OdiPassword>
      <WorkRepository>work_repository</WorkRepository>
   </Credentials>
   <OdiStopLoadPlanRequest>
      <LoadPlanInstanceId>load_plan_instance_id</LoadPlanInstanceId>
      <LoadPlanInstanceRunCount>load_plan_run_count</LoadPlanInstanceRunCount>
      <StopLevel>stop_level</StopLevel>
   </OdiStopLoadPlanRequest>
</OdiStopLoadPlanRequest>
```

The `invokeStopLoadPlan` operation returns the following values in the response:

- Load Plan Run ID
- Load Plan Run Count
- Master Repository ID
- Master Repository timestamp

The following is an example of an `OdiStopLoadPlan` response:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:OdiStopLoadPlanResponse xmlns:ns2="xmlns.oracle.com/odi/OdiInvoke/">
         <executionInfo>
            <StoppedRunInformation>
```

```
                <OdiLoadPlanInstanceId>3001</OdiLoadPlanInstanceId>
                <RunCount>1</RunCount>
                <MasterRepositoryId>0</MasterRepositoryId>
                <MasterRepositoryTimestamp>1290196542926</MasterRepositoryTimestamp>
             </StoppedRunInformation>
          </executionInfo>
       </ns2:OdiStopLoadPlanResponse>
   </S:Body>
</S:Envelope>
```

Restarting a Load Plan Instance Using a Web Service

The `invokeRestartLoadPlan` operation of the agent web service restarts a Load Plan instance identified by the Instance ID in a given work repository. The Load Plan instance is restarted by the agent providing the web service.

```
<OdiRestartLoadPlanRequest>
   <Credentials>
      <OdiUser>odi_user</OdiUser>
      <OdiPassword>odi_password</OdiPassword>
      <WorkRepository>work_repository</WorkRepository>
   </Credentials>
   <RestartLoadPlanRequest>
      <LoadPlanInstanceId>load_plan_instance_id</LoadPlanInstanceId>
      <LogLevel>log_level</LogLevel>
   </RestartLoadPlanRequest>
</OdiRestartLoadPlanRequest>
```

Monitoring a Load Plan Run Status Using a Web Service

The `getLoadPlanStatus` operation of the agent web service returns the status of one or more Load Plans by their Instance ID and Run Number in a given repository. It manages both running and completed Load Plan instances.

```
<OdiGetLoadPlanStatusRequest>
   <Credentials>
       <OdiUser>odi_user</OdiUser>
       <OdiPassword>odi_password</OdiPassword>
      <WorkRepository>work_repository</WorkRepository>
   </Credentials>
   <LoadPlans>
      <LoadPlanInstanceId>load_plan_instance_id</LoadPlanInstanceId>
      <LoadPlanRunNumber>load_plan_run_number</LoadPlanRunNumber>
   </LoadPlans>
</OdiGetLoadPlanStatusRequest>
```

The `getStopLoadPlan`Status operation returns the following values in the response:

- Load Plan Run ID
- Load Plan Run Count
- Load Plan Run return code
- Load Plan message

The following is an example of an `OdiGetLoadPlanStatus` response:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:OdiGetLoadPlanStatusResponse xmlns:ns2="xmlns.oracle.com/odi/OdiInvoke/">
```

```
         <LoadPlanStatusResponse>
            <LoadPlanInstanceId>3001</LoadPlanInstanceId>
            <LoadPlanRunNumber>1</LoadPlanRunNumber>
            <LoadPlanStatus>E</LoadPlanStatus>
            <LoadPlanReturnCode>ODI-1530</LoadPlanReturnCode>
            <LoadPlanMessage>ODI-1530: Load plan instance was stopped by user
request.</LoadPlanMessage>
         </LoadPlanStatusResponse>
      </ns2:OdiGetLoadPlanStatusResponse>
   </S:Body>
</S:Envelope>
```

Accessing the Web Service from a Command Line

Oracle Data Integrator contains two shell scripts for UNIX platforms that use the web service interface for starting and monitoring scenarios from a command line via the run-time agent web service operations:

- `startscenremote.sh` starts a scenario on a remote agent on its web service. This scenario can be started synchronously or asynchronously. When started asynchronously, it is possible to have the script polling regularly for the session status until the session completes or a timeout is reached.
- `getsessionstatusremote.sh` gets the status of a session via the web service interface. This second script is used in the `startscenremote.sh` script.

Before accessing a web service from a command line, read carefully the following important notes:

- The command line scripts, which are required for performing the tasks described in this section, are only available if you have installed the Oracle Data Integrator Standalone agent. See *Installing and Configuring Oracle Data Integrator* for information about how to install the Standalone agent.
- Unlike the `startscen.sh` command line, these scripts rely on the lightweight WGET utility installed with the UNIX or Linux platform to perform the web service calls. It does not use any java code and uses a polling mechanism to reduce the number of running processes on the machine. These scripts are suitable when a large number of scenarios and sessions need to be managed simultaneously from a command line.

Starting a Scenario

To start a scenario from a command line via the web service:

1. Change directory to the `/agent/bin` directory of the Oracle Data Integrator installation.
2. Enter the following command to start a scenario.

   On UNIX systems:

   ```
   ./startscenremote.sh <scenario_name> <scenario_version> <context_code>
   <work_repository> <remote_agent_url> <odi_user> <odi_password> -l
   <log_level> -s <sync_mode> -n <session_name> -k <session_keyword> -a
   <assign_variable> -t <timeout> -i <interval> -h <http_timeout> -v
   ```

Table 7-9 lists the different parameters of this command, both mandatory and optional.

Table 7-9 Startscenremote command Parameters

Parameters	Description
<scenario_name>	Name of the scenario (mandatory).
<scenario_version>	Version of the scenario (mandatory). If the version specified is -1, the latest version of the scenario is executed.
<context_code>	Code of the execution context (mandatory).
<work_repository>	Name of the work repository containing the scenario (mandatory).
<remote_agent_url>	URL of the run-time agent that will run this session (mandatory).
<odi_user>	Name of the Oracle Data Integrator user used to run this session (mandatory). This parameter is case-sensitive and must match the Oracle Data Integrator user as displayed in ODI Studio. Note that an incorrect value entered for this parameter will result in an error.
<odi_password>	This user's password (mandatory).
-l <log_level>	Level of logging information to retain. This parameter is in the format <n> where <n> is the expected logging level, between 0 and 6. The default log level is 5. Note that log level 6 has the same behavior as log level 5, but with in addition of variable tracking. See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator* for more information. Example: `startscen.cmd SCENAR 1 GLOBAL 5`
-s <sync_mode>	Execution mode: • 0: Synchronous • 1:Asynchronous (Do not wait for session completion) • 2: Asynchronous (Wait for session completion). The default execution mode is 1.
-n <session_name>	Name of the session. This parameter is optional and does not have a value if not specified.
-k <session_keyword>	List of keywords attached to this session. These keywords make session identification easier. The list is a comma-separated list of keywords. This parameter is optional and does not have a value if not specified.
-a <assign_variable>	Assign variable. Allows to assign a <value> to a <variable> for the execution of the scenario. <variable> is either a project or global variable. Project variables should be named <Project Code>.<Variable Name>. Global variables should be called GLOBAL.<variable Name>. This parameter is optional and does not have a value if not specified. You can repeat this parameter to assign several variables. Do not use a hash sign (#) to prefix the variable name on the startscen command line. For Example: `-a PROJ1.VAR1=100`
-t <timeout>	Timeout in seconds for waiting for session to complete if sync_mode = 2. The default timeout is 60.

Table 7-9 (Cont.) Startscenremote command Parameters

Parameters	Description
`-i <interval>`	Polling interval for session status if `sync_mode` = 2. The default polling interval is 10.
`-h <http_timeout>`	HTTP timeout for the web services calls. The default HTTP timeout is 60.
`-v`	Verbose mode. By default, the verbose mode is set to 'off'.

Monitoring a Session Status

To monitor the status of a session from a command line via the web service:

1. Change directory to the `/agent/bin` directory of the Oracle Data Integrator installation.
2. Enter the following command to start a scenario.

   On UNIX systems:

   ```
   ./getsessionstatusremote.sh <session_number> <work_repository>
   <remote_agent_url> <odi_user> <odi_password> -w <sync_mode> -t
   <timeout> -i <interval> -h <http_timeout> -v
   ```

Table 7-10 lists the different parameters of this command, both mandatory and optional.

Table 7-10 GetSessionStatusRemote command Parameters

Parameters	Description
`<session_number>`	Number of the session to monitor (mandatory).
`<work_repository>`	Name of the work repository containing the scenario (mandatory).
`<remote_agent_url>`	URL of the run-time agent that will run this session (mandatory).
`<odi_user>`	Name of the Oracle Data Integrator user used to run this session (mandatory). This parameter is case-sensitive and must match the Oracle Data Integrator user as displayed in ODI Studio. Note that an incorrect value entered for this parameter will result in an error.
`<odi_password>`	This user's password (mandatory).
`-w <sync_mode>`	Wait mode: • 0: Do not wait for session completion, report current status. • 1: Wait for session completion then report status. The default wait mode is 0.
`-t <timeout>`	Timeout in seconds for waiting for session to complete if `sync_mode` = 2. The default timeout is 60.
`-i <interval>`	Polling interval for session status if `sync_mode` = 2. The default polling interval is 10.
`-h <http_timeout>`	HTTP timeout for the web services calls. The default HTTP timeout is 60.
`-v`	Verbose mode. By default, the verbose mode is set to 'off'.

Using the Run-Time Web Services with External Authentication

The web services examples in this chapter use an ODI authentication within the SOAP body, using the *OdiUser* and *OdiPassword* elements.

When external authentication is set up for the repository and container based authentication with Oracle Platform Security Services (OPSS) is configured (See Configuring External Authenticationfor more information), the authentication can be passed to the web service using HTTP basic authentication, WS-Security headers, SAML tokens and so forth. OPSS will transparently handle the authentication on the server-side with the identity provider. In such situation, the *OdiUser* and *OdiPassword* elements can be omitted.

The run-time web services will first try to authenticate using OPSS. If no authentication parameters have been provided, OPSS uses anonymous user and the *OdiUser* and *OdiPassword* are checked. Otherwise (this is in case of invalid credentials to OPSS) OPSS throws an authentication exception and the web service is not invoked.

> **Note:**
>
> OPSS authentication is only possible for a Public Web Service or JEE agent deployed in a Oracle WebLogic Server.

Using WS-Addressing

The web services described in this chapter optionally support WS-Addressing. WS-Addressing allows replying on an endpoint when a run-time web service call completes. For this purpose, two endpoints, *ReplyTo* and *FaultTo*, can be optionally specified in the SOAP request header.

These endpoints are used in the following way:

- When the run-time web service call completes successfully, the result of an *Action* is sent to the *ReplyTo* endpoint.
- If an error is encountered with the SOAP request or if Oracle Data Integrator is unable to execute the request, a message is sent to the *FaultTo* address. If the *FaultTo* address has not been specified, the error is sent to the *ReplyTo* address instead.
- If the Oracle Data Integrator agent encounters errors while processing the request and needs to raise an ODI error message, this error message is sent back to the *ReplyTo* address.

Note that callback operations do not operate in callback mode unless a valid *ReplyTo* address is specified.

The following is an example of a request that is sent to retrieve the session status for session 20001:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:odi="xmlns.oracle.com/odi/OdiInvoke/">
<soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
<wsa:Action soapenv:mustUnderstand="1">xmlns.oracle.com/odi/OdiInvoke/
getSessionStatus</wsa:Action>
<wsa:ReplyTo soapenv:mustUnderstand="1">
<wsa:Address>http://host001:8080/examples/servlets/servlet/RequestPrinter</wsa:Address>
```

```
</wsa:ReplyTo>
<wsa:MessageID soapenv:mustUnderstand="1">uuid:71bd2037-fbef-4e1c-
a991-4afcd8cb2b8e</wsa:MessageID>
</soapenv:Header>
   <soapenv:Body>
      <odi:OdiGetSessionsStatusRequest>
         <Credentials>
            <!--You may enter the following 3 items in any order-->
            <OdiUser></OdiUser>
            <OdiPassword></OdiPassword>
            <WorkRepository>WORKREP1</WorkRepository>
         </Credentials>
         <!--Zero or more repetitions:-->
         <SessionIds>20001</SessionIds>
      </odi:OdiGetSessionsStatusRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

The following call will be made to the *ReplyTo* address (`http://host001:8080/examples/servlets/servlet/RequestPrinter`).

Note that this call contains the response to the *Action* specified in the request, and includes the original *MessageID* to correlate request and response.

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Header>
<To xmlns="http://www.w3.org/2005/08/addressing">http:// host001:8080/examples/
servlets/servlet/RequestPrinter</To>
<Action xmlns="http://www.w3.org/2005/08/addressing">xmlns.oracle.com/odi/
OdiInvoke/:requestPortType:getSessionStatusResponse</Action>
<MessageID xmlns="http://www.w3.org/2005/08/
addressing">uuid:eda383f4-3cb5-4dc2-988c-a4f7051763ea</MessageID>
<RelatesTo xmlns="http://www.w3.org/2005/08/addressing">uuid:71bd2037-fbef-4e1c-
a991-4afcd8cb2b8e</RelatesTo>
</S:Header>
<S:Body>
<ns2:OdiGetSessionsStatusResponse xmlns:ns2="xmlns.oracle.com/odi/OdiInvoke/">
<SessionStatusResponse>
               <SessionId>26001</SessionId>
               <SessionStatus>D</SessionStatus>
               <SessionReturnCode>0</SessionReturnCode>
          </SessionStatusResponse>
</ns2:OdiGetSessionsStatusResponse>
    </S:Body>
</S:Envelope>
```

For more information on WS-Adressing, visit these World Wide Web Consortium (W3C) web sites at the following URLs:

- Web Services Addressing: http://www.w3.org/Submission/ws-addressing/
- WS-Addressing SOAP Binding: http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/
- WS-Addressing WSDL Binding: http://www.w3.org/TR/2006/WD-ws-addr-wsdl-20060216/

Using Asynchronous Web Services with Callback

Long-running web service operations can be started asynchronously following the pattern of JRF asynchronous web services or asynchronous BPEL processes. These follow a "request-response port pair" pattern.

In this pattern, the web service client implements a callback operation. When the server completes the operation requested by the client, it sends the result to this callback operation.

Two specific operations in the agent web service support this pattern: *invokeStartScenWithCallback* and *invokeRestartSessWithCallback*.

These operations provide the following features:

- They do not return any response. These are one way operations.
- The client invoking these two operation must implement respectively the *invokeStartSceCallback* and *invokeRestartSessCallback* one way operations. Results from the *invokeStartScenWithCallback* and *invokeRestartSessWithCallback* actions are sent to these operations.
- The invocation should provide in the SOAP header the *ReplyTo* and possibly *FaultTo* addresses. If the methods are invoked without a *ReplyTo* address, the operation will execute synchronously (which corresponds to a *invokeStartScen* or *invokeRestartSess* operation). When a fault is generated in the operation, it will be sent to the *ReplyTo* address or *FaultTo* address.

A scenario or session started synchronously using the *invokeStartScenWithCallback* and *invokeRestartSessWithCallback* will start and will not return any SOAP response, as they are one way operations. When the session completes, the response is sent the callback address.

> **Note:**
>
> Oracle BPEL takes care of automatically implementing these operations and sends out WS-Addressing headers that point to these endpoints.

8

Debugging Integration Processes

This chapter describes how to use the Oracle Data Integrator debugger to solve problems with your integration processes.
This chapter includes the following sections:

About Sessions and Blueprints

Oracle Data Integrator Studio provides a graphical debugger that allows you to manually step through the steps and tasks of a session. You can use the debugger to identify problems with mappings, procedures, packages, or scenarios. Debugging is performed in the blueprint of an active session.

The blueprint of a session can only be viewed in a tab of the session in Oracle Data Integrator Studio. You will see the same blueprint in different sessions if the mapping/scenario has the same version and timestamp.

Once you edit a mapping, new sessions run based on it will not use a previously cached blueprint: instead, a new blueprint is generated. Every time a mapping is executed, a new blueprint is generated afresh. Only in the case of scenarios do you retain the same blueprint for multiple runs as long as the scenario is not modified. Once the scenario is modified, then a new blueprint is generated for the subsequent sessions.

Blueprint Source and Target Code

The blueprint cannot be edited directly. However, you can edit the Source/Target code for the task that the debugger is currently on in the Session Editor. The edited code is used for all sessions run from a mapping. Even if you delete all of the sessions, the cached blueprint survives and will be reused when you next start a session from the unchanged mapping.

Recommendations for Tuning-up Blueprint Cache Management Parameters

The blueprint cache management parameters (Maximum Cache Entries and Unused Lifetime) have to be tuned-up based on your environment specifics. Although blueprint cache parameters are helpful in improving the performance, it should be rarely changed. Tuning-up these parameters should only be considered after you have evaluated all other performance factors such as network latency, database access lags, etc.

You have to consider the following key factors before deciding to tune-up these parameters:

- Number of active Scenarios you have in a steady state running environment
- Average, cumulative size of the scenarios
- Frequency of repeating a scenario for execution
- Available memory in the system and allocated memory to the JVM running ODI Agent

If you have less active scenarios than the default value (i.e. 200), then do not alter the Maximum Cache Entries parameter. Only when you face memory crunch in the VM, then you may want to reduce this number to smaller value to free up memory for other activities. But this normally degrades the performance of a scenario execution, so it is not recommended.

If you run a scenario every hour and all active scenarios fit in the Maximum Cache Entries and if you are not changing generated scenarios in the runtime environment then you can increase the Unused Blueprint Lifetime parameter to more than 3600 seconds. But if the VM memory is in short supply, then increasing the Unused Blueprint Lifetime parameter may eat up memory that can be put to a better use.

So for tuning-up these parameters consider the above guidelines to sort-out performance issues.

Introduction to Debugging in the Session Editor

In the blueprint in the Session Editor, you can debug a running session or restart a completed session, by manually stepping through the steps and tasks of the session. The blueprint includes a Steps Hierarchy table, which identifies steps and tasks by an ID. The debugger tasks that you can perform include:

- Setting breakpoints for steps and tasks. Breakpoints are shown graphically in the blueprint, and listed in the Debug Breakpoints view.
- Viewing and updating/editing source and target code for any task within the session, and running queries to view data.
- Viewing the values of variables as well as uncommitted data, if the connection is not an **Autocommit** connection.
- Viewing all active debug sessions, and disconnecting or connecting to debuggable sessions.
- Viewing all threads for the connected session.

Icons

The debug execution can be controlled by the debug commands in the debug toolbar:

Icon	Command	Description
	Add Breakpoint	Toggles a breakpoint at the currently selected line in the blueprint, currently selected task in the procedure, or currently selected step in a package.

Icon	Command	Description
	Start Debug Session	Starts a debugging session for the editor in focus. In a Session Editor, this command can be used to restart a session.
	Connect To Debug Session	Connects to a debugging session running on an agent. This opens the currently running session and allow to step through the execution. It is not possible to connect to a local agent.
	Disconnect Debug Session	Disconnects the current debugging session. After the debugging session is disconnected, the execution continues and any breakpoint is ignored. It is possible to reconnect to a disconnected session. It is not possible to disconnect from a session running on a local agent.
	Current Cursor	Highlights the current cursor in the blueprint, and opens the Session Editor, if not already open.
	Get Data	Inserts the SQL code of the currently selected task into the SQL command field of the Debug Data window. Both Source Task Data and Target Task Data windows are populated.
	Step into	Steps to the beginning of the first child node of the currently selected node. If the currently selected node does not have a child, this is disabled.
	Run to Task End	Runs to the end of the currently selected task. If the task has children, execute all children until the end of the task is reached. This is disabled if it is not at the beginning of a task.
	Run to Next Task	Runs to the beginning of the next task. If at the last task in a list of tasks, go to the end of the parent task.
	Run to Step End	Runs to the end of the current step. Executes all tasks until the end of the step is reached.

Icon	Command	Description
	Run to Next Step	Runs to the beginning of the next step.
	Pause	Suspends the current execution. The execution can be continued by stepping through the steps/tasks, or by resuming execution.
	Resume	Resumes execution at the current cursor and continues until the session is finished or a breakpoint is reached.

Differences between Debugging Packages, Mappings, and Procedures

There are differences in where you can set breakpoints when debugging packages, mappings, and procedures:

- Breakpoints cannot be set on a mapping.
- In a procedure, breakpoints can be set on tasks
- In a package, you can set breakpoints on a step.

Starting a Debugging Session

You can start a debug session for an object by accessing a **Debug** command in the context menu or toolbar. Starting a debug session launches the Start Debug Session Dialog. See Debug Session Options.

Starting a Session from the Toolbar

To start a debug session from the toolbar in Oracle Data Integrator Studio:

1. In the **Projects** view, select a mapping, package, procedure, or scenario.
2. From the toolbar, select **Start Debug Session**.
3. In the **Debug** window, configure options described in Debug Session Options.
4. Click OK.

   An information message indicates that the session is started.

Starting a Session from the Navigator Tree

To start a debug session from the navigator tree in Oracle Data Integrator Studio:

1. In the navigator tree, select a mapping, package, procedure, or scenario.

2. Right-click and select **Debug**.
3. In the **Debug** window, configure options described in Debug Session Options.
4. Click **OK**.

   An information message indicates that the session has started.

Starting a Session from the Main Menu

To start a debug session from the main menu of Oracle Data Integrator Studio:

1. Select **Run** > **Debug** > **Start Debug Session**.
2. In the **Debug** window, configure options described in Debug Session Options.
3. Click **OK**.

   An information message indicates that the session has started.

Starting a Session for a Scenario

To start a debug session for a scenario:

1. Locate the scenario using one of the following methods:
   - In the Designer Navigator, expand the **Load Plans and Scenarios** tree and locate the scenario.
   - In the Designer Navigator, expand the **Projects** tree, and locate the scenario under the Scenarios node of the source.
   - In the **Scenarios** tab of the Mapping Editor, locate the scenario in the list of scenarios for the object.
2. Right-click the scenario and select **Debug**.
3. In the **Debug** window, configure options described in Debug Session Options.
4. Click **OK**.

   An information message indicates that the session has started.

Connecting to a Running Debugging Session

You can connect the debugger to a debuggable session on an agent registered with the current repository. The session that the debugger connects to has to be a debuggable session and started prior to opening the **Connect to Debug Session** dialog box. It is not possible to connect to sessions running locally without an agent.

To connect to a session:

1. Select **Run** > **Debug** > **Connect to Debug Session**.
2. In the **Connect Debug Session** window, enter the following connection parameters:
   - **Agent**: Select the agent that the debuggable session runs on. The **Any Agent** choice shows all debuggable sessions from all configured agents in the session list.
   - **Session**: Shows all debuggable sessions available for connection from the agent chosen in the agent list.
3. Click **OK**.

Built-in Agents and External Agents

A debugging session is executed on a context and an agent. You can select the name of the agent that is to execute the debugging session. By selecting *Local (No Agent)*, you select to use the built-in agent in Oracle Data Integrator Studio.

Debug Session Lifecycle

A session is shown as running in the Operator navigator regardless of whether it is a normal or debugging session. There is no indication that the session is paused. You can get back to a stopped session run in a local agent through the Debug Sessions window.

For more information on the lifecycle of a session, see Understanding ODI Executions.

Debug Session Options

Before launching, you can configure options for the debugger session.

Suspend Before the First Task

Check *Suspend Before the First Task*, to cause the debugger to suspend the execution before the first task, even if no client is connected to the debuggable session.

If unchecked, the debug session proceeds to completion if it is not connected to a debugging client. If the debug session is connected to a debugging client, it stops at the first breakpoint, if any.

The setting is checked by default.

Associate to Current Client

Check *Associate to Current Client* to set the debugger to start a session in debug mode and open the blueprint to debug the session.

If unchecked, the debugger starts a session in debug mode but not open the blueprint for debugging. The session can later be associated by this or other clients.

The setting is checked by default.

Delay Before Connect

Set *Delay Before Connect* to define the number of seconds before the debugger attempts a connection to the started session.

The default is 5 seconds.

Debug Descendant Sessions

Check *Debug Descendant Sessions* to prompt you to start a new debugging session whenever a Start command is executed. This setting only applies to packages or procedures, and not mappings.

If unchecked, descendant sessions are executed normally and the debugger client cannot connect to them.

The setting is unchecked by default.

Break On Error

Check *Break on Error* to cause the debugger to suspend execution at a task where an error happened. This enables you to review data within the transaction.

If unchecked, session execution engine reacts to any errors as usual. The debugger client does not provide you any notification.

The setting is unchecked by default.

Stepping through a Blueprint in the Session Editor

The blueprint shows all of the steps and task hierarchy of the selected session, and allows you to go through individual commands.

When you start a session, the Session Editor in its Blueprint view opens, if the *Associate to Current Client* option has been selected. Otherwise, in the debug toolbar, select **Current Cursor** to open the Session Editor.

