Oracle® FMW

Oracle Data Integrator Quick Reference Guide

G10243-01
December 2024

ORACLE

Oracle FMW Oracle Data Integrator Quick Reference Guide,

G10243-01

Primary Author: Oracle Corporation

Contents

1

Introduction to Oracle Data Integrator

Learn what you can do with Oracle Data Integrator and understand the important concepts before you get started.

Topics

1.1 About Oracle Data Integrator

A brief overview of Oracle Data Integrator.

Oracle Data Integrator (ODI) provides a fully unified solution for building, deploying, and managing complex data warehouses or as part of data- centric architectures in a SOA or business intelligence environment. In addition, it combines all the elements of data integration - data movement, data synchronization, data quality, data management, and data services - to ensure that information is timely, accurate, and consistent across complex systems.

Traditional ETL (Extract, Transform, Load) tools operate by, first, extracting the data from various sources, then, transforming the data in a proprietary, middle-tier ETL engine that is used as the staging area, and finally loading the transformed data into the target data warehouse, integration server, or Hadoop cluster. Hence the term ETL represents both the names and the order of the operations performed. ODI extracts the data from the various sources, and then loads it to the target where it's transformed. This is known as E-LT: Extract, Load, Transform.

ODI features an active integration platform that includes all styles of data integration: data-based, event-based and service-based. ODI transforms 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 - ODI meets the performance, flexibility, productivity, modularity and hot-pluggability requirements of an integration platform.

See Overview of Oracle Data Integrator in *Understanding Oracle Data Integrator*.

1.2 Oracle Data Integrator Repositories

An overview of Oracle Data Integrator Repositories.

The Oracle Data Integrator (ODI) Repository is composed of a master repository and at least one work repository. Objects developed or configured through the users are stored in one of these repository types. It's installed on an OLTP relational database. There's usually only one master repository that stores the Security information, Topology information, and versioned and archived objects. The work repository is the one that contains actual developed objects. Several work repositories may coexist in the same ODI installation (for example, to have separate environments or to match a particular versioning life cycle). A work repository stores information for Models, Projects, and Scenario execution. When the work repository contains only the execution information (typically for production purposes), it's called an execution repository.

Create Repositories and Connections for Oracle Data Integrator

1.3 Oracle Data Integrator Navigators

An overview of Oracle Data Integrator Navigators.

ODI Studio provides four Navigators for managing the different aspects and steps of an ODI integration project, Topology Navigator, Designer Navigator, Operator Navigator, and Security Navigator.

The Topology Navigator is used to manage the data describing the information system's physical and logical architecture. Through it you can manage the topology of your information system, the technologies and their data types, the data servers linked to these technologies and the schemas they contain, the contexts, the language and the agents, as well as the repositories. The site, machine, and data server descriptions enable Oracle Data Integrator to execute the same mappings in different environments.

The Designer Navigator is used to design data integrity checks and to build transformations, such as automatic reverse-engineering of existing applications or databases, graphical development and maintenance of transformations and mappings, visualization of data flows in the mappings, automatic documentation generation, and customization of the generated code. The main objects handled through it are Models and Projects.

The Operator Navigator is the production management and monitoring tool. It's designed for IT production operators. Through it you can manage your executions in the sessions, as well as the scenarios in production.

The Security Navigator is the tool for managing the security information in ODI. Through it you can create users and profiles and assign user rights for methods on generic objects, and fine-tune these rights on the object instances.

The Oracle Data Integrator User Interface

1.4 Oracle Data Integrator Run-Time Agent

An overview of Oracle Data Integrator's Run-Time Agent.

At design time, developers generate scenarios from the business rules that they have designed. The code of these scenarios is then retrieved from the repository by the Run-Time Agent. This agent then connects to the data servers and orchestrates the code execution on these servers. It retrieves the return codes and messages for the execution, as well as additional logging information - such as the number of processed records, and execution time - in the repository.

The Agent comes in three different flavors, standalone Agent, standalone Colocated Agent, and Java EE Agent. The standalone agent is more appropriate when you need to use a resource that is local to one of your data servers, and you do not want to install a Java EE application server on this machine. Standalone colocated agents can be installed on any server with a Java Virtual Machine installed, but require a connection to the WebLogic Administration Server. This type of agent is more appropriate when you need to use a resource that is local to one of your data servers but you want to centralize management of all applications in an enterprise application server. A Java EE agent is deployed as a web application in a Java EE application server, and can benefit from all the features of the application server. This type of agent is more appropriate when there is a need for centralizing the deployment and management of all applications in an enterprise application server, or when you have requirements for high availability.

Oracle Data Integrator Agents

1.5 Oracle Data Integrator Data Servers

Learn about Data Servers in Oracle Data Integrator.

The physical components that store and expose structured data in Oracle Data Integrator (ODI) are defined as data servers. Each data server is always linked to a single technology. It stores information according to a specific technical logic, which is declared in the physical schemas attached to it. Every database server, JMS message file, group of flat files, or other form of data source or target, that is used in ODI, must be declared as a data server. Similarly, every schema, database, JMS Topic, and so forth used in ODI, must be declared as a physical schema.

Data Servers

1.6 Oracle Data Integrator Models

An overview of Oracle Data Integrator models.

A model is the description of a set of datastores. It corresponds to a group of tabular data structures stored in a data server. A model is based on a Logical Schema defined in the topology. In a given Context, this Logical Schema is mapped to a physical schema. The data schema of this physical schema contains physical data structure: tables, files, JMS messages, elements from an XML file, that are represented as datastores. Models, as well as all their components, are based on the relational paradigm (table, attributes, keys, and so on). Models in Oracle Data Integrator only contain metadata - that is, the description of the data structures. They do not contain a copy of the actual data.

To automatically populate datastores into the model, you reverse-engineer the model. A standard reverse-engineering uses the capacities of the JDBC driver used to connect the data server to retrieve the model metadata. A customized reverse-engineering uses a reverse-engineering Knowledge Module (RKM), to retrieve metadata for a specific type of technology and create the corresponding datastore definition in the data model.

Models

1.7 Oracle Data Integrator Datastores

An overview of Oracle Data Integrator Datastores.

A datastore is a data structure that can be used as a source or a target in a mapping. It can be a table stored in a relational database, a Hive table in a Hadoop cluster, an ASCII or EBCDIC file (delimited, or fixed length), a node from a XML file, a JMS topic or queue from a Message Oriented Middleware, a node from a enterprise directory, or an API that returns data in the form of an array of records. Regardless of the underlying technology, all data sources appear in Oracle Data Integrator in the form of datastores that can be manipulated and integrated in the same way. The datastores are grouped into data models.

Data Stores

1.8 Oracle Data Integrator Data Flow

An introduction to data flow in Oracle Data Integrator.

Business rules defined in the mapping are automatically converted into a data flow that carries out the joins, filters, mappings, and constraints from source data to target tables.

By default, Oracle Data Integrator (ODI) uses the Target RDBMS as a staging area for loading source data into temporary tables and applying all the required mappings, staging filters, joins and constraints. The staging area is a separate area in the RDBMS (a user/database) where ODI creates its temporary objects and executes some of the rules. When performing the operations this way, ODI uses an E-LT strategy as it first extracts and loads the temporary tables and then finishes the transformations in the target RDBMS.

In cases when source volumes are small (less than 500,000 records), the staging area can be located in ODI's In-Memory Engine - its own in-memory relational database. ODI then behaves like a traditional ETL tool.

Data Flow

1.9 Oracle Data Integrator Integration Projects

An introduction to integration projects in Oracle Data Integrator.

An integration project is composed of several types of components. These components include organizational objects, such as folders, procedures, and packages, and development objects such as mappings, sequences, and variables. A project's components can be defined in the project and/or global components referenced by the project. Also, a project can use components defined in the models and the topology.

The package is a large unit of execution in ODI. A package is made up of a sequence of steps organized into an execution diagram. Each step can either succeed or fail its execution. Depending on the execution result (success or failure), a step can branch to another step.

When a package, mapping, procedure, or variable component has been fully developed, it's compiled in a scenario. A scenario is the execution unit for production environments. Scenarios can be scheduled for automated execution.

Projects

Packages

Procedures

1.10 Oracle Data Integrator Mappings

An overview of Oracle Data Integrator Mappings.

A mapping connects sources to targets through a flow of components such as Join, Filter, Aggregate, Set, Split, and so on. A mapping references the Knowledge Modules that will be used to generate the integration process. A mapping is made up of the source and target datastores. Optionally, you can use datasets within mappings as data sources. A dataset is a logical container organizing datastores by an entity relationship declared as joins and filters, rather than the flow mechanism used elsewhere in mappings.

Reusable mappings are modular, encapsulated flows of components which you can save and re-use. You can place a reusable mapping inside another mapping, or another reusable mapping (reusable mappings can be nested). A reusable mapping can also include datastores as sources and targets, like other mapping components.

Mappings

1.11 Oracle Data Integrator Knowledge Modules

An overview of Oracle Data Integrator Knowledge Modules.

Knowledge Modules (KM) implement "how" the integration processes occur. There are six Knowledge Module types, each referring to a specific integration task:

- Reverse-engineering metadata from the heterogeneous systems for Oracle Data Integrator (RKM). See Creating and Using Data Models and Datastores in *Developing Integration Projects with Oracle Data Integrator*.
- Journalizing Knowledge Modules handling Changed Data Capture (CDC) on a given system (JKM). See Using Journalizing in *Developing Integration Projects with Oracle Data Integrator*.
- Integrating Knowledge Modules (IKM) are used in mappings, to integrate data in a target system, using specific strategies (insert/update, slowly changing dimensions). See Creating and Using Mappings in *Developing Integration Projects with Oracle Data Integrator*.
- Controlling Knowledge Modules, control data integrity on the data flow (CKM) in a data model's static check and mappings flow checks. See Creating and Using Data Models and Datastores in *Developing Integration Projects with Oracle Data Integrator*.
- Loading Knowledge Modules (LKM) load data from one system to another, using system-optimized methods. These KMs are used in mappings. See Creating and Using Mappings in *Developing Integration Projects with Oracle Data Integrator*.
- Service Knowledge Modules (SKM) expose data in the form of web services. See Creating and Using Data Services in *Administering Oracle Data Integrator*.