Steps and Tasks

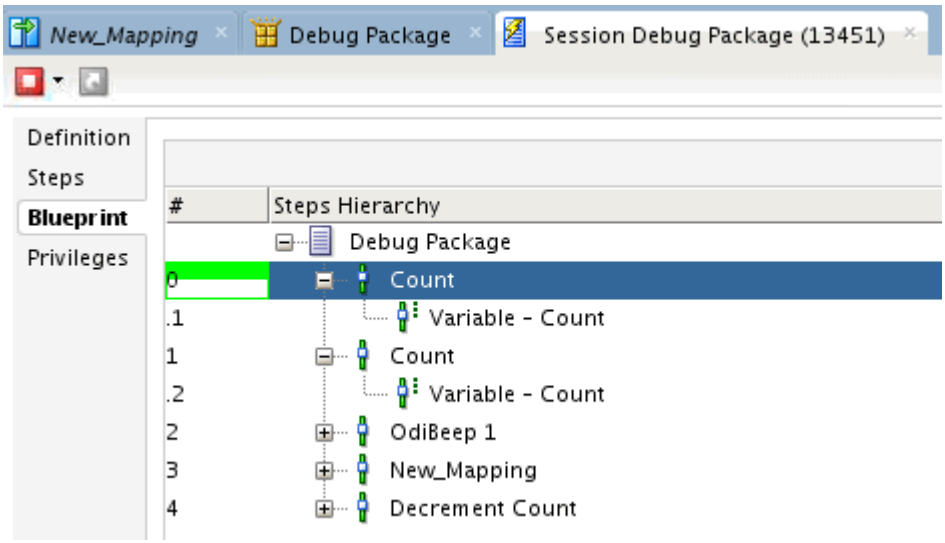
Each row in a blueprint represents a step or task. Tasks in a blueprint can be container tasks, and can execute serially or in parallel. Tasks inside a container task are leaf nodes that get executed. Sessions based on mappings have only one step, while sessions based on packages can have multiple steps.

Using the Debugging Cursor

You can place the cursor *before* or *after* the execution of a line. The debugger highlights half a cursor position before or after the line.

Figure 8-1 shows a cursor position before the execution of a line:

Figure 8-1 Cursor Positioned Before a Step



Debug Actions

You can perform debug actions for a selected cursor position.

Step Into

The **Step Into** action steps to the beginning of the first child node of the currently selected node. If the currently selected node does not have a child, this is disabled.

The Step Into action works for all things including steps and tasks.

Pause

The **Pause** action suspends the current execution. When the execution is paused, you can modify the code of the tasks that are yet to be executed, and that modified code is taken up when you resume the execution.

The execution can be continued by stepping through the steps/tasks, or by resuming execution.

Resume

The **Resume** action resume execution at the current cursor and continues until the session is finished or a breakpoint is reached.

Run to Task End

The **Run to Task End** action runs to the end of the currently selected task. If the task has children, execute all children until the end of the task is reached.

This action is disabled if the cursor is not at the beginning of a task.

Run to Next Task

The **Run to Next Task** action runs to the beginning of the next task. If the cursor is at the last task in a list of tasks, it goes to the end of the parent task.

Run to Step End

The **Run to Step End** action runs to the end of the current step, and executes all tasks until the end of the step is reached.

Run to Next Step

The **Run to Next Step** action runs to the beginning of the next step.

Multiple Cursors

You can use multiple cursors for in-session parallelism.

Special Behavior

A parallel container task opens a separate cursor for each child task, and a cursor that remains on the container task until the last child task has finished. Each child cursor can be used independently to step through tasks, use breakpoints, or view data or variables. The child cursor disappears when the child task is finished. Performing the **resume** command on a child cursor only progresses to the end of the child task.

Using Breakpoints

You can set breakpoints on blueprint steps and tasks to suspend execution for debugging purposes.

About the Debug Breakpoints Window

Use **Debug Breakpoints** to view the list of all breakpoints.

About Design vs. Runtime Breakpoints

There are two lists of breakpoints, *runtime breakpoints*, and *design breakpoints*.

- Runtime breakpoints can be added in the blueprint tab of a Session Editor. Runtime breakpoints on a blueprint are used across sessions that are started from the same blueprint. A scenario execution will reuse a blueprint as long as the scenario remains unchanged.
- Design breakpoints are defined in a design artifact such as package or procedure. When executing, a design breakpoint generates a runtime breakpoint in the blueprint it is associated with. You can add design breakpoints in the package editor.

Placing Breakpoints

To add a breakpoint:

1. Select **Run** > **Breakpoint** > **Add Breakpoint**.

2. A red dot is placed in the blueprint to represent the breakpoint.
3. Start or resume the execution.
4. When the executions suspends at the breakpoint, inspect the following:
   - debugging data. See Debugging Variables.
   - variable values. See Debugging Variables.
   - debugging session threads. See Debugging Threads.

Editing Breakpoints

To edit a breakpoint:

1. Select **Run** > **Breakpoint** > **Edit Breakpoint**.
2. The Breakpoint Properties window appears.
3. Set the following properties.
   - **Enabled**: checked if the breakpoint is enabled.
     - if checked, the breakpoint is active and suspends execution.
     - if unchecked, the breakpoint is ignored during execution.
   - **Suspend after executing the task**:
     - if checked, the breakpoint suspends after executing the task or step.
     - if unchecked, the breakpoint suspends before executing the task or step.
   - **Pass count**: the number of times the breakpoint has to be passed before suspending execution; 0 suspends the execution on the initial pass, 1 passes the breakpoint once before suspending execution.
4. Click **OK**.

Removing Breakpoints

To remove a single breakpoint:

- Right-click on the step or task and select **Remove Breakpoint**.

To remove all breakpoints (available only from the breakpoint window):

- Select **Run** > **Breakpoint** > **Remove All**.

Enabling and Disabling Breakpoints

To enable or disable a single breakpoint, select:

- **Run** > **Breakpoint** > **Enable**, or,
- **Run** > **Breakpoint** > **Disable**.

To enable or disable all breakpoints (available only from the breakpoint window), select:

- **Run** > **Breakpoint** > **Enable All**, or,
- **Run** > **Breakpoint** > **Disable All**.

Pass Count

The pass count indicates the number of times the breakpoint has to be passed before suspending execution; 0 suspends the execution on every pass; 1 suspends the execution only on the first pass; 5 suspends the execution only on the fifth pass.

Debugging Data

Use the Debug Data tab to query data in the context of the current debugger task.

The data window has two tabs, Source Task Data and Target Task Data, to query the data servers associated with the source and the target for the current task. The **Get Data** command in the main toolbar inserts the SQL code of the current task into both Source Task Data and Target Task Data fields. The window provides a field to enter a SQL command and execute it by clicking the **Run SQL Code** button.

SQL commands use the transaction context of the current session, uncommitted data can be queried. If there are multiple tasks at debug execution in the session, the commands use the context of the task synced to the data window.

Get Data

Get Data inserts the SQL code of the currently selected task into the SQL command field of the Debug Data window. Both the Source Task Data and Target Task Data windows are populated.

The **Get Data** command associates with the original cursor line; if you keep stepping the data window remains attached to the original cursor line. You must press **Get Data** again to go to the new current cursor line.

Source Task Data

Typically, the task code for Source command contains a Select query to retrieve data from source. By default, this query is displayed in the Source Task Data field. You must review and modify the query before executing, as the content may be inappropriate to execute unchanged.

Target Task Data

The Target Task Data typically contains a DML operation such as Insert or Update. Oracle Data Integrator attempts to parse the target SQL to identify the target table name and provide a simple query, such as:

```
select * from <parsedTargetTableName>
```

If Oracle Data Integrator is unable to parse the target SQL to identify the target table name, a suggested query appears, such as:

```
select * from <SpecifyTargetTableName>
```

You must review and modify the query before executing, as the content may be inappropriate to execute unchanged.

Run SQL Code

You are required to execute **Get Data** before doing a **Run SQL Code**. You can run queries to view or select data, but you cannot not apply changes to the data.

Editing SQL Code

To edit SQL code and query data:

1. Select the **Debug Data** tab in a session.
2. Select **Get Data** to insert the SQL code of the current task into the Source Task Data and Target Task Data fields.
3. Edit the SQL command. You can use a `select` statement to query the data on the target uncommitted transaction.
4. Click **Run SQL Code**.

Debugging Variables

Use the **Debug Variables** tab to view all the variables used in the current session, and see how a variable may change as you step through the steps and tasks. You can change the value of a variable to affect the execution of the session.

You can view the following information for each variable:

Properties	Description
Name	Name of the variable.
Value	Value of the variable. Can be modified to change execution.
Type	Type of the variable.

Modifying Variables

To modify a variable:

1. Set breakpoints, if required. See Using Breakpoints.
2. Position the cursor in the blueprint.
3. In the **Debug Variables** tab, select the variable whose value you want to modify.
4. Start or resume the execution of the session.

Debugging Threads

Use the **Debug Threads** tab to see all threads for the connected Session Editor in focus. The icon of the thread shows whether the thread is suspended or executing.

Go To Source

The **Go To Source** context menu command on each thread jumps to the thread location inside the blueprint.

Managing Debugging Sessions

You can see a list of all active debug sessions in the Debug Session window by selecting **Window** > **Debugger** > **Debug Sessions**.

From the Debug Sessions window, you can connect or disconnect the listed sessions.

Properties	Description
Agent	Agent executing the session.
Session Number	ID of the session.
Session Name	Name of the session.
Session Connected	Shows whether the session is connected or not connected to the debugger client.

Stop Normal and Stop Immediate

You can disconnect a session normally or immediately:

1. In the Debug Sessions window, right-click a session in the list and select **Disconnect Debug Session**.

   The session becomes active in the Studio.
2. Select the **Disconnect** button in the upper left of the session window, and choose:
   - **Stop Normal**: The session is stopped once the current task is finished.
   - **Stop Immediate**: The current task is immediately interrupted and the session is stopped. This mode allows to stop long-running tasks, as for example long SQL statements before they complete.
3. Confirm the disconnection by clicking **OK**.
4. When the session is stopped, you see the message 'Session debugging completed'. The session is removed from the Debug Sessions list.

See Stopping a Session.

Restart Execution

In the session window, you can restart a stopped session.

To restart a session:

1. In the upper left of the session window, select **Restart Execution**.

   The Restart Execution window appears.
2. To select the agent to execute the session, select one of the following options:
   - **Use the Previous Agent**: **<agent name>** to use the agent that was used to execute a previous session.
   - **Choose another Agent**: to select the agent that you want to use to execute the session. Select **Internal** to use the ODI Studio built-in agent.
3. Select the Log Level. Log level 6 has the same behavior as log level 5, but with the addition of variable and sequence tracking.

4. Click **OK** to restart the indicated session and to close the dialog.

   You see an informational message, 'Session Launched', and the session is added to the Debug Sessions list.

To restart a failed session in debug mode:

1. In the navigator tree, select the failed session.
2. Right-click and select **Debug**.
3. In the **Debug** window, configure options described in Debug Session Options.
4. Click **OK**.

   An information message indicates that the session has restarted.

9

Monitoring Integration Processes

This chapter describes how to manage your development executions in Operator Navigator. An overview of the Operator Navigator's user interface is provided.
This chapter includes the following sections:

- Introduction to Monitoring
- Monitoring Executions Results
- Managing your Executions

Introduction to Monitoring

Monitoring your development executions consists of viewing the execution results and managing the development executions when the executions are successful or in error. This section provides an introduction to the monitoring features in Oracle Data Integrator. How to work with your execution results is covered in Monitoring Executions Results. How to manage your development executions is covered in Managing your Executions.

Introduction to Operator Navigator

Through Operator Navigator, you can view your execution results and manage your development executions in the sessions, as well as the scenarios and Load Plans in production.

Operator Navigator stores this information in a work repository, while using the topology defined in the master repository.

Operator Navigator displays the objects available to the current user in six navigation trees:

- **Session List** displays all sessions organized per date, physical agent, status, keywords, and so forth
- **Hierarchical Sessions** displays the execution sessions also organized in a hierarchy with their child sessions
- **Load Plan Executions** displays the Load Plan Runs of the Load Plan instances
- **Scheduling** displays the list of logical agents and schedules
- **Load Plans and Scenarios** displays the list of scenarios and Load Plans available
- **Solutions** displays the list of solutions

The Operator Navigator Toolbar Menu

You can perform the main monitoring tasks via the Operator Navigator Toolbar menu. The Operator Navigator toolbar menu provides access to the features detailed in Table 9-1.

Table 9-1 Operator Navigator Toolbar Menu Items

Icon	Menu Item	Description
	Refresh	Click **Refresh** to refresh the trees in the Operator Navigator.
 	Filter Filter activated	Click **Filter** to define the filters for the sessions to display in Operator Navigator.
	Auto Refresh	Click **Auto Refresh** to refresh automatically the trees in the Operator Navigator.
	Connect Navigator	Click **Connect Navigator** to access the Operator Navigator toolbar menu. Through the Operator Navigator toolbar menu you can: • Import a scenario • Import and export the log • Perform multiple exports • Purge the log • Display the scheduling information • Clean stale sessions • Remove temporary objects

Scenarios

A *scenario* is designed to put a source component (mapping, package, procedure, variable) into production. A scenario results from the generation of code (SQL, shell, etc.) for this component.

When a scenario is executed, it creates a *Session*.

Scenarios are imported into production environment and can be organized into *Load Plan and Scenario* folders. See Managing Scenarios and Load Plansfor more details.

Sessions

In Oracle Data Integrator, an execution results in a *Session*. Sessions are viewed and managed in Operator Navigator.

A *session* is an execution (of a scenario, a mapping, a package or a procedure, and so forth) undertaken by an execution agent. A session is made up of *steps* which are themselves made up of *tasks*.

A *step* is the unit of execution found between a task and a session. It corresponds to a step in a package or in a scenario. When executing a mapping or a single variable, for example, the resulting session has only one step.

Two special steps called *Command On Connect* and *Command On Disconnect* are created if you have set up On Connect and Disconnect commands on data servers used in the session. See Setting Up On Connect/Disconnect Commandsfor more information.

The *task* is the smallest execution unit. It corresponds to a command in a KM, a procedure, and so forth.

Sessions can be grouped into *Session folders*. Session folders automatically group sessions that were launched with certain keywords. Refer to Organizing the Log with Session Foldersfor more information.

Note that certain privileges are required for you to stop a session or clean stale sessions. These privileges are listed below:

You can stop the session if you have one of the following Sessions privileges:

- You are a Supervisor user.
- You have started the session and you are the owner of the session.
- You have been explicitly assigned the Stop Immediate or the Stop Normal privileges. Note that you can view these method objects in the Security Navigator by expanding the Objects tree and expanding the Session Object node of a given profile.

You can clean stale sessions if you have one of the following Sessions privileges:

- You are a Supervisor user.
- You have been explicitly assigned the Clean Stale Sessions privileges. Note that you can view this method object in the Security Navigator by expanding the Objects tree and expanding the Agent Object node of a given profile.

You can also stop a session or clean stale sessions if the OPERATOR or NG OPERATOR profile is assigned to you.

Load Plans

A *Load Plan* is the most course-grained type of executable object in Oracle Data Integrator, used to organize and run finer-grained objects. It uses *Scenario*s in its steps. A Load Plan is an organized hierarchy of child steps. This hierarchy allows conditional processing of steps in parallel or in series.

Load Plans are imported into production environments and can be organized into *Load Plan and Scenario* folders. See Managing Scenarios and Load Plansfor more details.

Load Plan Executions

Executing a Load Plan creates a *Load Plan instance* and the first *Load Plan run* for the instance. This Load Plan instance is separated from the original Load Plan and can be modified independently. Every time a Load Plan instance is restarted, a *Load Plan run* is created for this Load Plan instance. A Load Plan run corresponds to an attempt to execute

the instance. See the Load Plan Execution Lifecycle section in *Developing Integration Projects with Oracle Data Integrator* for more information.

When running, a Load Plan Run starts sessions corresponding to the scenarios sequenced in the Load Plan.

Note that in the list of Load Plan executions, only the Load Plan runs appear. Each run is identified by a Load Plan Instance ID and an Attempt (or Run) Number.

Schedules

You can *schedule* the executions of your scenarios and Load Plans using Oracle Data Integrator's built-in scheduler or an external scheduler. Both methods are detailed in Scheduling Scenarios and Load Plans.

The schedules appear in Designer and Operator Navigator under the **Scheduling** node of the scenario or Load Plan. Each schedule allows a start date and a repetition cycle to be specified.

Log

The Oracle Data Integrator log corresponds to all the Sessions and Load Plan instances/runs stored in a repository. This log can be exported, purged or filtered for monitoring. See Managing the Log for more information.

Status

A session, step, task or Load Plan run always has a status. Table 9-2 lists the six possible status values:

Table 9-2 Status Values

Status Name	Status Icon for Sessions	Status Icon for Load Plans	Status Description
Done			The Load Plan, session, step or task was executed successfully.
Done in previous run			The Load Plan step has been executed in a previous Load Plan run. This icon is displayed after a restart.
Error			The Load Plan, session, step or task has terminated due to an error.

Table 9-2 (Cont.) Status Values

Status Name	Status Icon for Sessions	Status Icon for Load Plans	Status Description
Running			The Load Plan, session, step or task is being executed.
Waiting			The Load Plan, session, step or task is waiting to be executed.
Warning (Sessions and tasks only)			• For Sessions: The session has completed successfully but errors have been detected during the data quality check. • For Tasks: The task has terminated in error, but since errors are allowed on this task, this did not stop the session.
Queued (Sessions only)			The session is waiting for an agent to be available for its execution

When finished, a session takes the status of the last executed step (**Done** or **Error**). When finished, the step, takes the status of the last executed task (Except if the task returned a Warning. In this case, the step takes the status **Done**).

A Load Plan is successful (status **Done**) when all its child steps have been executed successfully. It is in **Error** status when at least one of its child steps is in error and has raised its exception to the root step.

Task Types

Tasks are the nodes inside of steps in a session. The types of tasks are shown in Table 9-3.

Table 9-3 Task Types

Task Type	Task Icon for Sessions	Task Description
Normal Task		This task is executed according to its sequential position in the session. It is marked to DONE status when completed successfully. If it completes in error status, it is failed and marked with the error status.

Table 9-3 (Cont.) Task Types

Task Type	Task Icon for Sessions	Task Description
Serial Task		The children tasks are executed in a sequential order. The serial task is completed and marked to DONE status when all its children tasks have completed successfully. It is considered failed and marked with error status when the first child task completes in error status.
Parallel Task		The children tasks are executed concurrently. The parallel task is considered completed and marked with DONE status when all its children tasks have completed successfully. It is considered failed and marked with ERROR status if any of its child tasks has failed.

Monitoring Executions Results

In Oracle Data Integrator, an execution results in a *session* or in a *Load Plan run* if a Load Plan is executed. A *session* is made up of steps which are made up of tasks. Sessions are viewed and managed in Operator Navigator.

Load Plan runs appear in the Operator Navigator. To review the steps of a Load Plan run, you open the editor for this run. The sessions attached to a Load Plan appear with the rest of the sessions in the Operator Navigator.

Monitoring Sessions

To monitor your sessions:

1. In the Operator Navigator, expand the Session List navigation tree.
2. Expand the All Executions node and click **Refresh** in the Navigator toolbar.
3. Optionally, activate a Filter to reduce the number of visible sessions. For more information, see Filtering Sessions.
4. Review in the list of sessions the status of your session(s).

Monitoring Load Plan Runs

To monitor your Load Plan runs:

1. In the Operator Navigator, expand the Load Plan Executions navigation tree.
2. Expand the All Executions node and click **Refresh** in the Navigator toolbar.
3. Review in the list the status of your Load Plan run.
4. Double-click this Load Plan run to open the Load Plan Run editor.
5. In the Load Plan Run editor, select the Steps tab.
6. Review the state of the Load Plan steps. On this tab, you can perform the following tasks:
   - Click **Refresh** in the Editor toolbar to update the content of the table.

- For the Run Scenario steps, you can click in the Session ID column to open the session started by this Load Plan for this step.

Handling Failed Sessions

When your session ends in error or with a warning, you can analyze the error in Operator Navigator.

To analyze an error:

1. In the Operator Navigator, identify the session, the step and the task in error.
2. Double click the task in error. The Task editor opens.
3. On the Definition tab in the Execution Statistics section, the return code and message give the error that stopped the session. The metrics about the performance of the loading task and any configuration data that have an impact on performance are displayed in the Execution Details section.
4. On the Code tab, the source and target code for the task is displayed and can be reviewed and edited.

   Optionally, click **Show/Hide Values** to display the code with resolved variable and sequence values. Note that:

   - If the variable values are shown, the code becomes read-only. You are now able to track variable values.
   - Variables used as passwords are never displayed.

   See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator* for more information.
5. On the Connection tab, you can review the source and target connections against which the code is executed.

You can fix the code of the command in the Code tab and apply your changes. Restarting a session (see Restarting a Session) is possible after performing this action. The session will restart from the task in error.

> **Note:**
>
> Fixing the code in the session's task does not fix the source object that was executed (mapping, procedure, package or scenario). This source object must be fixed in Designer Navigator and the scenario (if any) must be regenerated. Modifying the code within the session is useful for debugging issues.

> **WARNING:**
>
> When a session fails, all connections and transactions to the source and target systems are rolled back. As a consequence, uncommitted statements on transactions are not applied.

Reviewing Successful Sessions

When your session ends successfully, you can view the changes performed in Operator Navigator. These changes include record statistics such as the number of inserts, updates, deletes, errors, and the total number of rows as well as execution statistics indicating start and end time of the execution, the duration in seconds, the return code, and the message (if any).

Session level statistics aggregate the statistics of all the steps of this session, and each step's statistics aggregate the statistics of all the tasks within this step.

To review the execution statistics:

1. In the Operator Navigator, identify the session, the step, or the task to review.
2. Double click the session, the step, or the task. The corresponding editor opens.
3. The record and execution statistics are displayed on the Definition tab.

Record Statistics

Properties	Description
No. of Inserts	Number of rows inserted during the session/step/task.
No. of Updates	Number of rows updated during the session/step/task.
No. of Deletes	Number of rows deleted during the session/step/task.
No. of Errors	Number of rows in error in the session/step/task.
No. of Rows	Total number of rows handled during this session/step/task.

Execution Statistics

Properties	Description
Start	Start date and time of execution of the session/step/task.
End	End date and time of execution of the session/step/task.
Duration (seconds)	The time taken for execution of the session/step/task.
Return code	Return code for the session/step/task.
Execution Details	The metrics about the performance of the loading task and any configuration data that have an impact on performance.

For session steps in which a mapping has been executed or a datastore check has been performed, the target table details are displayed. Also, for session steps in which a mapping has been executed, the Knowledge Modules used in the mapping are displayed in the Knowledge Module Details section of the Session Step Editor.

If tracking is enabled for a session, information regarding the variable or sequence is displayed in the Variable and Sequence Values section of the Session Step Editor and Session Task Editor.

You can view the options selected to limit concurrent executions of a session in the Concurrent Execution Controller section of the Session Editor. See the Controlling Concurrent Execution of Scenarios and Load Plans section in *Developing Integration Projects with Oracle Data Integrator* for more information.

Handling Failed Load Plans

When a Load Plan ends in error, review the sessions that have failed and caused the Load Plan to fail. Fix the source of the session failure.

You can restart the Load Plan instance. See Restarting a Load Plan Runfor more information.

Note that it will restart depending on the Restart Type defined on its steps. See the Handling Load Plan Exceptions and Restartability section in *Developing Integration Projects with Oracle Data Integrator* for more information.

You can also change the execution status of a failed Load Plan step from **Error** to **Done** on the Steps tab of the Load Plan run Editor to ignore this particular Load Plan step the next time the Load Pan run is restarted. This might be useful, for example, when the error causing this Load Plan step to fail is not possible to fix at the moment and you want to execute the rest of the Load Plan regardless of this Load Plan step.

Reviewing Successful Load Plans

When your Load Plan ends successfully, you can review the execution statistics from the Load Plan run editor.

You can also review the statistics for each session started for this Load Plan in the Session Editor.

To review the Load Plan run execution statistics:

1. In the Operator Navigator, identify the Load Plan run to review.
2. Double click the Load Plan run. The corresponding editor opens.
3. The record and execution statistics are displayed on the Steps tab.

Managing your Executions

Managing your development executions takes place in Operator Navigator. You can manage your executions during the execution process itself or once the execution has finished depending on the action that you wish to perform. The actions that you can perform are:

- Managing Sessions
- Managing Load Plan Executions
- Managing the Log
- Managing Scenarios and Load Plans
- Managing Schedules

Managing Sessions

Managing sessions involves the following tasks

- New sessions can be created by executing run-time objects or scenarios. See Running Integration Processesfor more information on starting sessions.
- Sessions in progress can be aborted. How to stop sessions is covered in Stopping a Session.

- Sessions failed, or stopped by user action can be restarted. Restarting sessions is covered in Restarting a Session.

In addition to these tasks, it may be necessary in production to deal with stale sessions.

Cleaning Stale Sessions

Stale sessions are sessions that are incorrectly left in a running state after an agent or repository crash.

The agent that started a session automatically detects when this session becomes stale and changes it to *Error* status. You can manually request specific agents to clean stale sessions in Operator Navigator or Topology Navigator.

To clean stale sessions manually:

1. Do one of the following:
   - From the Operator Navigator toolbar menu, select **Clean Stale Sessions**.
   - In Topology Navigator, from the Physical Architecture navigation tree, select an agent, right-click and select **Clean Stale Sessions**.

   The Clean Stale Sessions Dialog opens.
2. In the Clean Stale Sessions Dialog specify the criteria for cleaning stale sessions:
   - From the list, select the agents that will clean their stale sessions.

     Select **Clean all Agents** if you want all agents to clean their stale sessions.
   - From the list, select the Work Repositories you want to clean.

     Select **Clean all Work Repositories** if you want to clean stale sessions in all Work Repositories.
3. Click **OK** to start the cleaning process. A progress bar indicates the progress of the cleaning process.

Note that certain privileges are required for you to clean stale sessions. See Sessionsfor information.

Removing Temporary Objects

To remove temporary objects that could remain between executions:

1. Select **Remove temporary objects** from the Operator Navigator toolbar menu.
2. In the Remove Temporary Objects dialog set the criteria listed in Table 9-4.

Table 9-4 Remove Temporary Objects Dialog Parameters

Parameter	Description
Session Count	Number of sessions for which to skip cleanup. If the count is zero, all sessions that match the filter criteria are cleaned up.
From	Start date for the cleanup. All sessions started after this date are cleaned up.
To	End date for the cleanup. All sessions started before this date are cleaned up.

Table 9-4 (Cont.) Remove Temporary Objects Dialog Parameters

Parameter	Description
Context	Cleans up only those sessions executed in this context.
Agent	Cleans up only those sessions executed by this agent.
User	Cleans up only those sessions launched by this user.
Session Name	Name of the session.

3. Click **OK**.

Managing Load Plan Executions

Managing Load Plan Executions involves the following tasks:

- New Load Plan Instances and Runs can be created by executing Load Plans. See Executing a Load Planfor more information on starting Load Plans.
- Load Plan Runs in progress can be aborted. How to stop Load Plan runs is covered in Stopping a Load Plan Run.
- Load Plan Runs failed, or stopped by user action can be restarted. Restarting Load Plan Runs is covered in Restarting a Load Plan Run.

Managing the Log

Oracle Data Integrator provides several solutions for managing your log data:

- Filtering Sessions to display only certain execution sessions in Operator Navigator
- Purging the Log to remove the information of past sessions
- Organizing the Log with Session Folders
- Exporting and Importing Log Data for archiving purposes
- Runtime Logging for ODI components (ODI Studio, ODI Java EE agent, ODI Standalone agent, and ODI Standalone Colocated agent)

Filtering Sessions

Filtering log sessions allows you to display only certain sessions in Operator Navigator, by filtering on parameters such as the user, status or duration of sessions. Sessions that do not meet the current filter are hidden from view, but they are not removed from the log.

To filter out sessions:

1. In the Operator Navigator toolbar menu, click **Filter**. The Define Filter editor opens.
2. In the Define Filter Editor, set the filter criteria according to your needs. Note that the default settings select all sessions.
   - **Session Number**: Use blank to show all sessions.
   - **Session Name**: Use `%` as a wildcard. For example `DWH%` matches any session whose name begins with `DWH`.
   - Session's execution **Context**
   - **Agent** used to execute the session

- **User** who launched the session
- **Status**: Running, Waiting etc.
- **Date** of execution: Specify either a date **From** or a date **To**, or both.

  While filtering to view running sessions in any version of ODI Operator, specify only the (From) field and leave the (To) field blank.

  If the (From) and (To) fields are specified, sessions that started after the (From) date and finished before the (To) date will be identified . Since running sessions do not have a known finish time, the (To) field should be left null. This will allow for running sessions to be identified.
- **Duration greater than** a specified number of seconds

3. Click **Apply** for a preview of the current filter.
4. Click **OK**.

Sessions that do not match these criteria are hidden in the Session List navigation tree. The Filter button on the toolbar is activated.

To deactivate the filter click **Filter** in the Operator toolbar menu. The current filter is deactivated, and all sessions appear in the list.

Purging the Log

Purging the log allows you to remove past sessions and Load Plan runs from the log. This procedure is used to keeping a reasonable volume of sessions and Load Plans archived in the work repository. It is advised to perform a purge regularly. This purge can be automated using the OdiPurgeLog tool (see: the OdiPurgeLog section in *Oracle Data Integrator Tools Reference*) in a scenario.

To purge the log:

1. From the Operator Navigator toolbar menu select **Connect Navigator** > **Purge Log...** The Purge Log editor opens.
2. In the Purge Log editor, set the criteria listed in Table 9-5 for the sessions or Load Plan runs you want to delete.

Table 9-5 Purge Log Parameters

Parameter	Description
Purge Type	Select the objects to purge.
From ... To	Sessions and/or Load Plan runs in this time range will be deleted. When you choose to purge session logs only, then the sessions launched as part of the Load Plan runs are not purged even if they match the filter criteria.When you purge Load Plan runs, the Load Plan run which matched the filter criteria and the sessions launched directly as part of the Load Plan run and its child/grand sessions will be deleted.
Context	Sessions and/or Load Plan runs executed in this context will be deleted.
Agent	Sessions and/or Load Plan runs executed by this agent will be deleted.

Table 9-5 (Cont.) Purge Log Parameters

Parameter	Description
Status	Session and/or Load Plan runs in this status will be deleted.
User	Sessions and/or Load Plan runs executed by this user will be deleted.
Name	Sessions and/or Load Plan runs matching this session name will be deleted. Note that you can specify session name masks using % as a wildcard.
Purge scenario reports	If you select **Purge scenario reports**, the scenario reports (appearing under the execution node of each scenario) will also be purged.

Only the sessions and/or Load Plan runs matching the specified filters will be removed:

- When you choose to purge session logs only, then the sessions launched as part of the Load Plan runs are not purged even if they match the filter criteria.
- When you purge Load Plan runs, the Load Plan run which matched the filter criteria and the sessions launched directly as part of Load Plan run and its child/grand sessions will be deleted.
- When a Load Plan run matches the filter, all its attached sessions are also purged irrespective of whether they match the filter criteria or not.

3. Click **OK**.

Oracle Data Integrator removes the sessions and/or Load Plan runs from the log.

> **Note:**
>
> It is also possible to delete sessions or Load Plan runs by selecting one or more sessions or Load Plan runs in Operator Navigator and pressing the **Delete** key. Deleting a Load Plan run in this way, deletes the corresponding sessions.

Organizing the Log with Session Folders

You can use **session folders** to organize the log. Session folders automatically group sessions and Load Plan Runs that were launched with certain keywords. Session folders are created under the **Keywords** node on the Session List or Load Plan Executions navigation trees.

Each session folder has one or more keywords associated with it. Any session launched with all the keywords of a session folder is automatically categorized beneath it.

To create a new session folder:

1. In Operator Navigator, go to the Session List or Load Plan Executions navigation tree.
2. Right-click the **Keywords** node and select **New Session Folder**.
3. Specify a **Folder Name**.
4. Click **Add** to add a keyword to the list. Repeat this step for every keyword you wish to add.

> **Note:**
>
> Only sessions or load plans with all the keywords of a given session folder will be shown below that session folder. Keyword matching is case sensitive.

Table 9-6 lists examples of how session folder keywords are matched.

Table 9-6 Matching of Session Folder Keywords

Session folder keywords	Session keywords	Matches?
DWH, Test, Batch	Batch	No - all keywords must be matched.
Batch	DWH, Batch	Yes - extra keywords on the session are ignored.
DWH, Test	Test, dwh	No - matching is case-sensitive.

To launch a session with keywords, you can for example start a scenario from a command line with the `-KEYWORDS` parameter. Refer to Running Integration Processesfor more information.

> **Note:**
>
> Session folder keyword matching is dynamic. If the keywords for a session folder are changed or if a new folder is created, existing sessions are immediately re-categorized.

Exporting and Importing Log Data

Export and import log data for archiving purposes.

Exporting Log Data

Exporting log data allows you to export log files for archiving purposes.

To export the log:

1. Select **Export...** from the Designer, Topology, Security or Operator Navigator toolbar menu.
2. In the Export Selection dialog, select **Export the Log**.
3. Click **OK**.
4. In the Export the log dialog, set the log export parameters as described in Table 9-7.

Table 9-7 Log Export Parameters

Properties	Description
Export to directory	Directory in which the export file will be created.

Table 9-7 (Cont.) Log Export Parameters

Properties	Description
Export to zip file	If this option is selected, a unique compressed file containing all log export files will be created. Otherwise, a set of log export files is created.
Zip File Name	Name given to the compressed export file.
Filters	**This set of options allow to filter the log files to export according to the specified parameters.**
Log Type	From the list, select for which objects you want to retrieve the log. Possible values are: `All|Load Plan runs and attached sessions|Sessions`
From / To	Date of execution: specify either a date From or a date To, or both.
Agent	Agent used to execute the session. Leave the default **All Agents** value, if you do not want to filter based on a given agent.
Context	Session's execution Context. Leave the default **All Contexts** value, if you do not want to filter based on a context.
Status	The possible states are `Done`, `Error`, `Queued`, `Running`, `Waiting`, `Warning` and `All States`. Leave the default **All States** value, if you do not want to filter based on a given session state.
User	User who launched the session. Leave the default **All Users** value, if you do not want to filter based on a given user.
Session Name	Use `%` as a wildcard. For example `DWH%` matches any session whose name begins with `DWH`.
Encryption	These fields allow you to provide an Export Key, used to encrypt any sensitive data that is contained in the exported object. See the Export Keys section in *Developing Integration Projects with Oracle Data Integrator* for details.
Export Key	Specifies the AES KEY for any sensitive data encryption needed during the export. The export key string is minimum 8 characters long and maximum 100 characters long. It should have at least one special character (@#$%+/=) or digit, and at least one alphabetic lower or upper case character.
Confirm Export Key	Enter your Export Key again.
Save Export Key	If checked, your Export Key is saved for all future exports.
Advanced options	**This set of options allow to parameterize the output file format.**
Character Set	Encoding specified in the export file. Parameter encoding in the XML file header. `<?xml version="1.0" encoding="ISO-8859-1"?>`
Java Character Set	Java character set used to generate the file.

5. Click **OK**.

The log data is exported into the specified location.

Note that you can also automate the log data export using the OdiExportLog tool (see: the OdiExportLog section in *Oracle Data Integrator Tools Reference*).

Importing Log Data

Importing log data allows you to import into your work repository log files that have been exported for archiving purposes.

To import the log:

1. Select **Import...** from the Designer, Topology, Security or Operator Navigator toolbar menu.
2. In the Import Selection dialog, select **Import the Log**.
3. Click **OK**.
4. In the Import of the log dialog:
5. a. Select the **Import Mode**. Note that sessions can only be imported in Synonym Mode INSERT mode. Refer to the Import Modes section in *Developing Integration Projects with Oracle Data Integrator* for more information.
   b. Select whether you want to import the files **From a Folder** or **From a ZIP file.**
   c. Enter the file import folder or zip file.
   d. Click **OK**.
   e. If prompted, enter the **Export Key** used when the log data was exported. If you do not enter an Export Key, any encrypted sensitive (cipher) data will be stripped from the imported log data. For more information about the Export Key, see the Export Keys section in *Developing Integration Projects with Oracle Data Integrator*.

The specified folder or ZIP file is imported into the work repository.

Runtime Logging for ODI components

You can set up runtime logging to trace ODI components or set a verbose level to investigate different issues or to monitor the system. The following ODI components can be traced: ODI Studio, ODI Java EE agents, ODI Standalone agents, and ODI Standalone Colocated agents.

ODI Studio - For Local Agent which runs in ODI Studio, debug service is enabled and logs are added in `SNP_SESSION_DBG` table.

> **Note:**
>
> A verbose logging will slow down the ODI performance.

Log Level

The log level can be set against a log_handler and/or logger elements. Note the following when setting the log level in the log configuration file for the ODI component:

- If it is set against a log_handler, then it applies to all usages of the log_handler.
- If it is set against a logger, then it applies to all of its handlers and any descendent loggers that do not have an explicit level setting.
- A message is logged if its log level is:

- Greater or equal than (>=) the level of its logger AND
- Greater or equal than (>=) the level of its log_handler

ODI components use the Oracle Java Debugging Levels (OJDL). Table 9-8 shows the mapping between the Java log levels and the Oracle Java Debugging Levels.

Table 9-8 Mapping between Java Log Levels and Oracle Java Debugging Levels

Java Log Levels	Oracle Java Debugging Levels
SEVERE intValue()+100	INCIDENT_ERROR:1
SEVERE	ERROR:1
WARNING	WARNING:1
INFO	NOTIFICATION:1
CONFIG	NOTIFICATION:16
FINE	TRACE:1
FINER	TRACE:16
FINEST	TRACE:32

Setting Up Runtime Logging

To set up runtime logging you have to enable different ODI loggers and log handlers. For ODI Java EE agents and ODI Standalone Colocated agents, this can be done through the logging configuration mechanism within Oracle Enterprise Manager Console. For ODI Studio and Standalone agents, set the related log level in the ODI logging system configuration file, as shown below:

1. Open the ODI logging system configuration file of the ODI component.

   Each component has its own configuration file:

   - ODI Studio:

     ```
     $ODI_HOME/odi/studio/bin/ODI-logging-config.xml
     ```

   - ODI Standalone agent:

     ```
     <DOMAIN_HOME>/config/fmwconfig/components/ODI/<INSTANCE_NAME>/ODI-logging-config.xml
     ```

2. Make sure that the path to your log files is a valid and existing path. For example:

   ```
   <log_handler name="ODI-file-handler"
   class="oracle.core.ojdl.logging.ODLHandlerFactory"
   level="ALL">
     <property name="format" value="ODL-Text"/>
     <property name="path" value="/u01/oracle/odi11g/oracledi/agent/log/${LOG_FILE}"/>
     <property name="maxFileSize" value="1000000"/> <!-- in bytes -->
     <property name="maxLogSize" value="50000000"/> <!-- in bytes -->
     <property name="encoding" value="UTF-8"/>
   </log_handler>
   ```

   Note the following concerning the log files path:

   - If you are on Windows, the path could be for example:

     ```
     %ODI_HOME%\oracledi\agent\log\${LOG_FILE}
     ```

   - You can use a relative path on Windows and Unix.

3. Enable the logger and set the log level. For example, for logger *oracle.odi.agent* you can enable the most verbose logging setting:

```
<logger name="oracle.odi.agent" level="TRACE:32" useParentHandlers="false">
  <handler name="ODI-file-handler"/>
  <handler name="ODI-console-handler"/>
</logger>
```

4. Save the configuration file and restart ODI Studio and the ODI Standalone agent for the changes to take effect.

Example INFO (NOTIFICATION:1) Message

The `INFO (NOTIFICATION:1)` Message is logged if:

- The logger level (possibly inherited) is `<= NOTIFICATION:1`
- The log_handler level is `<= NOTIFICATION:1` (for example: `TRACE:1`)

The `INFO (NOTIFICATION:1)` Message is *not* logged if:

- The logger level (possibly inherited) is `> NOTIFICATION:1`
- The log_handler level is `> NOTIFICATION:1` (for example: `WARNING:1`)

The Runtime logger is called *oracle.odi.agent*. Its message levels cover the following content:

```
NOTIFICATION:1 (or INFO) Agent Level
NOTIFICATION:16 (or CONFIG) the above + Session Level
TRACE:1 (or FINE) the above + Step Level
TRACE:16 (or FINER) the above + Task + SQL
TRACE:32 (or FINEST) the above + more detail
```

It is recommended that the console level for *oracle.odi.agent* is set so that agent startup messages are displayed (`NOTIFICATION:1`).

Managing Scenarios and Load Plans

You can also manage your executions in Operator Navigator by using scenarios or Load Plans.

Before running a scenario, you need to generate it in Designer Navigator or import from a file. See the Using Scenarios chapter in *Developing Integration Projects with Oracle Data Integrator*. Load Plans are also created using Designer Navigator, but can also be modified using Operator Navigator. See the Using Load Plans chapter in *Developing Integration Projects with Oracle Data Integrator* for more information.

Launching a scenario from Operator Navigator is covered in Executing a Scenario from ODI Studio, and how to run a Load Plan is described in Executing a Load Plan.

Load Plan and Scenario Folders

In Operator Navigator, scenarios and Load Plans can be grouped into Load Plan and Scenario folders to facilitate organization. Load Plan and Scenario folders can contain other Load Plan and Scenario folders.

To create a Load Plan and Scenario folder:

1. In Operator Navigator go to the Load Plans and Scenarios navigation tree.

2. From the Load Plans and Scenarios toolbar menu, select **New Load Plan and Scenario Folder**.
3. On the Definition tab of the Load Plan and Scenario Folder editor enter a name for your folder.
4. From the File menu, select **Save**.

You can now reorganize your scenarios and Load Plans. Drag and drop them into the Load Plan and Scenario folder.

Importing Load Plans, Scenarios, and Solutions in Production

A Load Plan or a scenario generated from Designer can be exported and then imported into a development or execution repository. This operation is used to deploy Load Plans and scenarios in a different repository, possibly in a different environment or site.

Importing a Load Plan or scenario in a development repository is performed via Designer or Operator Navigator. With a execution repository, only Operator Navigator is available for this purpose.

See the Importing Scenarios in Production section in *Developing Integration Projects with Oracle Data Integrator* for more information on how to import a scenario in production, and the Importing Load Plans section in *Developing Integration Projects with Oracle Data Integrator* for more information on the Load Plan import.

Similarly, a solution containing several scenarios can be imported to easily transfer and restore a group of scenarios at once. See the Using Version Control chapter in *Developing Integration Projects with Oracle Data Integrator* for more information. Note that when connected to an execution repository, only scenarios may be restored from solutions.

Managing Schedules

A schedule is always attached to one scenario or one Load Plan. Schedules can be created in Operator Navigator. See Scheduling Scenarios and Load Plansfor more information.

You can also import an already existing schedule along with a scenario or Load Plan import. See the Importing Scenarios in Production and Exporting, Importing and Versioning Load Plans sections in *Developing Integration Projects with Oracle Data Integrator* for more information.

You can view the scheduled tasks of all your agents or you can view the scheduled tasks of one particular agent. See Displaying the Schedulefor more information.

10

Using Oracle Data Integrator Console

This chapter describes how to work with Oracle Data Integrator Console. An overview of the Console user interface is provided.
This chapter includes the following sections:

Introduction to Oracle Data Integrator Console

Oracle Data Integrator Console is a web-based console for managing and monitoring an Oracle Data Integrator run-time architecture and for browsing design-time objects.

This section contains the following topic:

Oracle Data Integrator Console Concepts

Oracle Data Integrator Console is a web-based console available for different types of users:

- Administrators use Oracle Data Integrator Console to create and import repositories and to configure the Topology (data servers, schemas, and so forth).
- Production operators use Oracle Data Integrator Console to manage scenarios and Load Plans, monitor sessions and Load Plan runs, and manage the content of the error tables generated by Oracle Data Integrator.
- Business users and developers browse development artifacts in this interface, using, for example, the Data Lineage and Flow Map features.

This web interface integrates seamlessly with Oracle Fusion Middleware Control Console and allows Fusion Middleware administrators to drill down into the details of Oracle Data Integrator components and sessions.

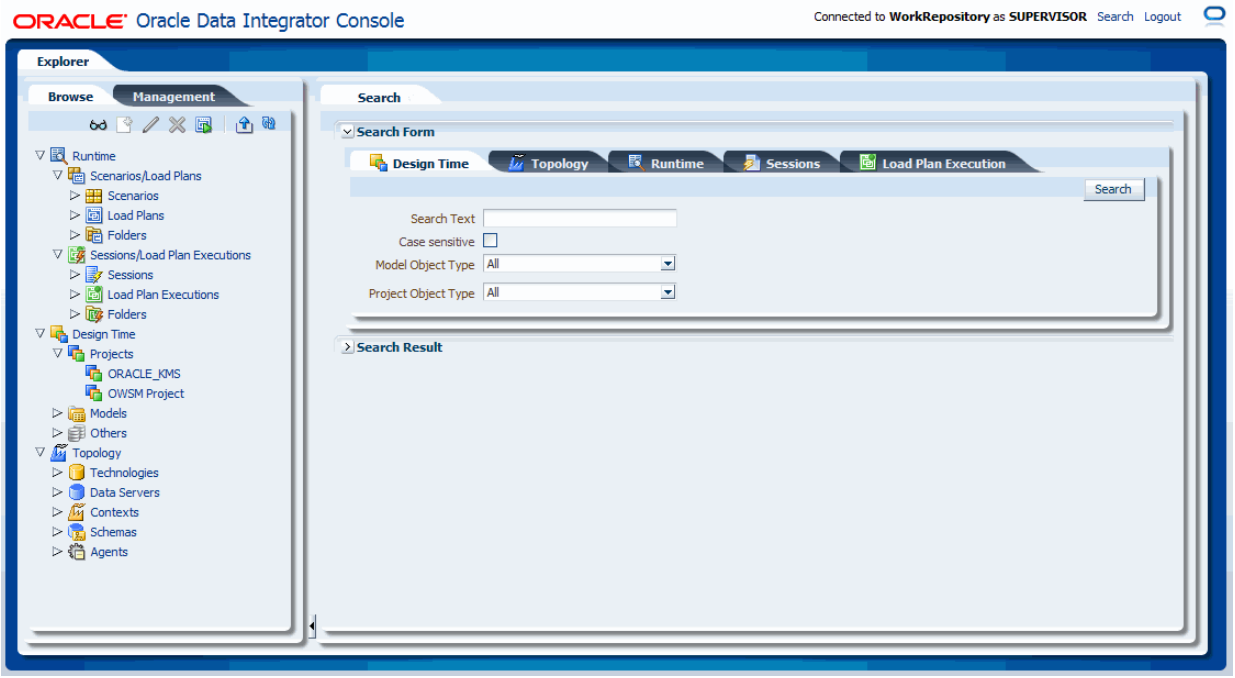
> **Note:**
>
> Oracle Data Integrator Console is required for the Fusion Middleware Control Extension for Oracle Data Integrator. It must be installed and configured for this extension to discover and display the Oracle Data Integrator components in a domain.

Oracle Data Integrator Console Interface

Oracle Data Integrator Console is a web interface using the ADF-Faces framework.

Figure 10-1 shows the layout of Oracle Data Integrator Console.

Figure 10-1 Oracle Data Integrator Console



Oracle Data Integrator Console displays the objects available to the current user in two Navigation tabs in the left panel:

- **Browse** tab displays the repository objects that can be browsed and edited. In this tab you can also manage sessions and error tables.
- Management tab is used to manage the repositories and the repository connections. This tab is available to connection users having Supervisor privileges, or to any user to set up the first repository connections.

The right panel displays the following tabs:

- Search tab is always visible and allows you to search for objects in the connected repository.
- One Master/Details tab is displayed for each object that is being browsed or edited. Note that it is possible to browse or edit several objects at the same time.

The search field above the Navigation tabs allows you to open the search tab when it is closed.

Working with the Navigation Tabs

In the Navigation tabs, you can browse for objects contained in the repository. When an object or node is selected, the Navigation Tab toolbar displays icons for the actions

available for this object or node. If an action is not available for this object, the icon is grayed out. For example, you can edit and add data server objects under the Topology node in the Browse Tab, but you cannot edit Projects under the Designer node. Note that the number of tabs that you can open at the same time is limited to ten.

Using Oracle Data Integrator Console

This section explains the different types of operations available in Oracle Data Integrator Console. It does not focus on each type of object that can be managed with the console, but gives keys to manage objects with the console.

This section includes the following topics:

- Connecting to Oracle Data Integrator Console
- Generic User Operations
- Managing Scenarios and Sessions
- Managing Load Plans
- Purging the Log
- Using Data Lineage and Flow Map
- Performing Administrative Operations

> **Note:**
>
> Oracle Data Integrator Console uses the security defined in the master repository. Operations that are not allowed for a user will appear grayed out for this user.
>
> In addition, the **Management** tab is available only for users with Supervisor privileges.

Connecting to Oracle Data Integrator Console

Oracle Data Integrator console connects to a repository via a Repository Connection, defined by an administrator.

Note that you can only connect to Oracle Data Integrator Console if it has been previously installed. See *Installing and Configuring Oracle Data Integrator* for more information about installing Oracle Data Integrator Console.