A Knowledge Module is a code template for a given integration task. At run time, Oracle Data Integrator sends this code for execution to the source and target systems. Knowledge Modules are also fully extensible. Their code is open and can be edited through a graphical user interface.

Knowledge Modules

1.12 Oracle Data Integrator Scenarios

Learn about Oracle Data Integrator Scenarios.

In Oracle Data Integrator (ODI), you can generate scenarios for packages, procedures, mappings, and variables. Scenarios are saved to the development work repository, which you can export to other development or execution repositories. You can run them from ODI Studio, or from the command line.

Scenarios

1.13 Oracle Data Integrator Load Plans

An introduction to load plans in Oracle Data Integrator.

When Oracle Data Integrator is used to populate very large data warehouses, it's common to have thousands of tables populated by using hundreds of scenarios. The execution of these scenarios must be organized in such a way that data throughput from the sources to the target is the most efficient within the batch window. Load plans help to organize the execution of scenarios in a hierarchy of sequential and parallel steps for these types of use cases. They provide features for parallel, sequential, and conditional scenario execution, restartability, and exception handling. Load plans can be created and modified in production environments.

Load Plans

1.14 Oracle Data Integrator Console

An overview of the Oracle Data Integrator Console.

Oracle Data Integrator Console allows you to manage and monitor Oracle Data Integrator run-time architecture, and browse run-time objects in a web-based console. It integrates with Oracle Fusion Middleware Console allowing Fusion Middeware administrators access to details of the Oracle Data Integrator sessions and components.

Console

2

Create Repositories and Connections for Oracle Data Integrator

Learn about master and work repositories in Oracle Data Integrator (ODI) and how to create them. Also learn about ODI connections and how to create them using the repositories.

Topics

2.1 How to Create Master and Work Repositories

Follow the steps to create a master and one or more work repositories for Oracle Data Integrator.

1. Use the Repository Creation Utility to create the repositories. The executable is in the `oracle_common/bin` directory in the Fusion Middleware home; launch it.
2. The first screen is just a welcome screen. Click **Next**.
3. Accept the default values and click **Next**.
4. Specify the database connection details. Once entered, you can see a dialog box showing the progress of the three checks carried out on the database, making sure it is suitable.
5. Select the repository components to install. Some are selected by default. Make sure you select the **Master and Work repository** option. Click **Next**.
   You can now see a dialog box showing the progress of the checks carried out on the prerequisites for the components you selected.
6. If all the prerequisite checks pass, then enter the password for the schema users for the repositories. It's easiest to use the same passwords for all schemas, however, make sure you choose the option that conforms to your company security policy. Click **Next**.
7. Enter the password for the Supervisor user, what type the work repository is, its name and the password for its user. Also specify the encryption algorithm. Click **Next**.
8. Map the tablespaces. This is where you specify the name of the schema owners, and the default and temporary tablespace names for each component. You can accept the default names, or choose to specify your own. Click **Next**.
9. The penultimate screen summarizes your selections, and from here, you create the repositories. Click **Create**. Note that it takes several minutes to create all the repositories.
10. Click **Close** after you have reviewed the summary of the repositories just created. .

Once you have created a master repository and at least one work repository, you then create a connection, log in to ODI, and connect to these repositories: Create a Connection

The OBE, Oracle Data Integrator 12c - Creating and Connecting to ODI Master and Work Repositories, gives detailed steps of how to create your master and work repositories.

2.2 Create a Connection

Follow the steps to create a connection to Master and Work repositories in Oracle Data Integrator.

Once you have a master repository and a work repository, you can create a connection in Oracle Data Integrator (ODI).

1. Start ODI Studio, and click **Connect to Repository**.
2. Enter Wallet Password if you are using the wallet. Enter the wallet password. Click **OK**.
3. Click **New** in the Oracle Data Integrator Login window.
   The Repository Connection Information dialog displays.
4. Enter a name for the Login.
5. Enter Supervisor as the user, and the password for it. (Note, it's what you entered when creating the repositories).
6. Enter the database connection details for the master repository.
7. Select **work repository** and enter the name of the one you just created.
8. Test the connection, and assuming it is fine, create the connection.
9. Now you can log into ODI.

See Connecting to the Master Repository in *Installing and Configuring Oracle Data Integrator*.

3
The Oracle Data Integrator User Interface

A brief explanation of the user interface of the Oracle Data Integrator including the different navigators and menu items. There are four navigators, Topology, Designer, Operator and Security.

Topics

3.1 The Topology Navigator

Learn about the topology navigator in Oracle Data Integrator.

Topology contains sections in which to define the physical architecture, the contexts and the logical architecture. Starting with Context, this is where you define the context, or say, the type of system you are running on. So it might be a test, development or production system, or it might be global or for a particular region. You specify one of the contexts as the default.

In Physical Architecture there are two nodes you can expand, Technologies and Agents. As you might expect, Agents is where you define the agents, which Oracle Data Integrator (ODI) uses to run jobs. Under technologies you can find all the possible data sources for ODI. In each one you can define data servers and the physical schemas or files to which you would connect. For example, under Oracle, you might define a data server as being a particular Oracle database, and for it you can define the schema to which you connect. The data sources don't just include databases, you can define flat, complex and XML files, Big Data, and applications such as from SAP or Hyperion as data sources.

Under Logical architecture you have a duplicate of what is defined in the physical, but here you specify the context for each logical item. If you have defined three contexts (say Development, Test and Production), then you create a logical data server in each context to correspond to the physical data server. Similarly for each physical agent there is a logical agent for each context.

You can specify the languages and various elements within them. For example, if you import SQL as a language, then you can group functionality as sub-languages, for example, aggregation, and then have individual aggregate functions (for example AVG, COUNT) defined as part of the aggregation sub-language.

You can view the properties of the repositories, both master and work, that are open. You can erase the work repository and create a new one.

The final section in the Topology navigator allows you to view generic actions, for example, Create Table, and import or create new ones.

See Overview of Oracle Data Integrator Topology in *Developing Integration Projects with Oracle Data Integrator*.

See Topology Navigator in *Understanding Oracle Data Integrator*.

3.2 The Designer Navigator

Learn about the Designer navigator in Oracle Data Integrator.

The Designer navigator is where you build your projects, models, load plans, and scenarios. There is a menu of five expandable sections entitled Projects, Models, Load Plans and Scenarios, Global Objects and Solutions. To the side of these is a space where the object definitions and properties are displayed.

Under Projects you can build your projects, which include packages, mappings, procedures, variables, sequences, user functions, and markers. You also can import and edit knowledge modules.

Under Models you can define and build new models and data stores, including reverse-engineering the models.

Under Load Plans and Scenarios you can define and edit load plans and scenarios. Load Plans are objects that organize the execution of packages and scenarios at a high level. They provide features for restarting, exception handling, parallel, sequential, and conditional scenario execution. Each load plan is made up of steps, and these steps are made up of scenarios, or choose other objects (packages, interfaces, variables, and procedures), for which ODI automatically generates a scenario to run the step. Scenarios can also be run and scheduled and are made up of the same objects as load plans.

Under Global Objects you can define objects that are can be used in any procedure. For example, variables, sequences, templates, user functions, reusable mappings, markers and knowledge modules.

See Designer Navigator in *Understanding Oracle Data Integrator*.

3.3 The Operator Navigator

Learn about the Operator Navigator in Oracle Data Integrator

When you run an object, for example, scenario or load plan, in Oracle Data Integrator, you can follow its progress and on completion confirm whether it succeeded or failed in the Operator Navigator. Expand the Session List and then choose to view or find executions by date, agent, sessions, status keywords, or user. Or just view all executions. Find your execution and expand the hierarchy. Each step shows whether it succeeded or failed. A step might succeed, but with a warning. You can view the details of each step, the code run, or the error generated, for example.

Each execution is listed by the object name followed by a date and time stamp. That way you can make changes to the object you want to run and easily see the results for each change.

See Introduction to Operator Navigator in *Administering Oracle Data Integrator*.

See Operator Navigator in *Understanding Oracle Data Integrator*.

3.4 The Security Navigator

Learn about the Security Navigator in Oracle Data Navigator.

There are three sections, Profiles, Users, and Objects. Profiles contain sets of privileges for working with Oracle Data Integrator. A user is an Oracle Data Integrator user. A user inherits all the privileges granted to its various profiles, and privileges on objects or instances, given to

this user. Objects include agents, projects, models, data stores, scenarios, mappings, and even repositories. You can grant users with privileges on instances on specific work repositories. For example, you may grant a developer user with the edit privilege on a scenario on the development repository and not on the production repository. You can thus assign privileges on methods, objects types, or specific object instances to users.

See Security Navigator in *Understanding Oracle Data Integrator*.

3.5 Menu Items

A short description of each of the menu items in Oracle Data Integrator.

Each of the main menu items in Oracle Data Integrator (ODI) is described below.

- File - usual functions, for example, New, Print, Exit.
- Edit - usual edit functions such as Cut, Copy, Paste.
- View - you can show the Editor and select Toolbars to display.
- Search - usual find functions, plus one to Find ODI Object.
- ODI - ODI-specific functions. Connect and Disconnect allow you to connect to a repository and disconnect from it, plus view the repository connection information. You can change the repository and wallet passwords and specify how to store passwords. You can switch authentication mode or repository compatibility mode. You can add, remove, or open tools, find locked objects and browse the versions of objects. You can view data, perform Change Data Capture functions, generate DDL, mappings, scenarios, services, or a server template. Finally you can reverse engineer a model.
- Tools - There are five options: Manage libraries, Quick Actions, Display Preferences, HTTP Analyzer, and a Groovy editor.
- Window - You can access lots of ODI functionality, for example, open the four navigators, the debugger, logs and adjust the windows layouts.
- Team - You can access functionality which helps members of a team work on the same repository and then merge or version objects.
- Run - You can run ODI objects, for example sequences or procedures, but debug them, by invoking a debug session or adding breakpoints.
- Help - includes the usual Help functions, plus links to the ODI forum and OTN.