> **Note:**
>
> The first time you connect to Oracle Data Integrator Console, if no repository connection is configured, you will have access to the **Management** tab to create a first repository connection. See Creating a Repository Connectionfor more information. After your first repository connection is created, the Management tab is no longer available from the Login page, and is available only for users with Supervisor privileges.

Connecting to Oracle Data Integrator Console

To connect to Oracle Data Integrator Console:

1. Open a web browser, and connect to the URL where Oracle Data Integrator Console is installed. For example: `http://odi_host:8001/odiconsole/`.
2. From the Repository list, select the Repository connection corresponding to the master or work repository you want to connect.
3. Provide a **User ID** and a **Password**.
4. Click **Sign In**.

Generic User Operations

This section describes the generic operations available in Oracle Data Integrator Console for a typical user.

This section includes the following operations:

> **Note:**
>
> Creating, editing, and deleting operations are not allowed for Scenarios and Load Plans. For more information on the possible actions that can be performed with these objects in ODI Console, see Managing Scenarios and Sessionsand Managing Load Plans.

- Viewing an Object
- Editing an Object
- Creating an Object
- Deleting an Object
- Searching for an Object

Viewing an Object

To view an object:

1. Select the object in the **Browse** or **Management** Navigation tab.
2. Click **View** in the Navigation tab toolbar. The simple page or the Master/Detail page for the object opens.

Editing an Object

To edit an object:

1. Select the object in the **Browse** or **Management** Navigation tab.
2. Click **Update** in the Navigation tab toolbar. The edition page for the object opens.
3. Change the value for the object fields.
4. Click **Save** in the edition page for this object.

Creating an Object

To create an object:

1. Navigate to the parent node of the object you want to create in the **Browse** or **Management** Navigation tab. For example, to create a Context, navigate to the **Topology** > **Contexts** node in the **Browse** tab.
2. Click **Create** in the Navigation tab toolbar. An **Add** dialog for this object appears.
3. Provide the values for the object fields.
4. Click **Save** in the Add dialog of this object. The new object appears in the Navigation tab.

Deleting an Object

To delete an object:

1. Select the object in the **Browse** or **Management** Navigation tab.
2. Click **Delete** in the Navigation tab toolbar.
3. Click OK in the confirmation window.

Searching for an Object

To search for an object:

1. In the **Search** tab, select the tab corresponding to the object you want to search:
   - **Design Time** tab allows you to search for design-time objects
   - **Topology** tab allows you to search for topology objects
   - **Runtime** tab allows you to search for run-time objects such as Load Plans, Scenarios, Scenario Folders, or Session Folders
   - **Sessions** tab allows you to search for sessions
   - **Load Plan Execution** tab allows you to search for Load Plan runs
2. Set the search parameters to narrow your search.

   For example when searching design-time or topology objects:

   a. In the **Search Text** field, enter a part of the name of the object that you want to search.

   b. Select **Case sensitive** if you want the search to be case sensitive (this feature is not provided for the sessions or Load Plan execution search.

   c. Select in **Models/Project** (Designer tab) or **Topology** (Topology tab) the type of object you want to search for. Select **All** to search for all objects.
3. Click **Search**.
4. The **Search Result**s appear, grouped by object type. You can click an object in the search result to open its master/details page.

Managing Scenarios and Sessions

This section describes the operations related to scenarios and sessions available in Oracle Data Integrator Console.

This section includes the following operations:

- Importing a Scenario
- Exporting a Scenario
- Running a Scenario
- Stopping a Session
- Restarting a Session
- Cleaning Stale Sessions
- Managing Data Statistics and Erroneous Records

Importing a Scenario

To import a scenario:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Scenarios/Load Plans > Scenarios**.
3. Click **Import** in the Navigation tab toolbar.
4. Select an **Import Mode** and select an export file in **Scenario XML File**.
5. Click **Import Scenario**.
6. If prompted, enter the **Export Key** used when this scenario was exported. If you do not enter an Export Key, any encrypted sensitive (cipher) data will be stripped from the imported object. For more information about the Export Key, see the Export Keys section in *Developing Integration Projects with Oracle Data Integrator*.

Exporting a Scenario

To export a scenario:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Scenarios/Load Plans > Scenarios**.
3. Click **Export** in the Navigation tab toolbar.
4. In the Export Scenario dialog, set the parameters as follows:
   - From the **Scenario Name** list, select the scenario to export.
   - In the **Encoding Java Charset** field, enter the Java character set for the export file.
   - In the **Encoding XML Charset** field, enter the encoding to specify in the export file.
   - In the **XML Version** field, enter the XML Version to specify in the export file.
   - In the **Export Key** field, enter the AES KEY for any sensitive data encryption needed during the export. The export key string is minimum 8 characters long and maximum 100 characters long. It should have at least one special character (@#$%+/=) or digit, one alphabetic lower or upper case character.
   - In the **Confirm Export Key** field, enter the export key again.
   - Optionally, select **Save export key for later exports** to save the export key for all future exports.
   - Optionally, select **Include Dependant objects** to export linked child objects.

5. Click **Export Scenario**.

Running a Scenario

To execute a scenario:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Scenarios/Load Plans > Scenarios**.
3. Select the scenario you want to execute.
4. Click **Execute** in the Navigation tab toolbar.
5. Select an **Agent**, a **Context**, and a **Log Level** for this execution.
6. Click **Execute Scenario**.

Stopping a Session

Note that you can perform a normal or an immediate kill of a running session. Sessions with the status *Done*, *Warning*, or *Error* cannot be killed.

To kill a session:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Sessions/Load Plan Executions > Sessions**.
3. Select the session you want to stop.
4. Click **Kill** in the Navigation tab toolbar.

Restarting a Session

To restart a session:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Sessions/Load Plan Executions > Sessions**.
3. Select the session you want to restart.
4. Click **Restart** in the Navigation tab toolbar.
5. In the Restart Session dialog, set the parameters as follows:
   - **Agent**: From the list, select the agent you want to use for running the new session.
   - **Log Level**: From the list, select the log level. Select **Log Level 6** in the Execution or Restart Session dialog to enable variable tracking. Log level 6 has the same behavior as log level 5, but with the addition of variable tracking.
6. Click **Restart Session**.

Cleaning Stale Sessions

To clean stale sessions:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Sessions/Load Plan Executions > Sessions**.
3. Click **Clean** in the Navigation tab toolbar.
4. In the **Clean Stale Sessions** dialog, select the **Agent** for which you want to clean stale sessions.

5. Click **OK**.

Managing Data Statistics and Erroneous Records

Oracle Data Integrator Console allows you to browse the details of a session, including the record statistics. When a session detects erroneous data during a flow or static check, these errors are isolated into error tables. You can also browse and manage the erroneous rows using Oracle Data Integrator Console.

> **Note:**
>
> Sessions with erroneous data detected finish in **Warning** status.

To view the erroneous data:

1. Select the **Browse** Navigation tab.
2. Navigate to a given session using **Runtime > Sessions/Load Plan Executions > Sessions**. Select the session and click **View** in the Navigation tab toolbar.

   The Session page is displayed.
3. In the Session page, go to the **Relationships** section and select the **Record Statistics** tab.

   This tab shows each physical table targeting in this session, as well as the record statistics.
4. Click the number shown in the **Errors** column. The content of the error table appears.
   - You can filter the errors by Constraint Type, Name, Message Content, Detection date, and so forth. Click **Filter Result** to apply a filter.
   - Select a number of errors in the **Query Results** table and click **Delete** to delete these records.
   - Click **Delete All** to delete all the errors.

> **Note:**
>
> Delete operations cannot be undone.

Managing Load Plans

This section describes the operations related to Load Plans available in Oracle Data Integrator Console.

This section includes the following operations:

- Importing a Load Plan
- Exporting a Load Plan
- Running a Load Plan

- Stopping a Load Plan Run
- Restarting a Load Plan Run

Importing a Load Plan

To import a Load Plan:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime** > **Scenarios/Load Plans** > **Load Plans**.
3. Click **Import** in the Navigation tab toolbar.
4. In the Import Load Plan dialog, select an **Import Mode** and select an export file in the **Select Load Plan XML File** field.
5. Click **Import**.
6. If prompted, enter the **Export Key** used when this scenario was exported. If you do not enter an Export Key, any encrypted sensitive (cipher) data will be stripped from the imported object. For more information about the Export Key, see the Export Keys section in *Developing Integration Projects with Oracle Data Integrator*.

> **Note:**
>
> When you import a Load Plan that has been previously exported, the imported Load Plan does not include the scenarios referenced by the Load Plan. Scenarios used in a Load Plan need to be imported separately. See Importing a Scenariofor more information.

Exporting a Load Plan

To export a Load Plan:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime** > **Scenarios/Load Plans** > **Load Plans**.
3. Select the Load Plan to export.
4. Click **Export** in the Navigation tab toolbar.
5. In the Export dialog, set the parameters as follows:
   - From the **Load Plan Name** list, select the Load Plan to export.
   - In the **Encoding Java Charset** field, enter the Java character set for the export file.
   - In the **Encoding XML Charset** field, enter the encoding to specify in the export file.
   - In the **XML Version** field, enter the XML Version to specify in the export file.
   - In the **Export Key** field, enter the AES KEY for any sensitive data encryption needed during the export. The export key string is minimum 8 characters long and maximum 100 characters long. It should have at least one special character (@#$%+/=) or digit, one alphabetic lower or upper case character.
   - In the **Confirm Export Key** field, enter the export key again.
   - Optionally, select **Save export key for later exports** to save the export key for all future exports.

- Optionally, select **Include Dependant objects** to export linked child objects.

6. Click **Export**.

> **Note:**
>
> The export of a Load Plan does not include the scenarios referenced by the Load Plan. Scenarios used in a Load Plan need to be exported separately. See Exporting a Scenariofor more information.

Running a Load Plan

To run a Load Plan:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime** > **Scenarios/Load Plans** > **Load Plans**.
3. Select the Load Plan you want to execute.
4. Click **Execute** in the Navigation tab toolbar.
5. Select a **Logical Agent**, a **Context**, a **Log Level**, and if your Load Plan uses variables, specify the **Startup values** for the Load Plan variables.
6. Click **Execute**.

Stopping a Load Plan Run

Note that you can perform a normal or an immediate kill of a Load Plan run. Any running or waiting Load Plan Run can be stopped.

To stop a Load Plan Run:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Sessions/Load Plan Executions > Load Plan Executions**.
3. Select the Load Plan run you want to stop.
4. Click **Kill** in the Navigation tab toolbar.

Restarting a Load Plan Run

A Load Plan can only be restarted if the selected run of the current Load Plan instance is in Error status and if there is no other instance of the same Load Plan currently running.

To restart a Load Plan Run:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Sessions/Load Plan Executions > Load Plan Executions**.
3. Select the Load Plan run you want to restart.
4. In the Restart Load Plan Dialog, select the **Physical Agent** that restarts the Load Plan. Optionally, select a different log level.
5. Click **Restart** in the Navigation tab toolbar.

Purging the Log

This section describes how to purge the log in Oracle Data Integrator Console by removing past sessions and/or Load Plan runs from the log.

To purge the log:

1. Select the **Browse** Navigation tab.
2. Navigate to **Runtime > Sessions/Load Plan Executions**.
3. Click **Purge** in the Navigation tab toolbar.
4. In the Purge Sessions/Load Plan Executions dialog, set the purge parameters listed in Table 10-1.

Table 10-1 Purge Log Parameters

Parameter	Description
Purge Type	Select the objects to purge.
From ... To	Sessions and/or Load Plan runs in this time range will be deleted. When you choose to purge session logs only, then the sessions launched as part of the Load Plan runs are not purged even if they match the filter criteria.When you purge Load Plan runs, the Load Plan run which matched the filter criteria and the sessions launched directly as part of the Load Plan run and its child/grand sessions will be deleted.
Context	Sessions and/or Load Plan runs executed in this context will be deleted.
Agent	Sessions and/or Load Plan runs executed by this agent will be deleted.
Status	Session and/or Load Plan runs in this status will be deleted.
User	Sessions and/or Load Plan runs executed by this user will be deleted.
Name	Sessions and/or Load Plan runs matching this session name will be deleted. Note that you can specify session name masks using % as a wildcard.
Purge scenario reports	If you select **Purge scenario reports**, the scenario reports (appearing under the execution node of each scenario) will also be purged.

Only the sessions and/or Load Plan runs matching the specified filters will be removed:

- When you choose to purge session logs only, then the sessions launched as part of the Load Plan runs are not purged even if they match the filter criteria.
- When you purge Load Plan runs, the Load Plan run which matched the filter criteria and the sessions launched directly as part of Load Plan run and its child/grand sessions will be deleted.
- When a Load Plan run matches the filter, all its attached sessions are also purged irrespective of whether they match the filter criteria or not.

5. Click **OK**.

Oracle Data Integrator Console removes the sessions and/or Load Plan runs from the log.

Using Data Lineage and Flow Map

This section describes how to use the Data Lineage and Flow Map features available in Oracle Data Integrator Console.

- **Data Lineage** provides graph displaying the flows of data from the point of view of a given datastore. In this graph, you can navigate back and forth and follow this data flow.
- **Flow Map** provides a map of the relations that exist between the data structures (models, sub-models and datastores) and design-time objects (projects, folders, packages, mappings). This graph allows you to draw a map made of several data structures and their data flows.

This section includes the following operations:

Working with the Data Lineage

To view the Data Lineage:

1. Select the **Browse** Navigation tab.
2. Navigate to **Design Time > Models > Data Lineage**.
3. Click **View** in the Navigation tab toolbar.
4. In the Data Lineage page, select a Model, then a Sub-Model and a datastore in this model.
5. Select **Show Mappings** if you want that mappings are displayed between the datastores nodes.
6. Select the prefix to add in your datastores and mapping names in the **Naming Options** section.
7. Click **View** to draw the Data Lineage graph. This graph is centered on the datastore selected in step 4.

   In this graph, you can use the following actions:

   - Click **Go Back** to return to the Data Lineage options and redraw the graph.
   - Use the **Hand** tool and then click a datastore to redraw the lineage centered on this datastore.
   - Use the **Hand** tool and then click a mapping to view this mapping's page.
   - Use the **Arrow** tool to expand/collapse groups.
   - Use the **Move** tool to move the graph.
   - Use the **Zoom In/Zoom Out** tools to resize the graph.
   - Select **View Options** to change the display options have the graph refreshed with this new option.

Working with the Flow Map

To view the Flow Map:

1. Select the **Browse** Navigation tab.
2. Navigate to **Design Time > Models > Flow Map**.
3. Click **View** in the Navigation tab toolbar.
4. In the Data Lineage page, select one or more **Model**. Select **All** to select all models.
5. Select one of more **Projects**. Select **All** to select all projects.
6. In the **Select the level of details of the map** section, select the granularity of the map. The object that you select here will be the nodes of your graph.

   Check **Do not show Projects, Folders...** if you want the map to show only data structure.
7. Optionally, indicate the grouping for the data structures and design-time objects in the map, using the options in the **Indicate how to group Objects in the Map** section.
8. Click **View** to draw the **Flow Map** graph.

   In this graph, you can use the following actions:

   - Click **Go Back** to return to the Flow Map options and redraw the graph.
   - Use the **Hand** tool and then click a node (representing a datastore, an mapping, and so forth) in the map to open this object's page.
   - Use the **Arrow** tool to expand/collapse groups.
   - Use the **Move** tool to move the graph.
   - Use the **Zoom In/Zoom Out** tools to resize the graph.

Performing Administrative Operations

This section describes the different administrative operations available in Oracle Data Integrator Console. These operations are available for a user with Supervisor privileges.

This section includes the following operations:

- Creating a Repository Connection
- Testing a Data Server or a Physical Agent Connection
- Administering Repositories
- Administering Java EE Agents

Creating a Repository Connection

A *repository connection* is a connection definition for Oracle Data Integrator Console. A connection does not include Oracle Data Integrator user and password information.

To create a repository connection:

1. Navigate to the **Repository Connections** node in the **Management** Navigation tab.
2. Click **Create** in the Navigation tab toolbar. A **Create Repository Connection** dialog for this object appears.
3. Provide the values for the repository connection:
   - **Connection Alias**: Name of the connection that will appear on the Login page.
   - **Master JNDI URL**: JNDI URL of the datasource to connect the master repository database.

- **Supervisor User Name**: Name of the Oracle Data Integrator user with Supervisor privileges that Oracle Data Integrator Console will use to connect to the repository. This user's password must be declared in the WLS or WAS Credential Store.
- **Work JNDI URL**: JNDI URL of the datasource to connect the work repository database. If no value is given in this field. The repository connection will allow connection to the master only, and the Navigation will be limited to Topology information.
- **JNDI URL**: Check this option if you want to use the environment naming context (ENC). When this option is checked, Oracle Data Integrator Console automatically prefixes the data source name with the string `java:comp/env/` to identify it in the application server's JNDI directory. Note that the JNDI Standard is not supported by Oracle WebLogic Server and for global data sources.
- **Default**: Check this option if you want this Repository Connection to be selected by default on the login page.

4. Click **Save**. The new Repository Connection appears in the **Management** Navigation tab.

Testing a Data Server or a Physical Agent Connection

This sections describes how to test the data server connection or the connection of a physical agent in Oracle Data Integrator Console.

To test the data server connection:

1. Select the **Browse** Navigation tab.
2. Navigate to **Topology > Data Servers**.
3. Select the data server whose connection you want to test.
4. Click **Test Connection** in the Navigation tab toolbar.
5. In the Test Connection dialog, select the:
   - **Physical Agent** that will carry out the test
   - **Transaction** on which you want to execute the command. This parameter is only displayed if there is any On Connect/Disconnect command defined for this data server. The transactions from 0 to 9 and the **Autocommit** transaction correspond to connection created by sessions (by procedures or knowledge modules). The **Client Transaction** corresponds to the client components (ODI Console and Studio).
6. Click **Test**.

A dialog showing "Connection successful!" is displayed if the test has worked. If not, an error message is displayed.

To test the physical agent connection:

1. Select the **Browse** Navigation tab.
2. Navigate to **Topology > Agents > Physical Agents**.
3. Select the physical agent whose connection you want to test.
4. Click **Test Connection** in the Navigation tab toolbar.

A dialog showing "Connection successful!" is displayed if the test has worked. If not, an error message is displayed.

Administering Repositories

Oracle Data Integrator Console provides you with features to perform management operations (create, import, export) on repositories. These operations are available from the **Management** Navigation tab, under the **Repositories** node. These management operations reproduce in a web interface the administrative operations available via the Oracle Data Integrator Studio and allow setting up and maintaining your environment from the ODI Console.

See Administering Repositories and the Exporting and Importing section in*Developing Integration Projects with Oracle Data Integrator* for more information on these operations.

Administering Java EE Agents

Oracle Data Integrator Console allows you to add JDBC datasources and create templates to deploy physical agents into WebLogic Server.

See Setting Up a Topologyfor more information on Java EE agents, datasources and templates.

To add a datasource to a physical agent:

1. Select the **Browse** Navigation tab.
2. Navigate to **Topology > Agents > Physical Agents**.
3. Select the agent you want to manage.
4. Click **Edit** in the Navigation tab toolbar.
5. Click **Add Datasource**
6. Provide a **JNDI Name** for this datasource and select the **Data Server Name**. This datasource will be used to connect to this data server from the machine into which the Java EE agent will be deployed.
7. Click **OK**.
8. Click **Save** to save the changes to the physical agent.

To create a template for a physical agent:

1. Select the **Browse** Navigation tab.
2. Navigate to **Topology > Agents > Physical Agents**.
3. Select the agent you want to manage.
4. Click **Edit** in the Navigation tab toolbar.
5. Click **Agent Deployment**.
6. Follow the steps of the **Agent Deployment** wizard. This wizard reproduces in a web interface the Server Template Generation wizard. See Deploying an Agent in a Java EE Application Serverfor more details.

ODI Domain

An ODI domain contains the Oracle Data Integrator components that can be managed using Enterprise Manager. An ODI domain contains:

- One master repository and one or more work repositories attached to it.
- One or several run-time agents attached to the master repositories. These agents must be declared in the master repositories to appear in the domain. These agents may be Standalone agents, Standalone Colocated agents, or Java EE agents. See Setting Up a Topologyfor information about how to declare the agents in the master repositories.
- One or several Oracle Data Integrator Console applications. An Oracle Data Integrator Console application is used to browse master and work repositories.

The Master Repositories and Agent pages display both application metrics and information about the master and work repositories. You can also navigate to Oracle Data Integrator Console from these pages, for example to view the details of a session. In order to browse Oracle Data Integrator Console, the connections to the work and master repositories must be declared in Oracle Data Integrator Console. See *Installing and Configuring Oracle Data Integrator* for information on how to create and configure a domain.

Oracle Enterprise Manager Fusion Middleware Control

Oracle Enterprise Manager Fusion Middleware Control organizes a wide variety of performance data and administrative functions into distinct, Web-based home pages for the farm, cluster, domain, servers, components, and applications.

Oracle Data Integrator provides a plug-in that integrates with Oracle Enterprise Manager Fusion Middleware Control. Using this plug-in, Oracle Enterprise Manager Fusion Middleware Control can be used in conjunction with ODI Console to obtain information about your ODI agents, repositories, sessions, and load plan executions.

This section includes the following topics:

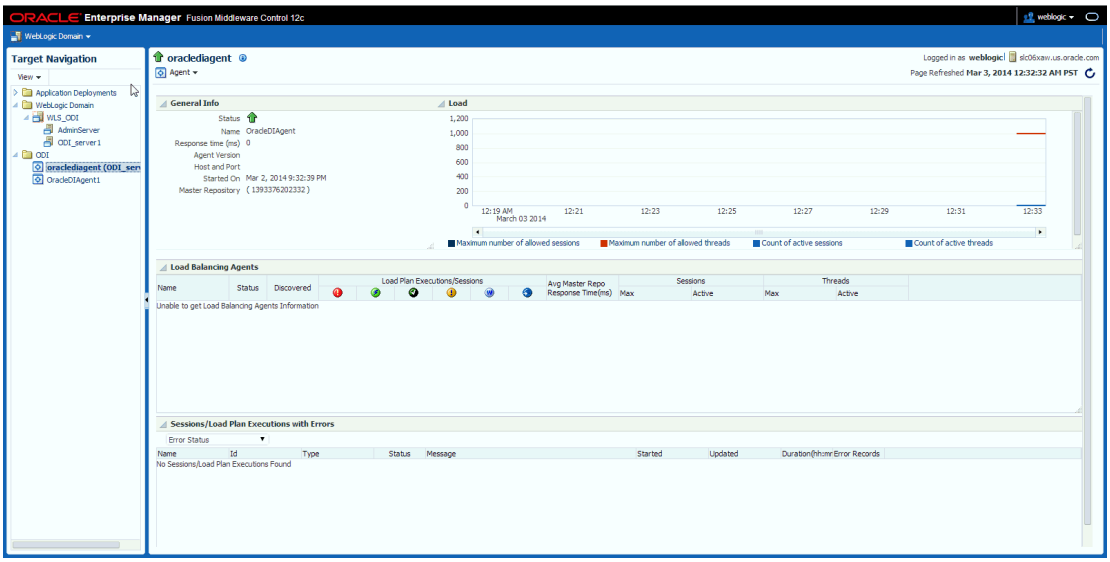
- Configuring Oracle Fusion Middleware Control with ODI Plug-in
- Configuring Oracle Data Integrator Console
- Managing Oracle Data Integrator
- Configuring ODI Standalone Agents
- Configuring Master Repository
- Searching Sessions
- Searching Load Plan Executions

Configuring Oracle Fusion Middleware Control with ODI Plug-in

From Oracle Enterprise Manager Fusion Middleware Control, expand the ODI menu item and click on your agent name in the Deployments pane.

Figure 10-2 shows the agents in the Deployment pane.

Figure 10-2 ODI Console Within Oracle Enterprise Manager Fusion Middleware Control



> **Note:**
>
> To use Oracle Enterprise Manager with Oracle Data Integrator Console, and your agent resides in a separate domain, you must first create the appropriate Credential Store Entries for Oracle Enterprise Manager. See the Specifying Supervisor Credentials section in *Installing and Configuring Oracle Data Integrator* for more information.

Domain discovery is performed with the following process:

1. Oracle Enterprise Manager parses the repository connections declared in Oracle Data Integrator Console, tries to connect all the master repositories available in this domain and retrieves their status and the list of agents. Even if an agent or repository is down, it will appear in the Oracle Enterprise Manager.
2. Any agent on the domain will appear in the domain with its status and will start posting notifications (if started).

> **Note:**
>
> If you want Oracle Enterprise Manager to drill down into Oracle Data Integrator Console using a different URL than the one detected by Oracle Enterprise Manager, you will need to reconfigure this in Oracle Enterprise Manager. Reconfiguration is not mandatory but may be needed when using a firewall for HTTP load balancing to Oracle Data Integrator Console.