There are also Accessibility Features in ODI to assist those with disabilities in using and navigating the product.

4

Oracle Data Integrator Agents

Learn about Oracle Data Integrator Agents, what they are and how to create them.

Topics

4.1 What is an Oracle Data Integrator Agent?

Learn what an Oracle Data Integrator Agent is.

There are three agents, stand-alone, stand-alone collocated, and Java Enterprise Edition. A more detailed explanation is available in Run-Time Agent in *Understanding Oracle Data Integrator*.

Agents are multi-threaded java programs that support load balancing and can be distributed across the information system. They hold their own execution schedule, which can be defined in Oracle Data Integrator (ODI), or they can be called from an external scheduler. Also they can be invoked from a Java API or a web service.

To be able to run jobs in ODI, you must have an agent configured and running. That requires the WebLogic domain to be configured correctly.

4.2 How to Create an Agent

A summary of the steps required to create an agent.

Once the domain is configured, you can create an agent.

1. In the Topology navigator, expand Physical Architecture and right click **Agent**.
2. Select **New Agent**. In the agent Definition enter a name for it and the host name and port on which it listens. Save it. (Leave the window open for later.)
3. Expand Logical Architecture and right click **Agent**.
4. Select **New Agent**. In the agent Definition enter a name for it
5. Specify the physical agent for each context. Save it.
6. Re-start the Administration server, (this assumes the Administration and Managed servers are not running).
7. Log into WebLogic console, once the Administration server has started, to update the Oracle Coherence parameters.
8. Expand the Environment node in the Domain Structure panel. Click **Coherence Clusters**.
9. Click **DataGridConfig** in the Coherence Clusters table.
10. Select the Configuration tab and ensure that the Cluster Listen port has been set. If there is no value then click and set Cluster Listen port to 9000. (If this port is in use then you may have to change it later on). Save your changes.

11. Click the Members tab. Select Part of the cluster under the Clusters section, and then select your managed server(s).
12. Start the managed server(s). The agent should now be deployed.
13. Expand Domain Structure and click **Deployments**. You can now see the agent.
14. Click **Deployment Order** twice in Summary of Deployments panel to sort it from low-to-high. You should see oracledіagent near the top, and it should be State=Active and Health=OK.
15. Return to Oracle Data Integrator and the physical agent page. Click **Test** and make sure the test is successful.

The following Oracle By Example tutorials provide more insight:

- Oracle Data Integrator 12c - Creating a Collocated Agent
- Oracle Data Integrator 12c - Creating a Standalone Agent
- Configuring Standalone or Standalone Collocated Agents to Use a Secure Transport

5
Data Servers

Learn what a data server is and how to create one.

Topics

5.1 What is a Data Server?

A brief description of what a data server is in Oracle Data Integrator.

Data Servers are defined in the Topology navigator. Each data server represents a technology, be that a database, a file type, or an application. For databases and applications, the data server logs in as a particular user on a particular instance; but you can define more than one user in a data server. For files, the data server represents the file type (for example, flat, complex, XML) and also the file name and directory. For each data server defined in physical architecture, you must define one in the logical architecture. For each context in the logical data server, you must specify a physical data server.

You use data servers to identify data sources and the targets (where the data is loaded).

5.2 How to Create a Data Server

Follow these steps to create a data server in Oracle Data Integrator.

First create the physical data server. This example assumes the technology is Oracle.

1. Click the Physical Architecture bar. Expand the Technologies node, scroll down and right-click the Oracle node.
2. Select **New Data Server**. In the resulting Definition page enter the data server's Name, Instance, system user name and system user password.
3. Click the JDBC tab. Click **Search** to the right of the JDBC Driver field. In the Drivers dialog box, select Oracle JDBC Driver and click **OK**.
4. Click **Search**, to the right of the JDBC URL field. In the URL examples dialog box, select the first URL in the Name list and click **OK**. If necessary, edit the JDBC URL as appropriate for your environment.
5. Click **Test Connection**. In the Confirmation dialog box, click **Yes** to confirm saving your data before testing the connection.
6. Click **OK** in the Information dialog box that displays.
7. In the Test Connection dialog box, click **Test**. Acknowledge the connection test as successful. (If not successful, correct the connection details.)

Create a physical schema for your data server.

1. Right-click the newly-created data server, and select **New Physical Schema**.

2. Select the schema name from the Schema (Schema) drop-down list in the new window that appears, then select the appropriate name from the Schema (Work Schema) drop-down list.
3. Click the **Default** check box, if this is to be the default schema for this data server.
4. Click **Yes** in the Confirmation dialog box that displays. Leave all the other fields unchanged. Click Save.
5. Click **OK** in the Information dialog box that displays.
6. Expand the data server hierarchy and confirm that the physical schema has been added to the data server.

Create a Logical Schema.

1. Select the Logical Architecture tab and expand the Technologies node in the Topology Navigator.
2. Right-click the technology you are using (in this example, it is Oracle), and select **New Logical Schema**.
3. Enter the name of the Logical Schema in the Definition page that displays.
4. Map it to a physical schema for each context. Save your logical schema.
   If instead of Oracle, the technology is a file, at this point you do not select the physical schema for the context, but, instead, select the physical file.