For more information on using Oracle Enterprise Manager, see *Administering Oracle Fusion Middleware*.

Configuring Oracle Data Integrator Console

You can configure Oracle Data Integrator Console from Fusion Middleware Control to define the linking between Fusion Middleware Control and Oracle Data Integrator Console.

By default, the fields on this page are populated with the Oracle Data Integrator Console host, the Oracle Data Integrator Console managed server port, and the default context root. If your Oracle Data Integrator Console must be accessed with a different configuration, you can change the configuration on this page.

The steps for this process are:

1. Navigate to the ODI Agent home page.
2. From the **Agent** menu, select **Administration > ODI Console Administration**.
3. Enter the following parameters:
   - **Host:** The name of the server where your application is deployed
   - **Port:** The HTTP listener port number
   - **Context Root:** The Web application's context root
   - **Protocol**: The protocol for the connection
4. Select **Enable ODI-ESS Integration features in EM** to activate the ODI-ESS Integration features in Fusion Middleware Control.

Managing Oracle Data Integrator

The following provides a summary of the tasks performed when managing Oracle Data Integrator Console.

1. Start and stop Oracle Data Integrator Console. How?
2. Start and stop ODI agents. How?
3. Configure ODI Standalone agents. How?
4. Configure Master Repository. How?
5. Search sessions. How?
6. Search Load Plan executions. How?
7. View log messages. How?

Configuring ODI Standalone Agents

You can modify Oracle Data Integrator Standalone agent information. By default, the fields on this page are populated with information about the standalone agent.

The steps for this process are:

1. Navigate to the Standalone Agent home page.
2. From the **Agent** menu, select **Administration > ODI Standalone Agent Configuration**.
3. Enter new values to modify the ODI Standalone Agent configuration.

4. Click **Apply** to modify the configuration or click **Revert** to revert to the previous settings.

Configuring Master Repository

You can modify Master Repository Configuration parameters for standalone agents.

The steps for this process are:

1. Navigate to the Standalone Agent home page.
2. From the **Agent** menu, select **Administration > Master Repository Configuration**.
3. Enter new values to modify the Master Repository configuration.
4. Click **Apply** to modify the configuration or click **Revert** to revert to the previous settings.

Searching Sessions

You can search for sessions that have been executed in the managed ODI domain.

The steps for this process are:

1. Navigate to the Agent or Master Repository home page.
2. From the **Agent** menu, select **Search Sessions**.
3. The Search Sessions page opens. Enter the search criteria.
4. Click **Search**. The results display in the **Sessions** section.

Searching Load Plan Executions

You can search for Load Plan runs that have been executed in the managed ODI domain.

The steps for this process are:

1. Navigate to the Agent or Master Repository home page.
2. From the **Agent** menu, select **Search Load Plan Executions**.
3. The Search Load Plan Executions page opens. Enter the search criteria.
4. Click **Search**. The results display in the **Load Plan Executions** section.

Management Pack for Oracle Data Integrator

The Management Pack for Oracle Data Integrator leverages Oracle Enterprise Manager Cloud Control best-in-class application performance management, service level management and configuration management capabilities to provide a centralized management solution for Oracle Data Integrator Enterprise Edition.

The Management Pack for ODI provides a consolidated view of the ODI infrastructure and enables you to monitor and manage all the components centrally from Oracle Enterprise Manager Cloud Control.

Using the Management Pack for ODI, you can:

- Manage multiple Oracle Data Integrator domains from a single location. See ODI Domainfor more information regarding Oracle Data Integrator domains.

- Monitor the availability and performance of Oracle Data Integrator components, access historical data, track logs, and receive notifications of potential problems.
- Trace end-to-end Oracle Data Integrator Sessions activity, review execution statistics, and drill-down from a particular step or task into a detailed report of the Oracle Database activity.
- Control Service Level Agreements (SLA) with robust and scalable alerting capabilities.
- Obtain real-time and historical performance statistics for the Oracle Data Integrator Standalone (11*g*), Standalone Colocated (12*c*), and Java EE (11*g* and 12*c*) agents.
- Discover and model dependencies between Oracle Data Integrator and various components such as databases or other Oracle Fusion Middleware.
- Capture Oracle Data Integrator components configuration and track changes over time. Compare configuration parameters over time.

For information regarding the tasks you need to perform when managing Oracle Data Integrator, see the Configuring and Monitoring Oracle Data Integrator chapter in *Oracle Enterprise Manager Cloud Control Getting Started with Oracle Fusion Middleware Management Plug-in*.

Part IV
Managing Security Settings

This part describes how to manage the security settings in Oracle Data Integrator.

Part IV contains the following chapter:

- Managing Security in Oracle Data Integrator

11

Managing Security in Oracle Data Integrator

This chapter describes how to set up security in Oracle Data Integrator. It includes an overview of Oracle Data Integrator security concepts and components.
This chapter includes the following sections:

Introduction to Oracle Data Integrator Security

Oracle Data Integrator security is used to secure any action performed by authenticated users against the design-time and run-time artifacts and components of Oracle Data Integrator.

The Oracle Data Integrator security model is based on granting privileges on methods, objects types, or specific object instances to users.

All the security information for Oracle Data Integrator is stored in the master repository.

This section contains the following topics:

Objects, Instances, and Methods

An **object** is a representation of a design-time or run-time artifact handled through Oracle Data Integrator. Examples of objects include agents, projects, models, data stores, scenarios, mappings, and even repositories. Specific objects have a double name, such as Agent/Context and Profile/Method. These objects represent links between objects. These links are also objects. For instance, Agent/Context corresponds to a physical/logical agent association made through the contexts. Privileges on this object enable the ability to change this association in the topology.

An **instance** is a particular occurrence of an object. For example, the `Datawarehouse` project is an instance of the `Project` object.

A **method** is an action that can be performed on an object, such as edit, delete, and so on. Each object has a predefined set of methods.

> **Note:**
>
> The notions of object instance and method in Oracle Data Integrator are similar to the same concepts used in object-oriented programming.

> **Caution:**
>
> Although they appear in the Security Navigator, objects and methods are predefined in Oracle Data Integrator and should not be altered.

Profiles

A **profile** contains a set of privileges for working with Oracle Data Integrator. You can assign one or more profiles to a user, or to a role, to grant the sum of the privileges in those profiles to that user or role.

A **profile method** is an authorization granted to a profile on a method of an object type. Each granted method allows a user with this profile to perform an action on all instances of an object type.

In Oracle Data Integrator Studio, methods granted to a profile appear under this profile in the **Profiles** navigation tree of the Security Navigator. When a method does not appear for a given profile, this profile does not have access to this method.

A method can be granted as a generic or nongeneric privilege:

- A method granted as a generic privilege is granted by default on all the instances of this object.
- A method granted as a nongeneric privilege is not granted by default on all object instances, but may be granted on a per instance basis.

Generic vs. Nongeneric Profiles

Generic profiles have the generic privilege option selected for all object methods. This implies that a user, or role, with such a profile is by default authorized for all methods of all instances of an object to which the profile is authorized.

Nongeneric profiles are not by default authorized for all methods on the instances because the generic privilege option is not selected for all object methods. You must grant the user, or role, the rights on the methods for each instance.

If you want a user, or role, to have the rights on no instance by default, but want to grant the rights on a per instance basis, the user or role must be given a nongeneric profile.

If you want a user or role to have the rights on all instances of an object type by default, you must give the user or role a generic profile.

Built-In Profiles

Oracle Data Integrator contains built-in profiles that you can assign to the users or roles you create.

Table 11-1 shows the built-in profiles provided by Oracle Data Integrator.

Table 11-1 Built-In Profiles

Profile Name	Description
CONNECT	Profile granted with the basic privileges to connect to Oracle Data Integrator. It should be granted with another profile.
CONSOLE	Profile granted with privileges to use the Oracle Data Integrator Console.
DESIGNER	Profile granted with privileges to perform development operations. Use this profile for users who will work mainly on projects.
METADATA_ADMIN	Profile granted with privileges to manage metadata. Use this profile for users that will work mainly on models.
NG_CONSOLE	Nongeneric version of the CONSOLE profile.
NG_DESIGNER	Nongeneric version of the DESIGNER profile.
NG_METADATA_ADMIN	Nongeneric version of the METATADA_ADMIN profile.
NG_VERSION_ADMIN	Nongeneric version of the VERSION_ADMIN profile.
OPERATOR	Profile granted with privileges to manage run-time objects. Use this profile for production users.
RELEASE_MANAGER	Profile granted with privileges to perform release management tasks through deployment archives.
SECURITY_ADMIN	Profile granted with privileges to edit security. Use this profile for security administrators.
TOPOLOGY_ADMIN	Profile granted with privileges to edit the Topology. Use this profile for system or Oracle Data Integrator administrators.
VCS_VERSION_ADMIN	Profile granted with privileges to configure SVN systems, create tags, and branches.
VERSION_ADMIN	Profile granted with privileges to create, restore and edit versions and solutions. Use this profile for project managers, or developers who are entitled to perform version management operations.

Table 11-1 (Cont.) Built-In Profiles

Profile Name	Description
DESIGN_REVIEWER	Profile granted with privileges to execute a mapping in simulation mode. Such users should not be able to perform a real execution, but should be able to simulate an execution using the simulation run mode only (i.e. code-review analysis).

> **Note:**
>
> Oracle recommends not modifying built-in profiles because they evolve to secure new features of Oracle Data Integrator. If you want to customize your own profiles or existing profiles, Oracle recommends that you create copies of the built-in profiles and then customize those copies.

Users

A **user** is an Oracle Data Integrator user. A user inherits the following privileges:

- All the privileges granted to its various profiles
- Privileges on objects or instances, or both, given to this user

A **user method** is a privilege granted to a user on a method of an object type. Each granted method allows the user to perform an action (edit, delete, and so forth) on instances of an object type (project, model, data store, and so forth). These methods are similar to profiles methods, but applied to users.

You can grant users with privileges on instances on specific work repositories where these instances exist. For example, you may grant a developer user with the edit privilege on the LOAD_DATAWAREHOUSE scenario on the a DEVELOPMENT repository and not on a PRODUCTION repository.

You grant an authorization by object instance to a user on an object instance. It allows you to grant to this user certain methods of this object instance.

An authorization by object instance for a given instance that appears in a user's tree indicates that the user is granted specific privileges on the object methods for the given instance. You specify these privileges in the object instance editor. If an instance is not visible in the tree of the user instances, then the User Method or Profile Method privileges for the object type apply.

Because an instance may be replicated over the different work repositories that are attached to the master repository, you can define authorizations by object instances for one, several, or all your work repositories that are attached to this master repository. For example, you can replicate a LOAD_DATAWAREHOUSE scenario instance (using, for example, versioning) in the DEVELOPMENT, TEST, and PRODUCTION repositories. Privileges on the instance can change depending on the repository. For example, it is common to replicate projects from a development repository to a test repository. You can grant edit privileges to a developer for that developer's project in the development repository, but not in the test repository. In the test repository, the developer is granted with only view privileges on the project.

Roles

If Oracle Data Integrator is configured to use **external authentication** (see Configuring External Authentication), you can leverage the **enterprise role integration** feature in Oracle Data Integrator. Enterprise role integration in Oracle Data Integrator is based on the authorization model in Oracle Platform Security Services (OPSS). This feature allows you to map enterprise roles to Oracle Data Integrator roles, enabling enterprise users of a particular enterprise role to access Oracle Data Integrator through the permissions granted to the Oracle Data Integrator roles in the mapping.

Enterprise role integration in Oracle Data Integrator enables you to:

- Create a set of Oracle Data Integrator roles.
- Grant Oracle Data Integrator privileges (instance level and type level) and profiles to Oracle Data Integrator roles.
- Map enterprise principals — that is, users or roles — defined in an external identity store to Oracle Data Integrator roles.
- Determine the privileges granted to a user who is authenticated into Oracle Data Integrator by evaluating all the Oracle Data Integrator roles to which the user is mapped directly or indirectly through enterprise roles. (Privileges are automatically calculated during authorization.).

For an introduction to enterprise roles, see the Understanding Users and Roles chapter in *Securing Applications with Oracle Platform Security Services*. For details about how enterprise role integration works in Oracle Data Integrator and how you can manage roles in Oracle Data Integrator Studio, see Mapping Principals Defined in an Identity Store to Oracle Data Integrator Roles.

User Management

This section explains how to use the different security capabilities to enforce security in Oracle Data Integrator.

This section contains the following topics:

Security Policy Approaches

Two main approaches are available for defining the security in Oracle Data Integrator:

- The strongly secured approach, where users have no default authorizations on objects. This approach uses only nongeneric profiles. The security administrator must grant users authorizations on object instances. This policy is complex to configure because it requires managing privileges by instance.
- The generic approach, where users inherit the privileges of the profiles they have. This policy is suitable in most cases and is simple to configure.

To implement a strongly secured approach:

1. Create the users.
2. Give the users nongeneric profiles (built-in or customized).
3. Grant the users privileges on object instances after those instances are created. This operation must be repeated for every new instance.

To implement a generic approach:

1. Create the users.
2. Give the users the generic profiles (built-in or customized).

> **Note:**
>
> It is possible to combine the two approaches by simultaneously using generic and non generic profiles. For example, by using DESIGNER and NG_METADATA_ADMIN for the users, you manage projects in a generic approach, and manage models in a strongly secured approach.

Managing Profiles

You can create, delete, or duplicate profiles. Creating and duplicating profiles enables you to customize the profiles assigned to users. The following topics provide the steps for performing these procedures in Oracle Data Integrator Studio.

Creating a New Profile

To create a profile:

1. In Security Navigator, expand the **Profiles** navigation tree.
2. Click **New Profile** in the toolbar of the **Profiles** navigation tree.
3. In the **Name** field, enter a name for your profile.
4. From the **File** main menu, select **Save**.

The new profile appears.

Duplicating a Profile

To duplicate a profile:

1. In Security Navigator expand the **Profiles** navigation tree.
2. Select the profile that you want to duplicate from the list of profiles.
3. Right-click and select **Duplicate**.

A new profile appears, which is a copy of the original profile.

Deleting a Profile

To delete a profile:

1. In Security Navigator expand the **Profiles** navigation tree.
2. Select the profile that you want to delete from the list of profiles.

3. Right-click and select **Delete**.
4. Click **OK** in the Confirmation dialog.

The profile disappears from the list. All users granted with this profile lose the privileges attached to this profile.

Managing Users

The way in which you manage users depends, in part, on where the user is defined. By default, a user corresponds to the login name used to connect to a repository and is persisted in the internal repository of Oracle Data Integrator. However, if external authentication is enabled for Oracle Data Integrator components:

- Users authenticated into Oracle Data Integrator are created and managed in an external identity store, such as Oracle Internet Directory. You use the tools and utilities provided with that store to create, update, and delete users.
- To let external users access Oracle Data Integrator, you can do any of the following:
  - Map the external user to an Oracle Data Integrator role.
  - Map any of the enterprise roles to which this external user belongs to an Oracle Data Integrator role.
  - Register the external user in Oracle Data Integrator using the Security Navigator as in prior releases.
- You can assign profiles or other security privileges directly to external users who are registered in Oracle Data Integrator. Or you can assign profiles or other security privileges to an Oracle Data Integrator role, and then map external users (or any of the external users' enterprise roles) to that Oracle Data Integrator role. By doing the latter, an external user can inherit all the profiles or privileges granted to the Oracle Data Integrator role.

  You can also map an enterprise user (or the user's enterprise roles) to multiple Oracle Data Integrator roles. When a user is authenticated, Oracle Data Integrator calculates the user's privileges as a union of all the privileges granted to all the Oracle Data Integrator roles in the mapping.

For more information about external authentication, see Configuring External Authentication. For more information about enterprise role integration, see Mapping Principals Defined in an Identity Store to Oracle Data Integrator Roles.

Creating a New User in the Internal Repository

To create a user in the internal repository of Oracle Data Integrator:

1. In Security Navigator expand the **Users** navigation tree.
2. Click **New User** in the toolbar of the **Users** navigation tree.
3. In the **Name** field, enter a name for your user.
4. Provide the **Initials** for this user.
5. Do one of the following:
   - If using internal authentication, click **Enter the Password** and provide a password for this user.

- If using external authentication, click **Retrieve GUID** to associate the new user with a user from the external identity store. The user name in Oracle Data Integrator must match the user name in the identity store. If a match is found, a Global Unique ID appears in the **External GUID** field.

6. From the **File** main menu, select **Save**.

The new user appears in the **Users** navigation tree. From the User editor, an ODI supervisor user can disable other ODI user accounts including other supervisor accounts.

> **WARNING:**
>
> At least one supervisor user account should be maintained at all times. If all supervisor user accounts expire, contact Oracle Support.

Assigning a Profile to a User

To assign a profile to a user:

1. In Security Navigator expand the **Users** and the **Profiles** navigation trees.
2. Select the profile that you want to assign, then drag it on the user you want to assign it to.
3. Click **OK** in the Confirmation dialog.

The profile appears under the profiles of this user. The user is immediately granted the privileges attached to this profile.

> **Note:**
>
> If external authentication is enabled, you can assign profiles to roles. Users who are defined in a given Oracle Data Integrator role are granted all the profiles assigned to that role. Enterprise role integration gives you a more convenient and coarse-grained approach to managing the privileges granted to users. For more information, see Assign Privileges or Profiles to Oracle Data Integrator Roles.

Removing a Profile from a User

To remove a profile from a user:

1. In Security Navigator expand the **Users** navigation tree.
2. Expand the **Profiles** node under the user.
3. Right-click the profile to be removed.
4. Select **Delete**.
5. Click **OK** in the Confirmation dialog.

The profile disappears from the list of profiles assigned to this user. All privileges granted to this user by this profile are removed.

> **Note:**
>
> If external authentication is enabled, you can remove profiles from roles. Users who are defined in a given Oracle Data Integrator role are granted only the profiles assigned to that role. For more information, see Assign Privileges or Profiles to Oracle Data Integrator Roles.

Deleting a User from the Internal Repository

To delete a user that is defined in the internal repository of Oracle Data Integrator:

1. In Security Navigator expand the **Users** navigation tree.
2. From the list of users, select the user that you want to delete.
3. Right-click and select **Delete**.
4. Click **OK** in the Confirmation window.

The user disappears from the list.

Viewing User Connection Information

To view user connection information:

1. From the Security Navigator toolbar menu, select **Review User Activity...**. The User Connections dialog appears.
2. Enter the filter criteria to view relevant user connection information and click **Filter**.

User connections are displayed in the Connections section, based on the set filter criteria.

Managing Privileges

After creating users and profiles, you can grant privileges to these users and profiles. You can grant profile methods and user methods that apply to object types, and you can also grant authorizations by object instances (for users only) that apply to specific object instances.

> **Note:**
>
> If external authentication is enabled for Oracle Data Integrator components, you can manage privileges for users more conveniently by using the enterprise role integration feature. For more information, see Using the Roles Editor.

Granting a Profile Method or User Method

To grant a profile method or user method:

1. In Security Navigator expand the **Users** or **Profiles** navigation tree.
2. Expand the **Objects** navigation tree, and expand the node of the object for which you want to grant a privilege.

3. Select the method that you want to grant, then drag it on the user or profile you want to grant the method to.
4. Click **OK** in the Confirmation window.

The method is granted to the user or the profile. The method appears either under the objects node of this user or under the profile.

> **Note:**
>
> You can grant privileges on all the methods of an object by dragging the object itself onto the user or profile instead of dragging one of its methods.

Revoking a Profile Method or User Method

To revoke a profile method or user method:

1. In Security Navigator expand the **Users** or **Profiles** navigation tree.
2. Expand the **Profiles** or the **Objects** node under the user for which you want you revoke privileges, then expand the object whose method needs to be revoked.
3. Right-click the method and then select **Delete**.
4. Click **OK** in the Confirmation dialog.

The method is revoked from the user or the profile. It disappears from the methods list under the objects node of this user or profile.

Granting an Authorization by Object Instance

To grant an authorization by object instance to a user:

1. In Security Navigator expand the **Users** navigation tree.
2. In the Designer, Operator, or Topology Navigator, expand the navigation tree containing the object to which you want to grant privileges.
3. Select this object, then drag it onto the user in the **Users** navigation tree. The authorization by object instance editor appears. This editor shows the list of methods available for this instance and the instances it contains. For example, if you grant privileges on a project instance, the folders and mappings contained in this project appear in the editor.
4. Fine tune the privileges granted per object and method. You may want to implement the following simple privilege policies on methods that you select from the list:
   - To grant all these methods in all repositories, click **Allow selected methods in all repositories**.
   - To deny all these methods in all repositories, click **Deny selected methods in all repositories**.
   - To grant all these methods in certain work repositories, click **Allow selected methods in selected repositories**, then select the repositories from the list.
5. From the **File** main menu, select **Save**.

> **Note:**
>
> Only certain objects support authorization by object instance. These object types are listed under the **Instances** node for each user.
>
> Methods for which the user has generic privileges are not listed in the Object Instance Editor.

Revoking an Authorization by Object Instance

To revoke an authorization by object instance from a user:

1. In Security Navigator expand the **Users** navigation tree.
2. Expand the **Instances** node under the user for which you want you revoke privileges.
3. Right-click the instance from which you want to revoke an authorization, and then select **Delete**.
4. Click **OK** in the Confirmation dialog.

The authorizations on this object instance are revoked from the user.

> **Note:**
>
> You can also revoke privileges per method by editing the Authorization by Object instance and denying certain methods to this user. After this operation, if the user no longer has any privilege on an instance, the instance automatically disappears from the tree in Security Manager.

Cleaning up Unused Authorizations

Authorizations by object instance are stored in the master repository. However, if objects are deleted from all work repositories, the authorization are not necessarily deleted. You may wish to retain certain unused authorizations; for example, if they refer to objects currently stored in an exported file or in a stored solution.

You should use the Security Clean-up Tool periodically to remove these unused authorizations from the master repository. Unused authorizations are removed if they refer to objects that do not exist in the master repository or in any work repository.

> **Note:**
>
> All work repositories attached to the master repository must be accessible so that you can verify the existence of the objects in those repositories

To clean up unused authorizations:

1. From the Security Navigator toolbar menu, select **Clean Up Security Settings...**. The Security Clean-up Tool dialog appears.

2. On the **Cleanable** tab, select **Clean** for the cleanable security settings you want to remove. The security settings that cannot be removed are shown on the **Non-cleanable** tab.
3. Click **OK** to clean up the selected security settings.

Advanced Encryption Standard

Oracle Data Integrator 12*c* (12.1.3) has strengthened security by using the Advanced Encryption Standard (AES), a Federal Information Processing Standard. The AES algorithm is a block cipher that can be used to encrypt and decrypt digital information. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits. ODI uses AES-128 for password encryption and you can also configure ODI to use AES-256.

AES is a symmetric block encryption algorithm and a secret key or master key is used when AES transforms plaintext to ciphertext and conversely when ciphertext is converted to plaintext. Due to security considerations, the master key must be generated automatically and cannot be hard coded. ODI generates a master key string for the AES algorithm during the creation of a master repository. Different repositories have different master keys and when ciphertext is migrated from one repository to another in ODI, you must provide an export key. The export key is used to encrypt any sensitive data that is contained in the exported object. For more information, see the Export Keys section in *Developing Integration Projects with Oracle Data Integrator*.

> **Note:**
>
> To use AES-256, you must download the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy files from the Oracle Java Download site, accept the download terms and conditions, and then replace the existing policy files with the downloaded policy files in `JRE_HOME/lib/security`.

Using a Password-Protected Wallet for Storing Login Credentials

Prior to logging in to Oracle Data Integrator Studio for the first time, you must create a login name for which you specify your Oracle Data Integrator login credentials. These credentials include your Oracle Data Integrator user name and password, and also the Oracle Data Integrator master repository database user name and password. Oracle Data Integrator can save these credentials in either an encrypted login XML file or a password-protected wallet. By default, they are saved in a password-protected wallet.

The use of password-protected wallets for storing database login credentials is strongly recommended by Oracle. To generate password-protected wallets, Oracle Data Integrator leverages the Oracle Wallet features in Oracle Platform Security Services (OPSS). Password-protected wallets use a strong encryption algorithm to secure the wallet's contents.

Note the following about using the password-protected wallet in Oracle Data Integrator Studio:

- When you create the wallet, you specify the wallet password and the days until the password expires. Oracle Data Integrator then stores the repository login credentials in the wallet.
- When you connect to an Oracle Data Integrator repository, you need only to enter the wallet password. All repository login credentials are then automatically retrieved from the wallet.
- The wallet file is named `ewallet.p12` and is placed in the following directory:

  Windows: `%APPDATA%\odi\oracledi\ewallet`

  UNIX: `$HOME/.odi/oracledi/ewallet`

  > **Note:**
  >
  > Depending on your environment, you might first need to create the root directory for the wallet. For more information, seeCreating the Password-Protected Wallet .