See Creating a Data Server in *Administering Oracle Data Integrator*.

6
Models

Learn about data models and reverse-engineering in Oracle Data Integrator, and how to create a model or define reverse-engineering.

Topics

6.1 What are Models?

Learn about models in Oracle Data Integrator.

A model in Oracle Data Integrator (ODI) is a description of a relational data model. It's a representation of the structure of a number of interconnected datastores stored in a single schema on a particular technology. The model in ODI does not, however, hold data, such as banking figures or names of customers. Instead, it holds only metadata: the names of columns, and the constraints that exist between tables. This metadata can be imported automatically into ODI by a process called reverse-engineering. You can also define the metadata manually, creating columns or constraints directly in ODI. These additions do not necessarily exist in any physical database table.

See Introduction to Models in *Developing Integration Projects with Oracle Data Integrator*.

6.2 What is Reverse-Engineering

Learn about reverse-engineering in Oracle Data Integrator, what it is and when you would use it.

Reverse-engineering is a completely automated process in ODI for retrieving metadata from a database. It can be used to create new models or to fill in gaps in existing models. ODI can perform reverse-engineering in two ways, Standard and Customized. Standard reverse-engineering uses the standard capabilities of the JDBC API. ODI queries the JDBC driver for information about table structures, foreign keys, and so on. It then writes these to the ODI repository, where all metadata is stored. Although it's technically independent of the technology being used, standard reverse-engineering requires a sufficiently capable JDBC driver. Customized reverse-engineering is quite different. In this case, ODI connects to the database by using the basic features of the JDBC driver and directly queries the system tables to retrieve the metadata. It then transforms and writes this metadata into the ODI repository. Naturally, the method for doing this varies greatly depending on the technology that is being reverse-engineered. Thus, for each technology, there is a specific Reverse-engineering Knowledge Module (RKM). This RKM tells ODI how to extract metadata for the given technology.

In addition to standard and customized reverse-engineering for databases, you can reverse-engineer files:

- Delimited files contain data separated by a delimiter, such as a comma or tab character. ODI can parse the file content directly and determine the number of columns and their name depending on the file header.
- Fixed format files allocate a specific length to each field, rather than using a defined delimiter between fields. They are reverse-engineered with a different method.

To collect metadata for these fields, you require a description of the fields in the COBOL copybook format. ODI can parse this type of file natively. ODI can also reverse-engineer metadata in XML files through standard reverse-engineering. ODI uses the ODI JDBC driver for XML to access them like any other data source. Similarly, lightweight directory access protocol (LDAP) directories can be reverse-engineered by using the standard method. Here, ODI uses the ODI JDBC driver for LDAP. You should, therefore, consider XML files and LDAP directories as normal databases for the purpose of reverse-engineering.

See Reverse-engineering in *Developing Integration Projects with Oracle Data Integrator*.

6.3 Create a Model

Follow these steps to create a data model in Oracle Data Integrator.

The basic procedure to create a model in ODI is as follows:

1. Create and set up an empty model.
2. Define the reverse-engineering strategy that ODI follows. This includes selecting a Knowledge Module and setting up other parameters. In particular, you must choose between standard and customized reverse-engineering.
3. Tell ODI to run the reverse-engineering process by using the strategy that you defined. If you use standard reverse-engineering, you can choose to import only certain metadata. This is known as performing a selective reverse.
4. Flesh out the model that you have reverse-engineered. This means that you add information that was not retrieved during the reverse-engineering process. For example, you add datastores, columns, constraints, or ODI-specific metadata.

The basic steps in creating and naming a model are as follows:

1. Select the Models view. You can choose to create the model in selected model folder or you can create a top-level model.
2. Click the **New Model** icon. The window for the new model appears.
3. Enter the name of your model. Like most objects in ODI, there is also a code to enter. By default, this is the same as the name, but uppercase with underscores instead of spaces.
4. Specify the technology of the underlying database in this field. A model must be linked to a specific technology.
5. Specify the logical schema to which the model belongs from the drop-down list. You later specify a context, which determines the physical schema that is be used as the source of the metadata.
6. Enter a description for your model.

See Creating a Model in *Developing Integration Projects with Oracle Data Integrator*.

6.4 Define a Reverse-Engineering Strategy

Learn the steps to standard or custom reverse-engineering in Oracle Data Integrator.

To define a reverse-engineering strategy in ODI, perform the following steps:

1. Click the Reverse Engineer tab in the model window.
2. Select the reverse-engineering type to be used.
   The standard mode requires a JDBC driver, which provides metadata for the specific technology. The customized mode requires a specific ODI Knowledge Module to directly access the system tables. This choice causes some differences in the reverse-engineering process, as is discussed.
3. Select the reverse-engineering context.
   Remember you previously defined only a logical schema, which describes where the model fits in the logical architecture. However, this logical schema might correspond to several physical schemas in different physical contexts, such as development, testing, and production.

   By selecting the context, you specify which one of those physical schemas is to be used to provide metadata for the model. You should therefore select the context that best conforms to the structure of the model that you want to create.
4. To start the reverse-engineering process, right-click the model, and then select **Reverse Engineer**.
   Reverse-Engineering Knowledge Modules (RKMs) are used to perform customized reverse-engineering of data models for a specific technology. Customized reverse-engineering is more complete and entirely customizable, but also more complex to run. RKMs are provided only for certain technologies.

To perform customized reverse-engineering by using an RKM, perform the following steps:

1. Click the Reverse Engineer tab of the model.
2. Select the **Customized** option button and fill in the following fields:
   - Context: the context used for the reverse-engineering process
   - Object Type: the type of objects to reverse-engineer (tables, view, and so on)
   - Mask: %
   - Select the Knowledge Module: “<Project_Name>.<Project_Folder>” for example, “Knowledge Module: HandsOnLabs.HandsOnLoads”
3. Click the **Reverse** button, and then click **Yes** to validate the changes. Click **OK**.
4. Click **OK** in the “session started” dialog box.

If you use standard reverse-engineering, you can individually select the tables to reverse-engineer. To do so, perform the following steps:

1. Click the Selective Reverse-Engineering tab.
2. Select the **Selective Reverse-Engineering** check box; this enables the next few options.
3. Select the **New Datastores** check box to reverse-engineer the datastores that do not exist in your model.
   Select the **Existing Datastores** check box to select the datastores that exist in your model. Selecting both options selects all datastores for reverse-engineering.
4. Select the **Objects to Reverse Engineer** check box to select the individual datastores that you want to include.

The datastores that are displayed depend on the New Datastores and Existing Datastores options that you just selected.

5. Select (with a check mark) the datastore that you want to include in the reverse-engineering process.
6. Click the **Reverse** button to launch the process.

You can manually add, remove, or edit any element of a model by using the ODI mapping. This includes creating datastores, columns, keys, and so on. You can do this through the Designer Navigator. Note that these changes are applied only to the model in ODI, they do not update the underlying model in the database. To edit models graphically, you can edit the model's diagram, a graphical depiction of the model and the relationship between its elements. You can drag and drop elements to define new relationships. This component also enables ODI to modify the model in the original database with your changes. Thus, you can create your model in ODI, and then generate the code to implement this on a database server.

See Reverse-engineering a Model in *Developing Integration Projects with Oracle Data Integrator*.

Steps 1.5 and 2.4 in the OBE Oracle Data Integrator 12c - Creating an ODI Project and Mapping: Flat File to a Table also take you through how to create and reverse-engineer a model.

7

Data Stores

Learn about data stores, what they are and how to create them.

Topics

7.1 What are Datastores

Learn what datastores are in Oracle Data Integrator.

A datastore in Oracle Data Integrator (ODI) is a data structure, for example a table in a relational database, or a JMS topic from a Message-Oriented Middleware, that can be used in a mapping as a source or target. All datasources in ODI, regardless of the underlying technology, appear as datastores, and can be manipulated and integrated in the same way. The datastores are grouped in data models, which contain all declarative rules and constraints.

See Datastores in *Developing Integration Projects with Oracle Data Integrator*.

7.2 Create a Datastore

Learn how to create a datastore in Oracle Data Integrator.

The easiest and most common way to create a datastore in ODI is by reverse-engineering its structure from the technology (for example, Oracle or MySQL database). However, you can create datastores in two other ways:

- Directly in the model.
- By using the Common Format Designer component.

In these cases, the datastores do not necessarily have to exist in the data server. This can be useful for creating a new text file with a structure that you define.

To create a datastore directly in a model, perform the following steps:

1. Select a model.
2. Right-click and select **New Datastore**.
   The Resource Name field refers to the physical name of the table in a database or the name of the file for a file-based datastore. If you use a mapping to create this datastore in a database, the resource name is the name of the table that is generated. The alias, on the other hand, is the name that you use to refer to the datastore in mappings, joins, and so on.
3. Set the name. This is the name that is displayed in ODI in the Model tree view. Often, this is similar to the resource name.
4. Provide a description for the datastore (optional).

After you set up the basic parameters, you may want to add columns to the definition. To do this, perform the following steps:

1. Click the Attributes tab.
2. Click the **Add Column** icon to add another column.
3. Define properties for each column.
   The meanings of the Name, Type, Logical length, and Scale options depend on the technology of the model. Not Null means that the field is mandatory. You can set other properties in the columns by double-clicking their names in the Models view.
4. Reorder the newly-added columns using the up and down arrows.

See Creating and Reverse-Engineering a Datastore in *Developing Integration Projects with Oracle Data Integrator*.

8
Data Flow

Learn about data flows in Oracle Data Integrator.

Topics

- What is a Data Flow

8.1 What is a Data Flow

Learn what is a data flow in Oracle Data Integrator.

A data flow is the automatic conversion of the business rules that were defined in a mapping. It carries out the joins, filters, mappings, and constraints from source data to target tables.