- When the wallet password expires, Oracle Data Integrator Studio automatically prompts you to enter a new one when you attempt to log in to Studio.
- You can change the wallet password and expiration at any time from Studio.

The following sections explain how to create and use the password-protected wallet for storing Oracle Data Integrator login credentials:

Creating the Password-Protected Wallet

When logging in to Studio for the first time, you can create the password-protected wallet using the following steps:

1. Create the root directory on your machine for storing the wallet, if it does not already exist.

   Windows:

   On Windows machines, you must create the root directory path `odi\oracledi` in the location represented by the environment variable `%APPDATA%`. For example:

   ```
   c:\> mkdir %APPDATA%\odi\oracledi
   ```

   UNIX:

   On UNIX machines, you must create the root directory path `.odi/oracledi` in the location represented by the environment variable `$HOME`. For example:

   ```
   prompt> mkdir -p $HOME/.odi/oracledi/
   ```

2. Change to the *ORACLE_HOME*`/odi/studio` directory.
3. Launch Oracle Data Integrator Studio by running the following command:

Windows:

```
odi.exe
```

UNIX:

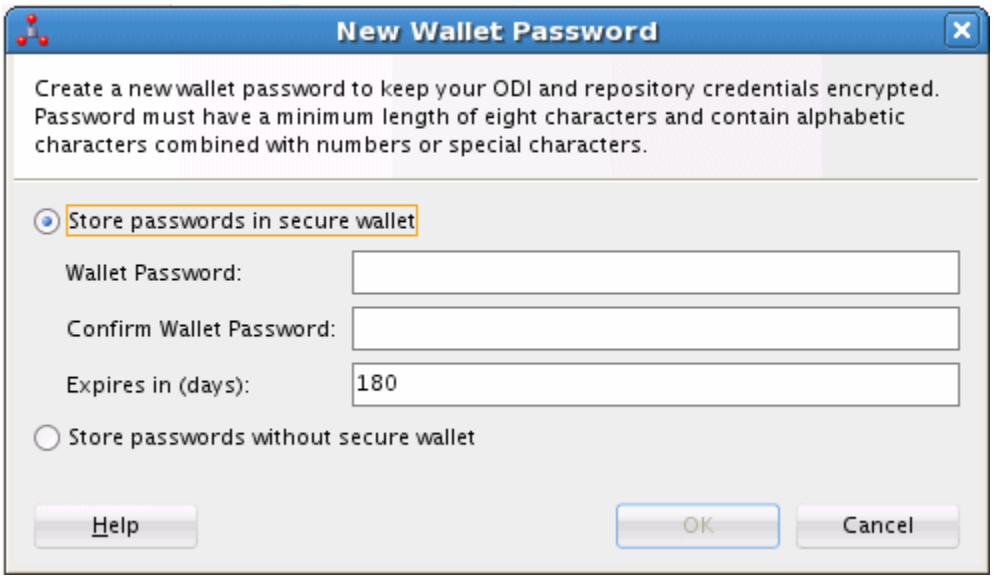
```
./odi.sh
```

The Studio main window is displayed.

4. Click **Connect to Repository...**

   The New Wallet Password dialog box is displayed, shown in Figure 11-1.

Figure 11-1 New Wallet Password Dialog Box



5. Choose the default option, **Store passwords in a secure wallet**, then enter the password in the **Wallet Password** and **Confirm Wallet Password** fields.

   Optionally, you can change the password expiration.

6. Click **OK**.

   You are then prompted to enter login credentials for the master repository. For more information about the required credentials, see Creating the Master Repository.

Subsequently, each time you log in to Studio and connect to a repository, you are prompted for the wallet password.

Permitting Repository Access to Other Users

To give Oracle Data Integrator users access to the repository, you can distribute the password-protected wallet to them in either of the following ways:

- Provide each user with a copy of the `ewallet.p12` file you created, along with the password you specified.
- Create an individual password-protected wallet for each Oracle Data Integrator user. This allows you to set a different password for each wallet.

When creating a wallet for other users, be sure to do the following:

1. Prior to creating a new wallet, move the existing `ewallet.p12` file to a different directory.
2. When creating a wallet for other users, set the expiration to `0` days. This way, when users connect to the repository for the first time, they are prompted to set a new password for their wallets (see Changing the Wallet Password).

All password-protected wallet files for Oracle Data Integrator Studio must be named `ewallet.p12` and must be placed in the `~oracledi/ewallet` directory on each machine from which it is used, as explained in Creating the Password-Protected Wallet.

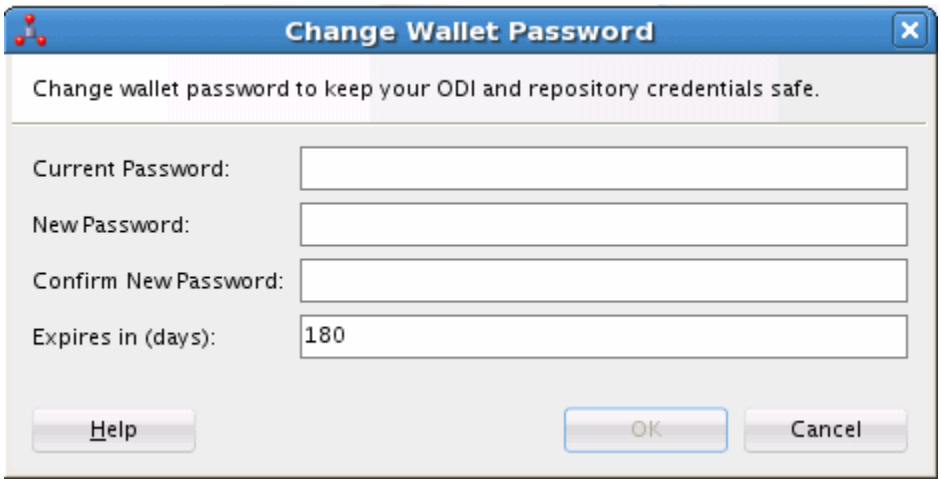
Changing the Wallet Password

When the wallet password expires, you are automatically prompted to enter a new one when you try to connect to a repository. However, you can also change the wallet password from Studio at any time, as follows:

1. Launch Oracle Data Integrator Studio.
2. From the **ODI** menu, choose **Change Wallet Password...**.

   The Change Wallet Password dialog box is displayed, shown in Figure 11-2.

Figure 11-2 Change Wallet Password Dialog Box



3. Enter the current password and the new password, then confirm your new password.

   Optionally, you can also change the expiration.
4. Click **OK**.

Customizing How Login Credentials Are Saved

Oracle Data Integrator Studio provides the following options for storing login credentials:

- You can enable or disable the saving of login credentials.
- If login credentials are being saved, you can choose between saving them in a password-protected wallet or a login XML file.

To change whether or how login credentials are saved, complete the following steps:

1. Launch Oracle Data Integrator Studio.
2. From the **Tools** menu, choose **Preferences...**

   The Preferences dialog box is displayed.
3. In the Preferences dialog box, expand the **ODI** node, then select **User Interface**.

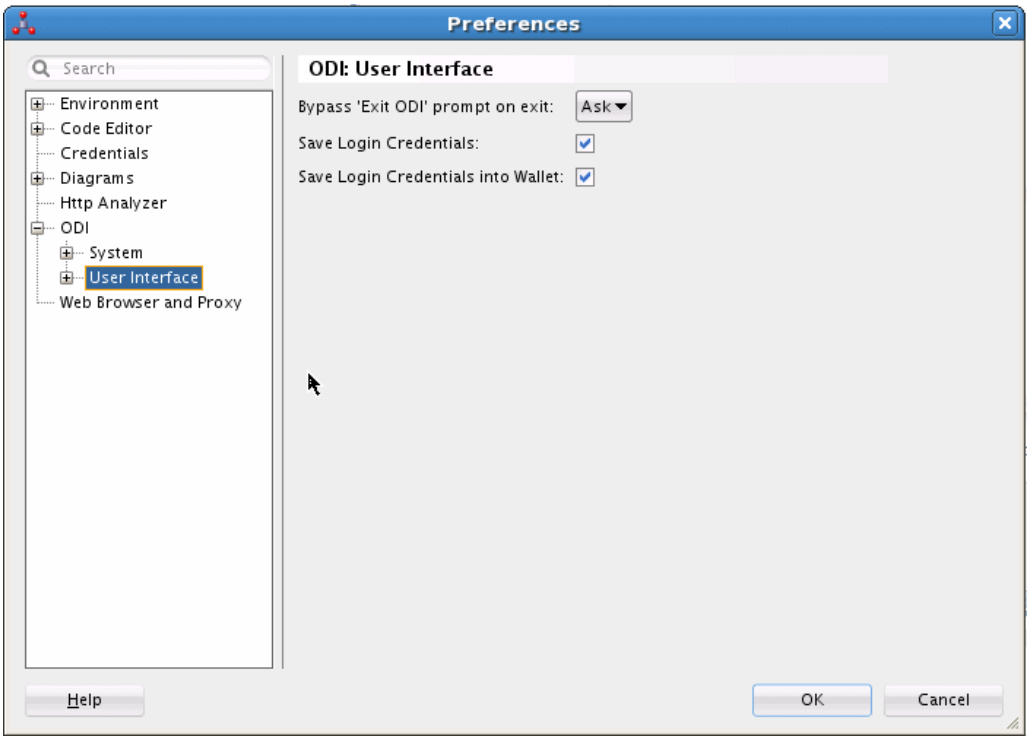
   The Preferences dialog box, shown in Figure 11-3, displays the following options for saving login credentials:

   - **Save Login Credentials** — Select this option to save credentials.
   - **Save Login Credentials into Wallet** — Select this option to store the Oracle Data Integrator login credentials in a password-protected wallet.

     If **Save Login Credentials** is selected, but **Save Login Credentials into Wallet** is deselected, login credentials are saved in a login XML file.

   By default, both options are selected, which is the setting recommended by Oracle.
4. If you change any preferences, click **OK**.

Figure 11-3 Preferences Dialog Box for Saving Login Credentials



Setting Up External Password Storage

Oracle Data Integrator stores by default all security information in the master repository. This password storage option is called **internal password storage**.

Oracle Data Integrator can optionally use OPSS for storing critical security information. When using OPSS with Oracle Data Integrator, the data server passwords and context

passwords are stored in the OPSS Credential Store Framework (CSF). This password storage option is called **external password storage**.

> **Note:**
>
> When using external password storage, other security details such as user names, password, and privileges remain in the master repository. It is possible to externalize the authentication and have users and password stored in an identity store using external authentication. **Note also:** The *external password storage* option is unrelated to the *external authentication* feature. For more information about external authentication, see Configuring External Authentication.

To use the external password storage option:

1. You need to install a WebLogic Server instance configured with OPSS.
2. All Oracle Data Integrator components, including the run-time agent, need to have access to the remote credential store.

See the Configuring Java EE Applications to Use OPSS chapter in *Securing Applications with Oracle Platform Security Services* for more information.

Setting the Password Storage

There are four ways to set or modify the password storage:

- Importing the master repository (see the Importing the Master Repository section in *Developing Integration Projects with Oracle Data Integrator*) allows you to change the password storage.
- Creating the master repository (see Creating the Master Repository) allows you to define the password storage.
- Switching the Password Storage modifies the password storage for an existing master repository.
- Recovering the Password Storage allows you to recover from a credential store crash.

Switching the Password Storage

Switching the password storage of the Oracle Data Integrator repository changes how data servers and contexts passwords are stored. This operation must be performed by a SUPERVISOR user.

Use the **Switch Password Storage** wizard to change the password storage options of the data server passwords.

Before launching the Switch Password Storage wizard perform the following tasks:

- Disconnect Oracle Data Integrator Studio from the repository.
- Shut down every component using the Oracle Data Integrator repository.

To launch the Switch Password Storage wizard:

1. From the **ODI** main menu, select **Password Storage** > **Switch**...

2. Specify the login details of your Oracle Data Integrator master repository as defined when connecting to the master repository (see: Connecting to the Master Repository).
3. Click **Next**.
4. Select the password storage:
   - Select **Internal Password Storage** if you want to store passwords in the Oracle Data Integrator repository.
   - Select **External Password Storage** if you want use OPSS Credential Store Framework (CSF) to store the data server and context passwords.

     If you select external password storage, you must provide the **MBean Server Parameters** to access the credential store described in Table 11-2, and then click **Test Connection** check the connection to the MBean Server.

Table 11-2 MBean Server Parameters

Server	Host	JMX Port	User	Password
From the list, select the application server.	MBeans server host. For example, *machine.example.com*	MBeans Server Port. For example, 7001	MBeans Server user name. For example, *weblogic*	MBean server password

5. Click **Finish**.

The password storage options have been changed. You can now reconnect to the Oracle Data Integrator repository.

Recovering the Password Storage

Oracle Data Integrator offers a password recovery service that should be used only in case of an external password storage crash. Using this procedure, password storage is forced to internal password storage because external storage is no longer available. This operation should be performed by a supervisor user.

> **Caution:**
>
> When performing a password storage recovery, passwords for context, data servers, JDBC password of the work repository and Enterprise Scheduler related passwords are lost and need to be re-entered manually in Topology Navigator.

Use the **Recover Password Storage** wizard to start the password recovery.

To launch the Recover Password Storage wizard:

1. From the **ODI** main menu, select **Password Storage** > **Recover**...
2. Specify the login details of your Oracle Data Integrator master repository defined when connecting to the master repository (see: Connecting to the Master Repository).

3. Click **Finish**.
4. Re-enter manually data server and context passwords. Refer to Setting Up a Topologyfor more information.

Managing the Authentication Mode

By default, Oracle Data Integrator users are authenticated using the identity information contained in the master repository. However, Oracle Data Integrator users can be authenticated using an external authentication service instead: Using Oracle Platform Security Services (OPSS), Oracle Data Integrator users are authenticated against an external enterprise identity store (for example, an LDAP server such as Oracle Internet Directory or Microsoft Active Directory), which contains enterprise users and enterprise roles in a central place.

When external authentication is enabled, the master repository retains the Oracle Data Integrator specific privileges. External users do not need to be created in Oracle Data Integrator's internal repository. Instead, external user credentials rely on a centralized identity store, and authentication always takes place against this external store.

The authentication mode (internal or external) can be defined when you create the repository, and can also be switched for existing repositories, as explained in Setting the Authentication Mode in the Master Repository.

For details about configuring external authentication, see Configuring External Authentication.

Setting the Authentication Mode in the Master Repository

There are two ways to set the authentication mode:

- When you create the master repository. The Master Repository Creation Wizard contains options for setting the authentication mode. For information about how to set external authentication when creating the master repository, see Set External Authentication when Creating the Master Repository. For general information, see also Creating the Master Repository.
- When you switch the authentication mode from Studio. The Switch Authentication Mode Wizard allows you to change the authentication mode used by an existing master repository. For detailed steps to switch to external authentication mode in Studio, using this wizard, see Switching an Existing Master Repository to External Authentication Mode.

> **Note:**
>
> Before you can select external authentication mode in the master repository, you must configure external authentication, as explained in Configuring External Authentication.

Configuring External Authentication

By default, Oracle Data Integrator stores all user information as well as users' privileges in the master repository. When a user logs to Oracle Data Integrator, it authenticates against the master repository. This authentication method is called **internal authentication**.

However, you can customize Oracle Data Integrator to use Oracle Platform Security Services (OPSS) to authenticate users that are defined in an enterprise identity store, which is typically an LDAP server such as Oracle Internet Directory or Microsoft Active Directory. The identity store contains enterprise user information and enterprise roles. Such an identity store is used at the enterprise level by applications to have centralized user definitions and to support single sign-on (SSO). In Oracle Data Integrator, this authentication method is called **external authentication**. Configuring external authentication enables you to leverage the standard Java Security Model authentication and authorization services that are provided by OPSS. When using external authentication, ODI uses the LDAP common name (cn) attribute to match ODI users to LDAP users.

To use external authentication, you need to configure an enterprise identity store and also configure each Oracle Data Integrator component to refer to that identity store.

> **Note:**
>
> When using external authentication, only users and passwords are externalized. Oracle Data Integrator privileges remain within the repository.

The following sections explain how to set up external authentication for the Studio and the Standalone agent:

- Configuring Oracle Data Integrator Studio for External Authentication
- Switching an Existing Master Repository to External Authentication Mode
- Reactivating Users After Switching to External Authentication
- Configuring Standalone or Standalone Colocated Agents for External Authentication
- Mapping Principals Defined in an Identity Store to Oracle Data Integrator Roles

Configuring Oracle Data Integrator Studio for External Authentication

To configure Oracle Data Integrator Studio with external authentication, you must do the following:

1. Set Up the OPSS Configuration File
2. Create a Wallet File for an LDAP Bootstrap User
3. Set External Authentication when Creating the Master Repository

Set Up the OPSS Configuration File

The configuration to connect and use the identity store is contained in the OPSS configuration file, called `jps-config-jse.xml`. To configure Oracle Data Integrator components with external authentication, you must set up this configuration file to correspond to the external identity store that you plan to use.

To set up the OPSS configuration file, complete the following steps:

1. Create the `odi/oracledi` directory on your machine, if it does not already exist. For example:

Windows:

```
c:\> mkdir %APPDATA%\odi\oracledi
```

UNIX:

```
prompt> mkdir -p $HOME/.odi/oracledi/
```

2. Copy the following files into the `odi/oracledi` directory:
   - *ORACLE_HOME*`/oracle_common/modules/oracle.jps/domain_config/jse/jps-config.xml`
   - *ORACLE_HOME*`/oracle_common/modules/oracle.jps/domain_config/jse/system-jazn-data.xml`
3. Change to the `odi/oracledi` directory.
4. Rename `jps-config.xml` to `jps-config-jse.xml`.
5. Open `jps-config-jse.xml` in an editor.
6. Add a service instance for the identity store provider and include the following properties:
   - The service instance name. For example, for Oracle Internet Directory, `name="idstore.oid"`.
   - The `idstore.type` property.
   - The `bootstrap.security.principal.map` property.
   - The `bootstrap.security.principal.key` property.

   For an example service instance configuration, see Example 11-1.

   For details about specifying these properties for your identity store, see the LDAP Identity Store Properties section in *Securing Applications with Oracle Platform Security Services*.
7. In the default jps context, change the `serviceInstanceRef name="idstore.xml"` value to `"idstore.<`*`your-idstore-type`*`>"`, as in the following example that sets the identity store type for Oracle Internet Directory:

   ```
   <serviceInstanceRef ref="idstore.oid"/>
   ```
8. Comment out the `keystore` and `audit` service instance references in the default jps context. For example:

   ```
   <jpsContext name="default">
    <serviceInstanceRef ref="credstore"/>
   <!--  <serviceInstanceRef ref="keystore"/>  -->
   <serviceInstanceRef ref="policystore.xml"/>
   <!--  <serviceInstanceRef ref="audit"/>  -->
   ```
9. Set any additional property attribute values as appropriate for your identity store. For complete details about the OPSS configuration file, see the "Configuring Java SE Applications to Use OPSS" chapter in *Securing Applications with Oracle Platform Security Services* for more information.

Example 11-1 shows the configuration of Oracle Internet Directory in the service instance section of the configuration file.

Example 11-1 Example Service Instance Configuration

```
<!-- OID LDAP Identity Store Service Instance -->
  <serviceInstance name="idstore.oid" provider="idstore.ldap.provider">
  <property name="subscriber.name" value="dc=us,dc=oracle,dc=com" />
```

```
<property name="idstore.type" value="OID" />
 <property name="bootstrap.security.principal.map" value="BOOTSTRAP_JPS"/>
 <property name="bootstrap.security.principal.key" value="bootstrap_idstore"/>
<property name="username.attr" value="uid"/>
<property name="groupname.attr" value="cn"/>
<property name="ldap.url" value="ldap://hostname:port" />
<extendedProperty>
 <name>user.search.bases</name>
 <values>
  <value>cn=users,dc=us,dc=oracle,dc=com</value>
 </values>
</extendedProperty>
<extendedProperty>
 <name>group.search.bases</name>
 <values>
  <value>cn=groups,dc=us,dc=oracle,dc=com</value>
 </values>
</extendedProperty>
</serviceInstance>
 .
 .
 .
<serviceInstanceRef ref="idstore.oid"/>
```

Create a Wallet File for an LDAP Bootstrap User

You must create a wallet file to store the identity store bootstrap user's credentials. This wallet file is referenced from the OPSS configuration file and is used by Oracle Data Integrator to establish the connection to the identity store that is configured in the OPSS configuration file.

> **Note:**
>
> This wallet file is different from the password-protected wallet used for storing login credentials for Oracle Data Integrator Studio.

To create this wallet file, complete the following steps:

1. Change to the *ORACLE_HOME*`/odi/sdk/bin` directory.
2. Run the `odi_credtool` script.

   You are prompted to enter the following parameters:

For the following parameter . . .	. . . enter the following value
`User name`	The Distinguished Name (DN) of the bootstrap user account used to connect to the identity store with necessary privileges. For example, for Microsoft Active Directory, specify this user as follows: `CN=Administrator,CN=Users,DC=ad,DC=vm,DC=`*`mycompany`*`,DC=com`

For the following parameter . . .	. . . enter the following value
`Password`	The password for the bootstrap user account that is used to connect to the identity store.

After you enter the correct credentials for the bootstrap user account, the following message is displayed:

```
The credentials have been successfully added to the boostrap credential store.
```

Set External Authentication when Creating the Master Repository

To create the master repository and set it to use external authentication:

1. Change to the *ORACLE_HOME*`/odi/studio` directory.
2. Launch Oracle Data Integrator Studio by running the following command:

   Windows:

   ```
   odi.exe
   ```

   UNIX:

   ```
   ./odi.sh
   ```

3. In Oracle Data Integrator Studio, select **File** > **New**, select **Master Repository Creation Wizard**, and click **OK**.
4. In the repository selection screen of the Master Repository Creation Wizard, enter the connection parameters for the database that is to contain the master repository, and click **Test Connection** to ensure the parameters are correct. For details about configuring this database, see *Installing and Configuring Oracle Data Integrator*.

   When the connection test is successful, click **OK**.
5. In the authentication screen of the Master Repository Creation Wizard, select **Use External Authentication**.

For information about how to select principals who have supervisor privileges in the master repository, see Mapping Principals Defined in an Identity Store to Oracle Data Integrator Roles.

For information about how to update an existing master repository to use external authentication, see Switching an Existing Master Repository to External Authentication Mode.

Switching an Existing Master Repository to External Authentication Mode

To switch the authentication mode from internal to external in an existing master repository, complete the following steps:

1. Change to the *ORACLE_HOME*`/odi/studio` directory.
2. Launch Oracle Data Integrator Studio by running the following command:

   Windows:

   ```
   odi.exe
   ```

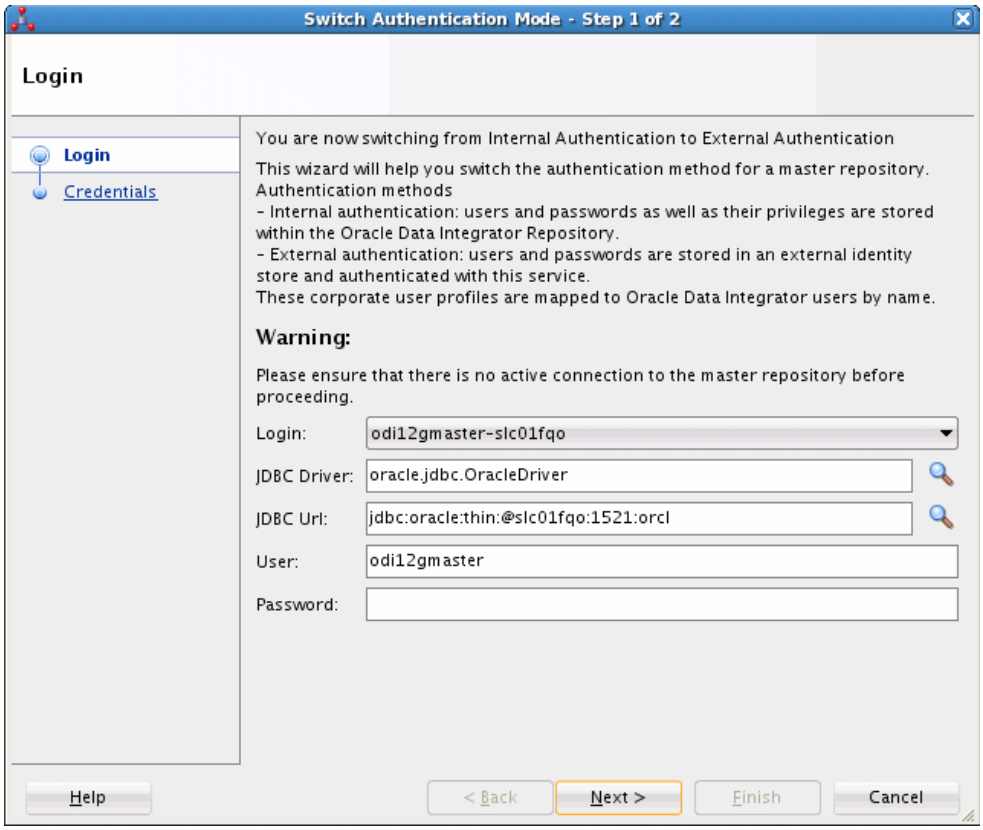
   UNIX:

   ```
   ./odi.sh
   ```

3. In Oracle Data Integrator Studio, select **ODI** > **Switch Authentication Mode...**.

   The Login screen of the Switch Authentication Mode wizard is displayed, as shown in Figure 11-4.

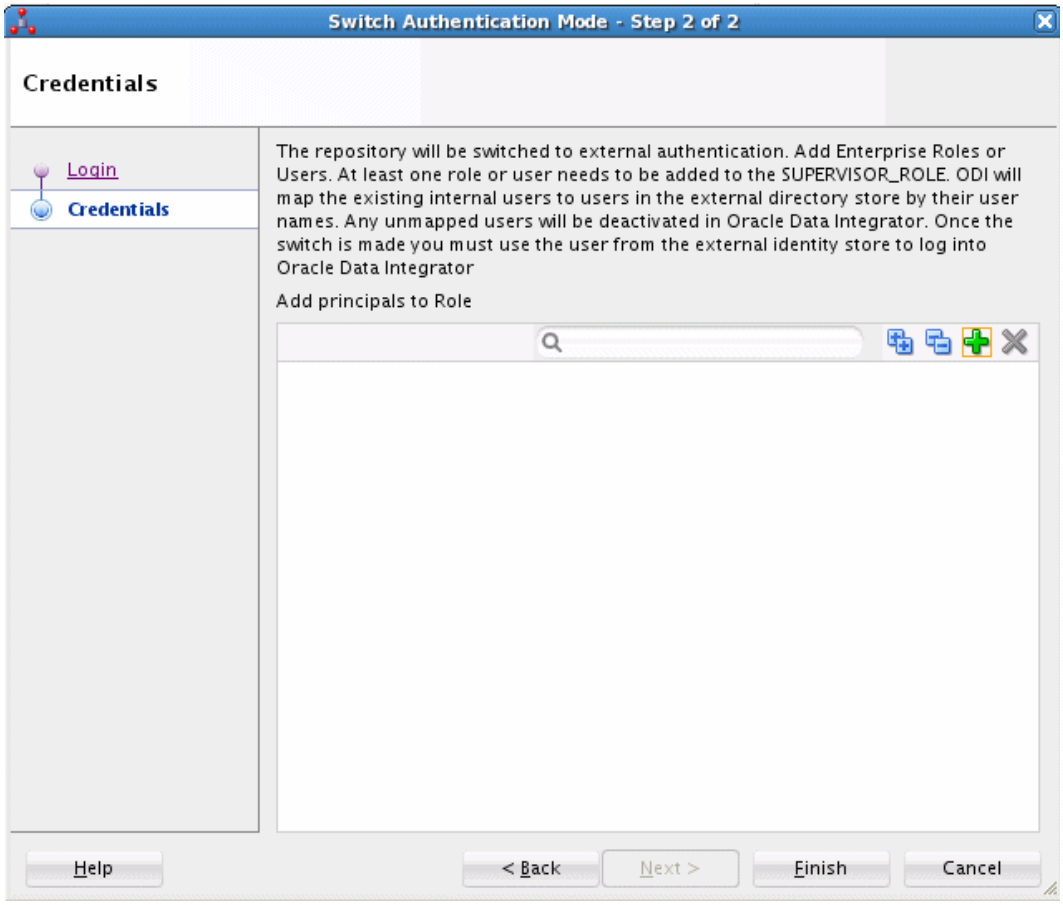
Figure 11-4 Login Screen of Switch Authentication Mode Dialog Box



4. Enter the login name of the master repository, the JDBC connection parameters, and the user name and password for the master repository database user, then click **Next**.

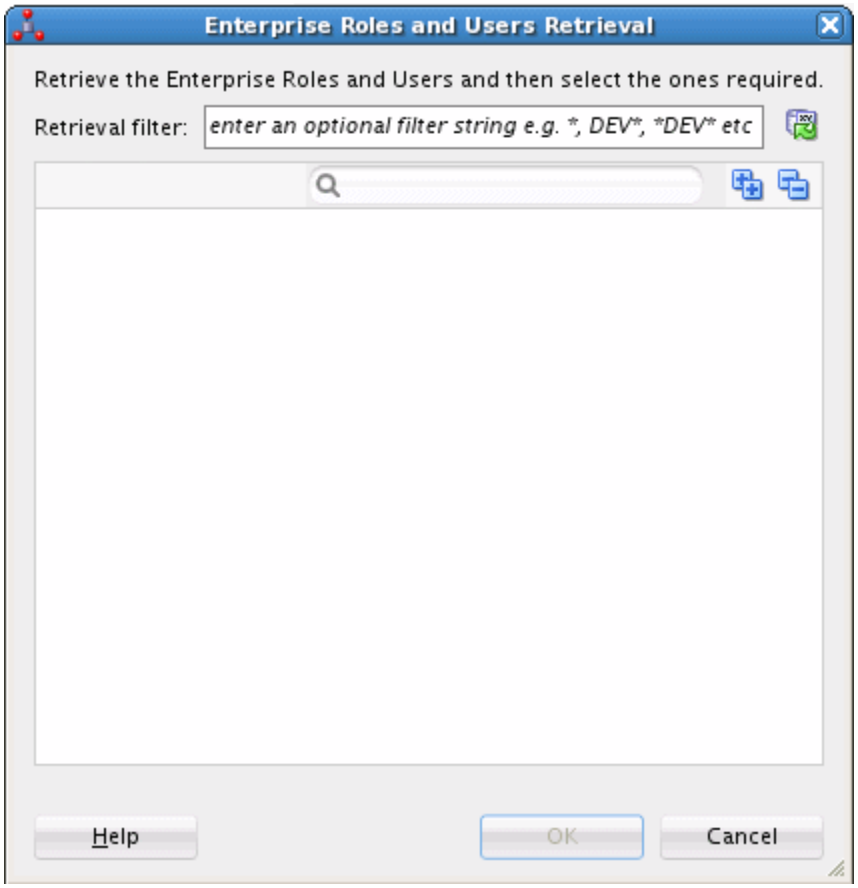
   The Credentials screen of the Switch Authentication Mode wizard is displayed, as shown in Figure 11-5.

Figure 11-5 Credentials Screen of Switch Authentication Mode Wizard



5. Click the **Add** button, which is shown as the green plus symbol adjacent to the search field, to display the Enterprise Roles and Users Retrieval dialog box, shown in Figure 11-6.

Figure 11-6 Enterprise Roles and Users Retrieval Dialog Box



You can use the Enterprise Roles and Users Retrieval dialog box to either:

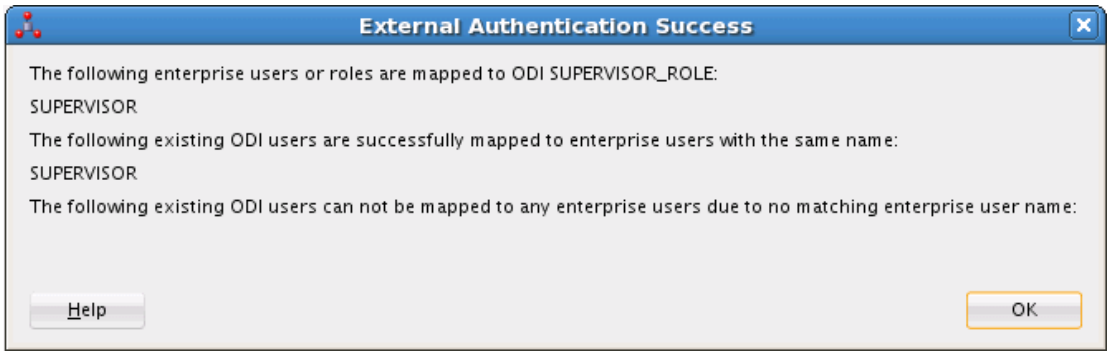
- Enter a filter expression to display the roles and users in the identity store that match the filter.
- Search the identity store for a specific role or user name.

You can expand the Enterprise Roles and Enterprise Users nodes to browse the available enterprise roles and users in the identity store.

6. Select the user or role you wish to assign to the `SUPERVISOR_ROLE` role, then click **Finish**.

If the external authentication mode switch is successful, a dialog box similar to the one in Figure 11-7 is displayed:

Figure 11-7 Successful Switch to External Authentication Mode Message Box

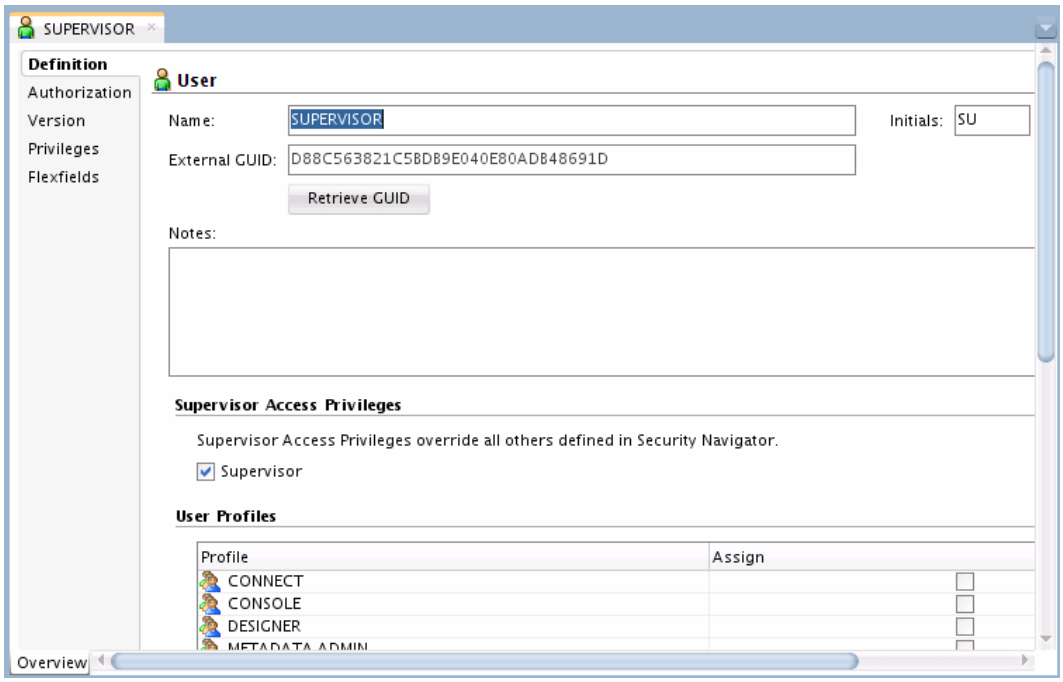


Reactivating Users After Switching to External Authentication

If you have an existing set of Oracle Data Integrator users defined, you can re-enable them after switching to external authentication as explained in this section. To re-enable users, first reconnect to Studio as a user with supervisor privileges—for example, `SUPERVISOR`—using the password specified in the external identity store, and not the one stored in the internal Oracle Data Integrator repository. Then complete the following steps:

1. In the Security Navigator, expand the **Users** navigation tree.
2. From the list of users displayed, select the user that you want to re-enable.
3. Right-click and select **Open**. The User editor appears, shown in Figure 11-8.

Figure 11-8 Open USER Administrator Dialog Box



The **Name** field displays the user you selected in Step 2.

4. Click **Retrieve GUID**. If the user name has a match in the identity store, this external user's GUID appear in the External GUID field.
5. From the **File** main menu, select **Save** and disconnect.
6. Reconnect as this user (for example, the user `SUPERVISOR` shown in Figure 11-8) using the password in the external identity store.

You should now be able to connect to Oracle Data Integrator Studio using the external authentication.

Note the following:

- For troubleshooting LDAP server issues, it is very useful to use any LDAP client to browse the LDAP tree. You can download an LDAP client from the following URL:

  `http://www.ldapbrowser.com/`

- For any LDAP directory other than Microsoft Active Directory, make sure that the property `user.filter.object.classes` is set correctly to the user's object class, which by default is set to `USER` if not specified.
- If Oracle Data Integrator Console and Java EE agent are installed in your environment, and you have switched Oracle Data Integrator Studio to external authentication, you might be unable to log in to Oracle Data Integrator Console and the Java EE agent also might fail. If this occurs, you must also configure Oracle Data Integrator Console and Java EE agent for external authentication using the same external identity store. The procedure for doing this is documented in Note 1510434.1 at My Oracle Support, available at the following URL:

  `https://support.oracle.com/epmos/faces/DocumentDisplay?id=1510434.1`

Configuring Standalone or Standalone Colocated Agents for External Authentication

To configure the Standalone or Standalone Colocated agents for external authentication, complete the following steps:

1. Copy the following files into the *DOMAIN_HOME*`/config/fmwconfig` directory, if they do not already exist there, where *DOMAIN_HOME* represents the root directory of your Oracle Data Integrator domain.
   - *ORACLE_HOME*`/oracle_common/modules/oracle.jps_12.1.2/domain_config/jse/jps-config.xml`
   - *ORACLE_HOME*`/oracle_common/modules/oracle.jps_12.1.2/domain_config/jse/system-jazn-data.xml`

   > **Note:**
   >
   > For information about creating an Oracle Data Integrator domain, see *Installing and Configuring Oracle Data Integrator*.

2. Change to the *DOMAIN_HOME*`/config/fmwconfig` directory.
3. Rename `jps-config.xml` to `jps-config-jse.xml`.
4. Open `jps-config-jse.xml` in an editor.

5. Add a service instance for the identity store provider and include the following properties:
   - The service instance name. For example, for Oracle Internet Directory, `name="idstore.oid"`.
   - The `idstore.type` property.
   - The `bootstrap.security.principal.map` property.
   - The `bootstrap.security.principal.key` property.

   For an example service instance configuration, see Example 11-1.

   For details about specifying these properties for your identity store, see the LDAP Identity Store Properties section in *Securing Applications with Oracle Platform Security Services*.
6. In the default jps context, change the `serviceInstanceRef name="idstore.xml"` value to `"idstore.<your-idstore-type>"`, as in the following example that sets the identity store type for Oracle Internet Directory:

```
 <serviceInstanceRef ref="idstore.oid"/>
```

7. Comment out the `keystore` and `audit` service instance references in the default jps context element. For example:

```
<jpsContext name="default">
 <serviceInstanceRef ref="credstore"/>
<!--  <serviceInstanceRef ref="keystore"/>  -->
<serviceInstanceRef ref="policystore.xml"/>
<!--  <serviceInstanceRef ref="audit"/>  -->
```

8. Save your changes to `jps-config-jse.xml` and exit from your editor.
9. Change to the *`DOMAIN_HOME`*`/bin` directory.
10. Run the `odi_credtool` script.

    You are prompted to enter the following parameters:

For the following parameter . . .	. . . enter the following value
`User name`	This is the Distinguished Name (DN) of the bootstrap user account used to connect to the identity store with Administrator privileges. For example, for Microsoft Active Directory, specify this user as follows: `CN=Administrator,CN=Users,DC=ad,DC=vm,DC=`*`company`*`,DC=com`
`Password`	The password for the bootstrap user account that is used to connect to the identity store.

    After you enter the correct credentials for the bootstrap user account, the following message is displayed:

```
The credentials have been successfully added to the boostrap credential store.
```

Mapping Principals Defined in an Identity Store to Oracle Data Integrator Roles

Oracle Data Integrator's enterprise role integration feature leverages the authorization model in Oracle Platform Security Services (OPSS) to allow you to map enterprise roles to Oracle Data Integrator roles. Enterprise users who belong to the mapped enterprise role can access Oracle Data Integrator by inheriting the privileges that are granted to the mapped Oracle Data Integrator role. Enterprise role integration simplifies both the management of users and also of privileges because you do not need to register users individually in Oracle Data Integrator to grant them access, and you do not need to manage privileges on a per user basis.

The following sections explain how enterprise role integration works and how you can use it to manage the privileges that are assigned to objects, instances, and profiles.

- How Enterprise Role Integration Works
- Defining and Managing Oracle Data Integrator Roles

How Enterprise Role Integration Works

When external authentication is enabled, users are managed in an external identity store, such as Oracle Internet Directory or Microsoft Active Directory.

When using enterprise role integration:

- Any enterprise principal in the identity store that can be mapped to an Oracle Data Integrator role is entitled to the privileges granted to that role. For example, if the enterprise role `DESIGNERS` is mapped to the Oracle Data Integrator role `DESIGNERS_ROLE`, any enterprise user who is a member of `DESIGNERS` in the identity store is given all the privileges granted to `DESIGNERS_ROLE` when authenticated into Oracle Data Integrator.
- You can map multiple enterprise users and roles to an Oracle Data Integrator role, and you can also map an enterprise user or role to multiple Oracle Data Integrator roles. (Note that you cannot map an Oracle Data Integrator role to another Oracle Data Integrator role.).
- When a user is authenticated into Oracle Data Integrator, privileges granted to that user are determined by a union of all the Oracle Data Integrator roles to which the user is mapped directly or indirectly through an enterprise role.

  If the external user is also registered directly in Oracle Data Integrator, as in previous releases, the overall set of privileges given to the user consists of a union of:

  – The Oracle Data Integrator roles to which the user is mapped
  – Any privileges granted directly to the user
- Users do not need to be created and registered in Oracle Data Integrator in order to enable them to be granted access to Oracle Data Integrator.
- Any existing privileges and profiles assigned to individual users, as described in Users, remain in effect. This enables backward compatibility with previous releases of Oracle Data Integrator.

Defining and Managing Oracle Data Integrator Roles

At the time you create the master repository and configure it to use external authentication, the following occurs:

1. The Oracle Data Integrator role `SUPERVISOR_ROLE` is automatically created. This is the main administrative role in Oracle Data Integrator.
2. You are prompted to select one or more enterprise roles, or enterprise users, from the identity store to be mapped to this `SUPERVISOR_ROLE`. Note that Oracle Data Integrator can retrieve all available enterprise roles or enterprise users existing in the identity store that is configured in the `jps-config-jse.xml` file, as explained in Set Up the OPSS Configuration File.

Once you can be authenticated into Oracle Data Integrator and mapped to the `SUPERVISOR_ROLE` role, you can use the Roles Editor, available from Studio, to map additional enterprise principals to this role. Consequently, any user mapped to this role can then use the Roles Editor to create additional Oracle Data Integrator roles, assign privileges and profiles to those roles, and create other principal mappings from the identity store.

The following sections explain how to map enterprise principals to the SUPERVISOR_ROLE role and how to use the Roles Editor to create, update, and remove Oracle Data Integrator roles:

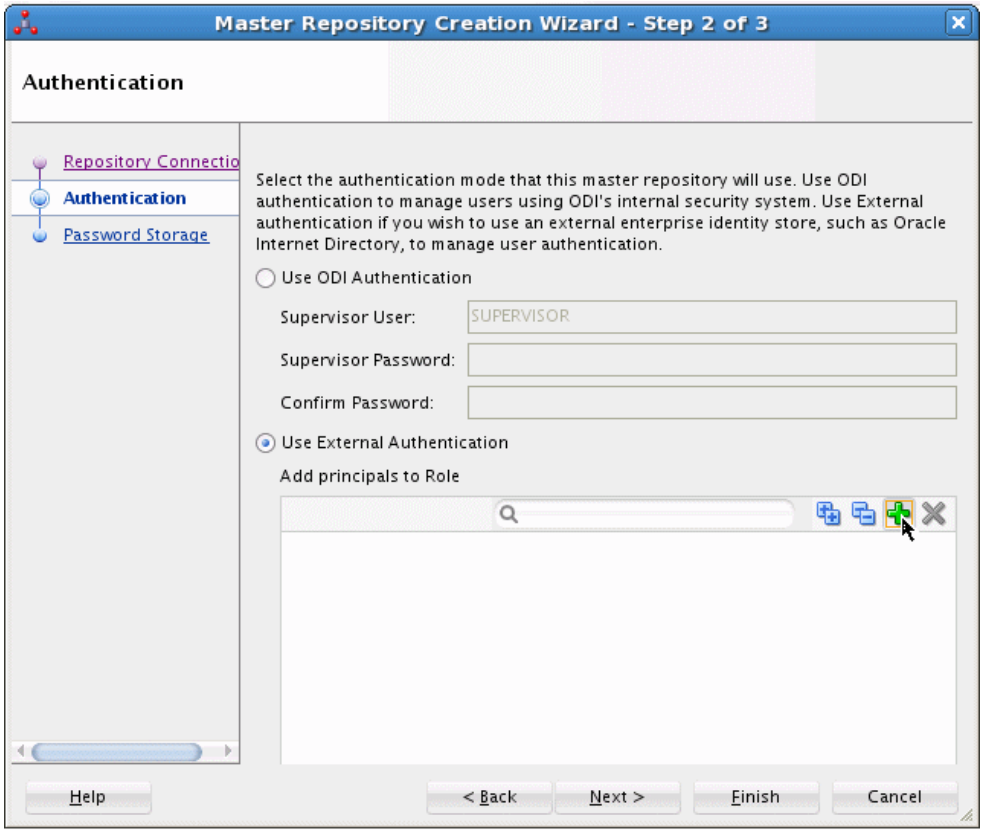
Configuring Enterprise Roles for the Supervisor Role

When you select **Use External Authentication** when creating the master repository, as explained in Set External Authentication when Creating the Master Repository, you need to configure enterprise roles. The initial enterprise role you configure is the principal (one or more) in the identity store to be mapped to the Oracle Data Integrator `SUPERVISOR_ROLE`.

To configure enterprise roles for the `SUPERVISOR_ROLE` in the master repository:

1. Start the Master Repository Creation Wizard, as explained in Set External Authentication when Creating the Master Repository.
2. In the **Master Repository Creation Wizard — Step 2 of 3** page, click the **Add** button retrieve the available enterprise roles and users from the external identity store, as shown in Figure 11-9:

Figure 11-9 Authentication Screen of the Master Repository Creation Wizard

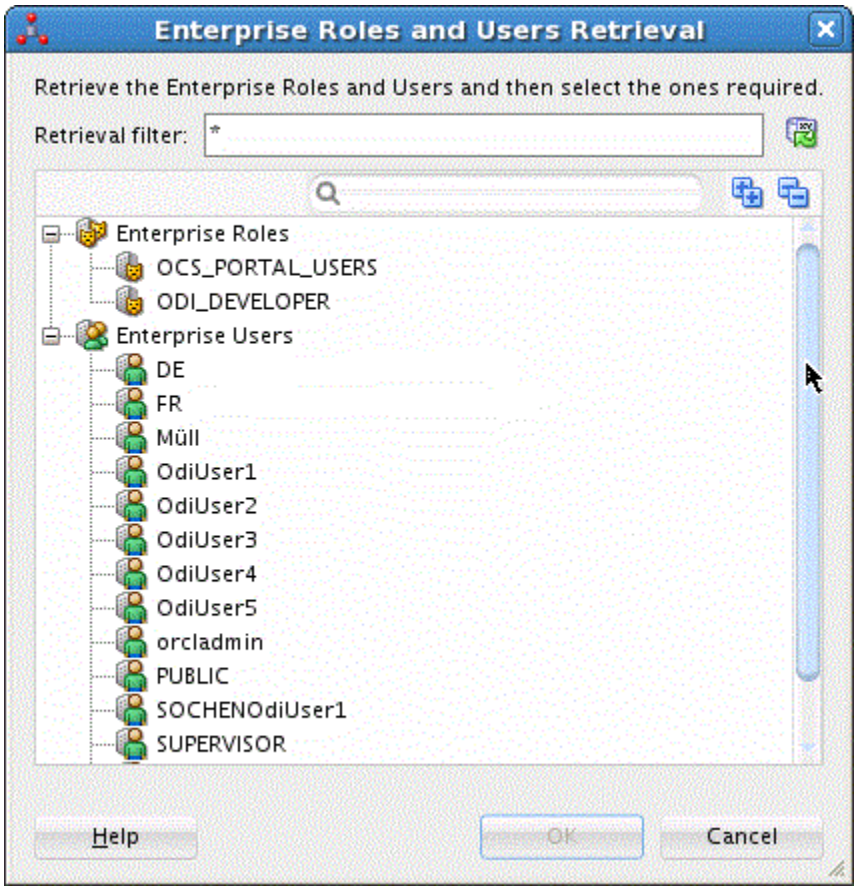


When you click the **Add** button, the Enterprise Roles and Users Retrieval dialog box is displayed, shown in Figure 11-10, in which you can either:

- Enter a filter expression to display the roles and users in the identity store that match the filter.
- Search the identity store for a specific role or user name.

You can expand the **Enterprise Roles** and **Enterprise Users** nodes to display the individual roles and users, respectively, that are defined in the identity store.

Figure 11-10 Enterprise Roles and Users Retrieval Dialog Box



3. Select the enterprise user or role from the Enterprise Roles and Enterprise Users dialog box that you want to assign to the `SUPERVISOR_ROLE` role, and click **OK**.
4. Finish the creation of the master repository, as explained in Creating the Master Repository.

Using the Roles Editor

The following sections explain how to use the Roles Editor to manage enterprise role integration with Oracle Data Integrator:

- Start the Roles Editor
- Create a Role in Oracle Data Integrator
- Map Enterprise Principals to Oracle Data Integrator Roles
- Assign Privileges or Profiles to Oracle Data Integrator Roles
- Delete Oracle Data Integrator Roles

Start the Roles Editor

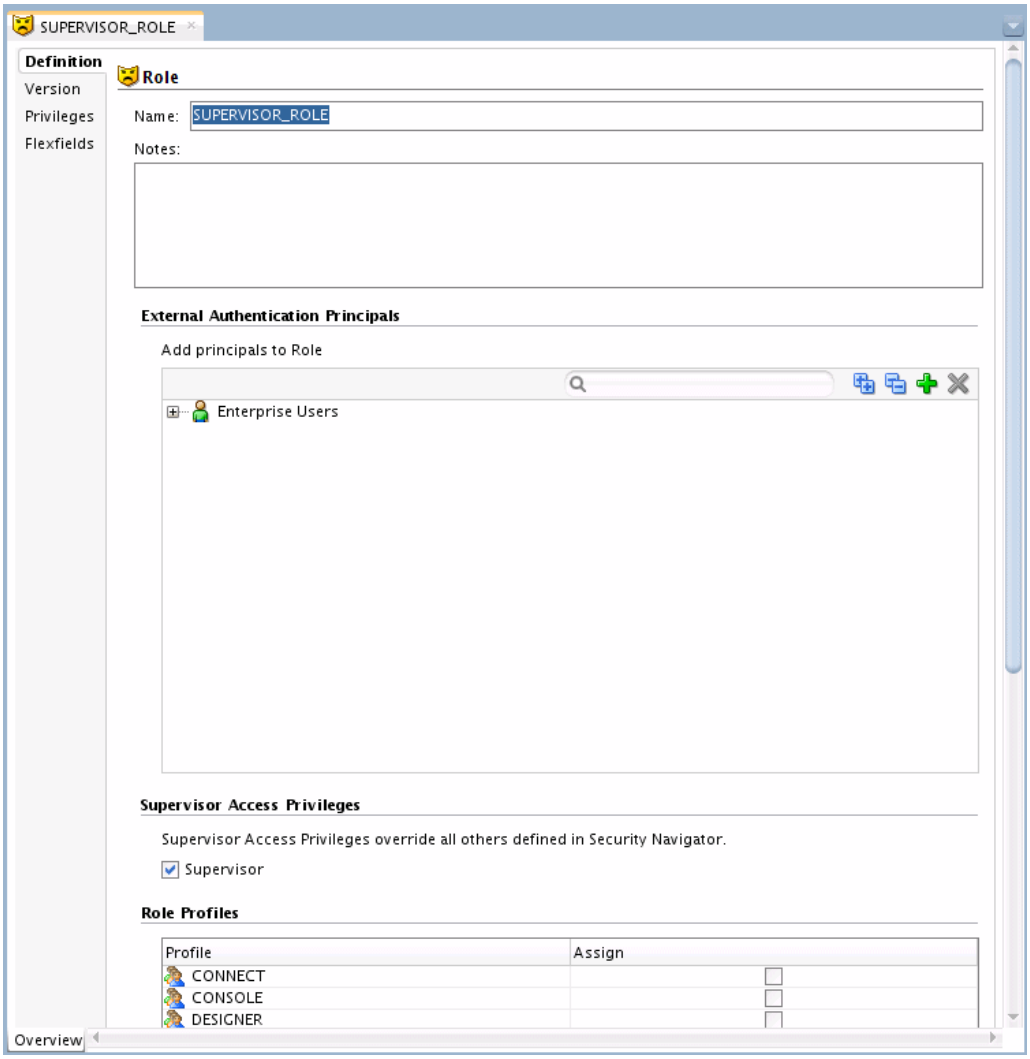
You can use the Roles Editor to create, update, and delete Oracle Data Integrator roles.

To start the Roles Editor:

1. In Oracle Data Integrator Studio, click **Connect to Repository**, if necessary, and enter the required credentials to connect to the repository you want to use.
2. In Studio, choose the **Security Navigator** and select the **Roles** navigation tree.
3. Start the Roles Editor in either of the following ways:
   - Click **New Role** in the toolbar of the Roles navigation tree. This enables you to create a new Oracle Data Integrator role.
   - From the list of roles that is displayed in the Roles navigation tree, right-click a role name and select **Open**. This enables you to update an existing role.
4. Right-click and select **Open**.

The Roles Editor is shown in Figure 11-11.

Figure 11-11 Roles Editor



Create a Role in Oracle Data Integrator

To create an Oracle Data Integrator role:

1. Start the Roles Editor in Studio, if necessary.
2. Choose the **Security Navigator** and select the **Roles** navigation tree.
3. Click **New Role** in the toolbar of the **Roles** navigation tree.
4. In the **Name** field, enter the name of the Oracle Data Integrator role you want to create.
5. From the **File** main menu, select **Save**.

The new role name is displayed in the **Roles** navigation tree.

Map Enterprise Principals to Oracle Data Integrator Roles

To map enterprise principals to an Oracle Data Integrator role:

1. If the Oracle Integrator Role you want to modify is not already open in the Roles Editor, right-click the role name in the **Roles** navigation tree and select **Open**.
2. In the **Add Principals to Role** panel of the Roles Editor, click the **Add** button (plus sign) to display the Enterprise Roles and Users Retrieval dialog box.

   From this dialog box, you can either:

   - Enter a filter expression to display the roles and users in the identity store that match the filter.
   - Search the identity store for a specific role or user name.
3. Do either or both of the following:
   - To select one or more enterprise roles to map to the Oracle Data Integrator role, expand the Enterprise Roles node and select those roles.
   - To select one or more enterprise users to map to the Oracle Data Integrator role, expand the Enterprise Users node and select those users.
4. From the **File** main menu, select **Save**.

Assign Privileges or Profiles to Oracle Data Integrator Roles

Studio supports two ways to assign privileges or profiles to an Oracle Data Integrator role:

- Using the Roles Editor
- Clicking and dragging privilege, profiles, and instance objects onto the role name in the Roles navigation tree

Using the Roles Editor:

1. If the Oracle Data Integrator role to which you want to assign privileges is not already open in the Roles Editor, right-click the role name in the **Roles** navigation tree and select **Open**.
2. If you want to assign supervisor privileges to the role, select **Supervisor** in the **Supervisor Access Privileges** section of the Roles Editor.
3. In the **Role Profiles** panel of the Roles Editor, select each profile you want to assign to the role.

> **Note:**
>
> If you have assigned supervisor privileges to the role, this step is unnecessary.

4. From the **File** main menu, select **Save**.

Clicking and Dragging:

Privileges on any of the following items can be added to a role by clicking it and dragging it onto a role name in the **Roles** navigation tree:

- An object node, or an entry within an object node, listed in the **Objects** navigation tree.
- A profile node, or an entry with a profile node, listed in the **Profiles** navigation tree.
- An object instance listed in an navigation tree in the Designer, Operator, or Topology Navigator. Select the instance and drag it onto the desired role name the same way as you grant an object instance privilege to an Oracle Data Integrator user (see Granting an Authorization by Object Instance).

After updating the privileges for an Oracle Data Integrator role, select **Save** from the **File** main menu.

Delete Oracle Data Integrator Roles

To delete an Oracle Data Integrator role:

1. In the **Roles** navigation tree of the Security Navigator, select the role you want to delete.
2. Right-click and select **Delete**.
3. When prompted to confirm your choice, click **Yes**.

Configuring OWSM Policies for JRF-ODI Asynchronous Web Services with Callback

The *invokeStartScenWithCallback* and *invokeRestartSessWithCallback* web service operations are implemented on a JRF platform and can be secured through internal and external authentication methods. The external authentication methods include the use of Oracle Platform Security Services (OPSS) and by attaching Oracle Web Services Manager (OWSM) policies. OWSM provides a policy framework to manage and secure web services. For information on internal and external authentication, see Configuring External Authentication.

> **Note:**
>
> When a JRF-based web service receives a request, it forwards the same to the agent. The agent requires a single key storing the login and password of the user to connect to the repository. The Oracle Data Integrator user names and passwords required by the agent to connect to the repositories are not stored in ODI Configuration files but are stored in the Application Server Credential store. Therefore, it is mandatory that the credential store entries are added before using internal or external authentication methods. For information on adding credential store entries, see step 2 in the following procedure.

To attach an OWSM policy to an asynchronous web service using Fusion Middleware Control, see the following sections in the *Securing Web Services and Managing Policies with Oracle Web Services Manager*:

1. Attaching Policies to Asynchronous Web Service Callback Clients
2. Setting up the environment for OWSM policies. To set up the environment for using OWSM policies, follow the procedures in the Generating Private Keys and Creating the Java Keystore, Configuring the Oracle WSM Keystore Using Fusion Middleware Control, and Adding Keys and User Credentials to the Credential Store Using Fusion Middleware Control sections.

> **Note:**
>
> OWSM policies can also be attached during run-time using WebLogic Scripting Tool (WLST) commands. For information, see *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

Enforcing Password Policies

The password policies consist of a set of rules applied to user passwords when using internal authentication. This set of rules is applied when the password is defined or modified by the user.

To define the password policy:

1. From the Security Navigator toolbar menu, select **Password Policy...**

   The Password Policy dialog appears. This dialog displays a list of rules.
2. If you want your password to expire automatically, check **Password are valid for (days)**, and set a number of days after which passwords need to be modified.
3. Click **Add a Policy**. The Policy Definition dialog appears. A policy is a set of conditions that are checked on passwords.
4. Set a **Name** and a **Description** for this new policy.
5. In the Rules section, add several conditions on the password text or length. You can define, for example, a minimum length for the passwords.
6. From the **Condition to match** list, select whether you want the password to meet at least one or all conditions.

7. Click **OK**.
8. Add as many policies as necessary, and select **Active** for those of the rules that you want to keep active. Only passwords that meet all the policies are considered as valid for the current policy.
9. Click **OK** to update the password policy.

Configuring Standalone or Standalone Colocated Agents to Use a Secure Transport

Hyper-Text Transfer Protocol Secure (HTTPS) is a secure version of Hyper Text Transfer Protocol (HTTP) and is used to provide an additional level of security. This section describes how to set up Hypertext Transfer Protocol Secure (HTTPS) support in Oracle Data Integrator and contains the following topics:

Creating an SSL Certificate

An SSL certificate helps you to encrypt the data being transmitted over the internet.

To create an SSL certificate:

1. Remove the default SSL certificate `demoidentity` from the keystore, using the following command:

```
keytool -delete -alias demoidentity -keystore <domain>/security/
DemoIdentity.jks
Enter the keyStore password, DemoIdentityKeyStorePassPhrase
```

> **Note:**
>
> The keyStore, key, and trustStore passwords are `DemoIdentityKeyStorePassPhrase`, `DemoIdentityPassPhrase`, and `DemoTrustKeyStorePassPhrase` respectively. However, the keyStore and key passwords must be `DemoIdentityKeyStorePassPhrase`.

2. Substitute the alias name with `demoidentity` for a default certificate or `agent` for a user defined certificate and generate the certificate.

```
Default Certificate <aliasName>: demoidentity
User defined Certificate <aliasName>: agent
```

3. Generate a key for the agent using the Keytool utility. The agent uses its fully qualified hostname to fetch the key from the keystore.

```
keytool -genkey -alias <aliasName> -keyalg RSA -keystore <domain>/security/
DemoIdentity.jks
Enter keystore password:  DemoIdentityKeyStorePassPhrase
What is your first and last name?
  [Unknown]:agenthostname.us.example.com
What is the name of your organizational unit?
  [Unknown]:  ODI
What is the name of your organization?
  [Unknown]:  company name
What is the name of your City or Locality?
  [Unknown]:  San Jose
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=agent.us.example.com, OU=ODI, O=company name, L=San Jose, ST=CA, C=US
correct?
  [no]:  yes
Enter key password for <agent>
        (RETURN if same as keystore password) : <return>
```

   A key is created with an alias name <*aliasName*>.

4. Export the key using the following keytool command syntax:

```
keytool -export -alias <aliasName> -keystore <domain>/security/DemoIdentity.jks
-rfc -file public.cert
Enter keystore password:  DemoIdentityKeyStorePassPhrase
   Certificate stored in file <public.cert>
```

5. Import the key into a client truststore using the following keytool command syntax:

```
keytool -import -alias <aliasName> -file public.cert -storetype JKS -keystore
   <MW_HOME>/Oracle_Home/wlserver/server/lib/DemoTrust.jks
Enter keystore password: DemoTrustKeyStorePassPhrase


...
…
Trust this certificate? [no]:  yes
```

   The self-signed certificate is created and added to the client truststore.

Configuring SSL for Standalone Agents or Standalone Colocated Agents

To configure SSL for Standalone or Standalone Colocated agents, the agent must be created with the HTTPS protocol.

To create a Standalone or Standalone Colocated agent with HTTPS protocol:

1. Create the Oracle Data Integrator domain and configure the agent template:
   - If using the ODI Standalone install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Agent** template to create an ODI Instance configuration.
   - If using the ODI Enterprise install type, create an ODI domain and configure the **Oracle Data Integrator - Standalone Colocated Agent** template to create an ODI Instance configuration.

See *Installing and Configuring Oracle Data Integrator* for more information.

2. Create an agent in the master repository with the HTTPS protocol. If the agent already exists, modify the agent's protocol to *https* in the master repository. See Creating a Physical Agentfor more information.
3. Edit the `instance.properties` file at the following location and set `PROTOCOL` to `https`:

   ```
   <domain>/config/fmwconfig/components/ODI/<agentName>
   ```

   Set the SSL password parameters in an encrypted format, as shown below:

   ```
   ODI_KEYSTORE_ENCODED_PASS=<Encrypted password>
   ODI_KEY_ENCODED_PASS=<Encrypted password>
   ODI_TRUST_STORE_ENCODED_PASS=<Encrypted password>
   ```

   For information on how to encrypt a password, see Encoding a Password.
4. Configure other `Keystore Location` and `Truststore` SSL parameters' location and type in the `instance.sh` file, as shown below:

   ```
   ODI_INSTANCE_JAVA_OPTIONS="
   -Djavax.net.ssl.keyStore=<domain>/security/DemoIdentity.jks
   -Djavax.net.ssl.keyStoreType=JKS -Djavax.net.ssl.trustStore= <MW_HOME>/
   Oracle_Home/wlserver/server/lib/DemoTrust.jks
   -Djavax.net.ssl.trustStoreType=JKS $ODI_ADDITIONAL_JAVA_OPTIONS"
   ```
5. Enter the following command to start the agent:

   ```
   <domain>/bin/agent.sh -INSTANCE=agentName
   ```

Configuring ODI Studio to Connect with an SSL-enabled Agent

SSL parameter values are populated in ODI Studio. This is the default and only credential set that is used by ODI Studio when Studio Internal Agent connects to an SSL server.

To connect to an SSL-enabled agent from Oracle Data Integrator Studio:

1. Go to **Tools > Preferences > Credentials**.
2. Edit the **Client Trusted Certificate Keystore** and **Client Trusted Keystore Password** fields.

To point to another trustStore, edit the fields mentioned above. For more information, see the Using SSL With HTTP Analyzer section in *Developing Integration Projects with Oracle Data Integrator*.

Configuring Oracle Data Integrator Client Tools to Communicate with an SSL-enabled Agent

ODI client tools do not have access to the master repository, and therefore cannot decrypt the SSL passwords stored in the `instance.properties` file.

ODI truststore and keystore are configured via instance.sh or instance.cmd, corresponding to the standalone/collocated server.

The passwords are to be given either in plain text or as ODI-encoded passwords.

For tool invocations to succeed, truststore configuration is needed.

This includes importing the necessary certificates into the truststore, and configuring the truststore password.

instance.sh/cmd may already have code that points to a specific truststore.

In that case, work with the specified truststore.

If there is not truststore configured, the JVM truststore will be used.

To run Oracle Data Integrator client tools that connect to an SSL-enabled agent, set a plain text/ODI-encoded password for the `trustStorePassword` system property. The `trustStorePassword` system property must be appended to the `ODI_INSTANCE_JAVA_OPTIONS` variable in the following location:

```
<domain>/config/fmwconfig/components/ODI/OracleDIAgent1/bin/instance.sh
```

Append the parameter using the following command:

```
ODI_INSTANCE_JAVA_OPTIONS="-Djavax.net.ssl.trustStorePassword=<plain_text-password>
-Djavax.net.ssl.trustStore=<domain>/security/DemoIdentity.jks   -
Djavax.net.ssl.trustStoreType=JKS
-Djavax.net.ssl.trustStore=<Domain>/oracle/work/middleware/wlserver/server/lib/
DemoTrust.jks
-Djavax.net.ssl.trustStoreType=JKS $ODI_ADDITIONAL_JAVA_OPTIONS"
```

Sample ODI client tools that require the `trustStorePassword` system property are listed below:

```
./agentstop.sh -NAME= OracleDIAgent1
./startcmd.sh -INSTANCE=OracleDIAgent1 OdiPingAgent -AGENT_NAME=OracleDIAgent1
./startcmd.sh -INSTANCE=OracleDIAgent1 OdiKillAgent -NAME=OracleDIAgent1 -
IMMEDIATE=YES -MAX_WAIT=0
./startscen.sh -INSTANCE=OracleDIAgent1 OdiStartScen -SCEN_NAME=SAM -SCEN_VERSION=001 -
CONTEXT=GLOBAL -SYNC_MODE=1
./startscen.sh -INSTANCE=OracleDIAgent1 SAM 001 GLOBAL -AGENT_URL=https://
<hostname>:<port>/oraclediagent
```

For information on ODI client tools, see *Oracle Data Integrator Tools Reference*.

Encoding a Password

To encode a cleartext password:

1. Execute the following script with `agentName`:

   ```
   <domain>/bin/encode.sh -INSTANCE=<agentName>
   ```

2. Enter the password in the prompt, as shown below:

   ```
   Enter password to encode: <password>
   ```

   The encrypted password is displayed.

Disabling Weak Cipher Suites for an SSL-enabled Standalone Agent or Standalone Colocated Agent

This section describes how to disable weak cipher suites for an SSL-enabled Standalone agent or Standalone Colocated agent. To view the cipher suites in the environment, start the Standalone or Standalone Colocated agent and a list of available suites is displayed.

To disable weak cipher suites:

1. Set the cipher suites that must be excluded in the `instance.properties` file. For example:

   ```
   ODI_EXCLUDED_CIPHERS=TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5,TLS_KRB5_EXPORT_WITH
   _DES_CBC_40_SHA
   ```

2. Invoke the agent in SSL mode, using the following command:

   ```
   ./agent.sh-NAME=OracleDIAgent -PORT=20910
   ```

   A list of available cipher suites is displayed and a list of cipher suites that can be used by the SSL-enabled Standalone agent or Standalone Colocated agent is also displayed.

Configuring Single Sign-On (SSO) for Oracle Data Integrator Console and Enterprise Manager using Oracle Access Manager

This section describes how to optionally configure Single Sign-On for Oracle Data Integrator Console and Enterprise Manager with Oracle Access Manager (OAM) 11g and 12*c*.

To configure Single Sign-On for Oracle Data Integrator Console and Enterprise Manager:

1. Log in to the OAM Console using your browser:

   ```
   http://host:port/oamconsole
   ```

2. Go to **Policy Configuration** > **Application Domains**.

   The Policy Manager pane displays.

3. Locate the application domain you created using the name while registering the WebGate agent.

4. Expand the Resources node and click **Create**.

   The Resource page displays.

5. Add the resources that must be secured. For each resource:

   a. Select `http` as the **Resource Type**.

   b. Select the **Host Identifier** created while registering the WebGate agent.

   c. Enter the **Resource URL** as follows:

Resource URL	ODI Component
/odiconsole*	ODI Console
/odiconsole/.../*	ODI Console
/em*	Enterprise Manager
/em/.../*	Enterprise Manager

d. Enter a **Description** for the resource and click **Apply**.

6. Go to **Authentication Policies** > **Protected Resource Policy** and add the newly created resources.
7. Do the same under **Authorization Policies** > **Protected Resource Policy**.
8. In your WebTier, modify the `mod_wl_ohs.conf` file (in `WT_ORACLE_HOME/instances/<your_instance>/config/OHS/ohs1/`) to include the ODI Console and Enterprise Manager, by adding two additional Location entries using the actual host ID for the WebCenter Portal Administration Server for WebLogicHost.

```
<Location /odiconsole*>
      SetHandler weblogic-handler
      WebLogicHost webcenter.example.com
      WebLogicPort 7001
</Location>
<Location /em*>
      SetHandler weblogic-handler
      WebLogicHost webcenter.example.com
      WebLogicPort 7001
</Location>
```

9. Restart the Oracle HTTP Server for your changes to take effect.

   You should now be able to access the ODI Console and Enterprise Manager with the following links:

   `http://host:OHS port/odiconsole`

   `http://host:OHS port/em`

   and be prompted with the OAM SSO login form.

Configuring Oracle Data Integrator with a Web Proxy Server

This section describes how to reach external http connections if your network setup has a web proxy configuration.

To configure Oracle Data Integrator with web proxy setup from Oracle Data Integrator Studio, go to **Tools** > **Preferences** and set up the proxy in the Web Browser and Proxy page.

You can set the web proxy settings from a command line if you have installed the Oracle Data Integrator Standalone or the Standalone Colocated agent. For the Java EE agent, the web proxy settings must be set in the `setDomainEnv.sh` file in the WebLogic Server. For information on the standard system properties used to alter the mechanisms and behavior of the various classes of the java.net package, see *Networking Properties*.

> **Note:**
>
> The standard Java proxy password property, `http.proxyPassword` only accepts a cleartext password, so Oracle Data Integrator has introduced an additional property, `oracle.odi.http.encoded.proxyPassword` to set an encrypted proxy server password instead of a cleartext password, if required.
>
> The encrypted proxy server password can be set in the following manner:
>
> - Standalone or Standalone Colocated agent: Append the `oracle.odi.http.encoded.proxyPassword` property using the `-D` option of the Java command to the `ODI_INSTANCE_JAVA_OPTIONS` variable in the `instance.sh` file, which is available in the following location:
>
>   `<DOMAIN_HOME>/config/fmwconfig/components/ODI/<INSTANCE_NAME>/bin`
>
> - Java EE agent: Edit the `setDomainEnv.sh` file in the WebLogic Server. The `setDomainEnv.sh` file is available in the following location:
>
>   `<DOMAIN_HOME>/bin`

Best Security Practices for Oracle Data Integrator

Table 11-3 provides a list of best practices for securing the Oracle Data Integrator environment.

Table 11-3 Best Security Practices in an Oracle Data Integrator Environment

Security Action	Description
Secure Oracle Data Integrator Studio	To provide optimal security for Oracle Data Integrator Studio, configure the following: • Password-protected wallet file — Create a password-protected wallet for storing the Oracle Data Integrator Studio login credentials. For more information, see Using a Password-Protected Wallet for Storing Login Credentials. • External authentication — Use external authentication mode instead of the default internal authentication mode. You can select external authentication mode as one of the options when creating the master repository. For more information, see Configuring External Authentication. • External password storage — For data server connection passwords, Oracle Data Integrator provides two options: internal password storage, and external password storage. Oracle recommends external password storage for secure production environments. You can choose the external password storage option at the time you create the master repository. For more information, see Setting Up External Password Storage.
Configure a Standalone agent or Standalone Colocated agent to use a secure transport	For information on how to configure a Standalone agent or Standalone Colocated agent to use a secure transport, see Configuring Standalone or Standalone Colocated Agents to Use a Secure Transport.

Table 11-3 (Cont.) Best Security Practices in an Oracle Data Integrator Environment

Security Action	Description
Configure the Oracle Data Integrator Console to use a secure transport	Oracle recommends the use of a secure transport for access to the Oracle Data Integrator Console. For example, you can configure the Oracle Data Integrator Console to use port 7002, which is the default SSL listen port. Users would then access the Oracle Data Integrator Console using the following URL: `https://<host>:7002/odiconsole` You can use the WebLogic Server Administration Console to secure the transport used for the Oracle Data Integrator Console. For information, see the Configuring SSL part in *Administering Security for Oracle WebLogic Server*.

Index