By default, Oracle Data Integrator (ODI) uses an Extract Load Transform (E-LT) strategy as it first extracts and loads the temporary tables and then finishes the transformations in the target relational database. It uses the target database as a staging area for loading source data into temporary tables before applying all the required mappings, staging filters, joins and constraints. Staging is a separate area in the relational database (either a schema or database) where ODI creates its temporary objects and executes some of the rules, for example, mappings, joins, final filters, and aggregations. The business rules are transformed into code by Knowledge Modules.

If source volumes are small (less than 500,000 records), the staging area can be located in ODI's In-Memory Engine (its in-memory relational database). In this case, ODI behaves like a traditional Extract Transform Load (ETL) tool.

See Data Flow in *Understanding Oracle Data Integrator*.

9 Projects

Find out about projects in Oracle Data Integrator and how and when to create them.

Topics

9.1 Projects and Folders

Learn about projects and folders in Oracle Data Integrator.

A project is a collection of ODI objects created by users for a particular functional domain. However, only certain objects can belong to projects. Similarly, certain objects always belong to projects. Packages, procedures, and mappings always belong to folders, and folders always belong to projects. Variables, sequences, and user functions either belong to projects or can be created with global scope. Knowledge Modules (KMs) either belong to projects or can be imported with global scope. Also some KMs are considered built-in (neither project nor global). Markers always belong to a project.

A folder is a hierarchical grouping beneath a project and can contain other folders and objects. Every package, mapping, or procedure must belong to a folder.

When should you create folders? One guideline is to create a folder per “package” or scenario. Thus, all mappings that are used in the same package are grouped together. The folder represents all that is necessary for a given execution unit. As a result, maintenance is typically simplified.

It's not always easy to know when to create a project or when to create a subfolder within the same project. As a general rule, if you have a new functional domain or are starting an integration project, you should create a new project. Also, if you specifically want to keep objects separate from each other, you should create them in separate projects. On the other hand, a folder is useful when you want to organize an existing project. When you have a large number of mappings, procedures, or packages in a project, you should consider grouping them into folders. You can also use folders to set up different security levels within the same project. Each folder can have its own unique privileges. Remember that projects do impose strict boundaries on information sharing. This means that objects created in one project cannot be used by another project. If you want to enable an object to be used by other projects, you should make the object global. Thus, you enable the global variables, sequences, Knowledge Modules, and user functions to be used by any project. Folders, however, do not impose boundaries. Thus, objects in one folder can be used by any other objects in a project.

See Introduction to Integration Projects in *Developing Integration Projects with Oracle Data Integrator*.

See Organizing Projects with Folders in *Developing Integration Projects with Oracle Data Integrator*.

9.2 Create Projects and Folders

Learn the steps to create a project, or create a folder in Oracle Data Integrator.

To create a new project in ODI, perform the following steps:

1. In the Designer Navigator, select **New Project** from the drop-down menu at the top of the Projects view.
2. Name the project.
   You should generally make it reflect the functional domain that it covers.
3. Change the code to anything you like. (optional)
   Even though the code is generated in UPPERCASE it doesn't have to remain that way.

   The code is used as a prefix when referring to the variables created within this project.

Default marker sets are automatically added to your blank project. Similarly, a folder is created automatically to hold the mappings, packages, and procedures that you develop.

You can create folders within projects or within other folders as subfolders. To create a new folder, perform the following:

1. Right-click the project or folder where you want to create the new folder.
2. Select **Insert Folder** from the context menu.
3. Give a name to the folder.

After the folder is created, you can drag it to other folders, or onto the parent project, to reorganize the structure of your project.

See Creating a New Project in *Developing Integration Projects with Oracle Data Integrator*.

See Organizing Projects with Folders in *Developing Integration Projects with Oracle Data Integrator*.

10 Mappings

Learn about Mappings in Oracle Data Integrator and how to create them.

Topics

10.1 What are Mappings

Find out what Mappings are in Oracle Data Integrator.

Mappings in Oracle Data Integrator (ODI) are the logical and physical organization of your data sources, targets, and the transformations through which the data flows from source to target. A mapping connects sources to targets through a flow of components such as Join, Filter, Aggregate, Set, Split, and so on. A mapping also references the Knowledge Modules (code templates) that are used to generate the integration process. Each of the sources and targets is represented by a datastore. Business rules define how the data progresses from source to target and the manipulations that occur along the way. You define the business rules in natural language and using ODI translate them into SQL. You can save mappings as reusable mappings to use again in different mappings.

Mappings are made up of several parts, datastores, datasets, re-usable mappings, connectors, knowledge modules, variables, sequences, user functions, and other components. Optionally, you can specify a staging schema.

You create and manage mappings using the mapping editor, which opens whenever you open a mapping. Mappings are organized in folders under individual projects, found under Projects in the Designer Navigator.

See Introduction to Mappings in *Developing Integration Projects with Oracle Data Integrator*.

See Reusable Mappings in *Developing Integration Projects with Oracle Data Integrator*.

10.2 Create Mappings

Learn how to create a mapping in Oracle Data Integrator.

To create a new mapping:

1. Go to the Designer Navigator, expand the project and folder where you want to create the mapping and select the Mappings node.
2. (optional) Select Create Empty Dataset if you want the new mapping to contain a new empty dataset.
3. Construct your mapping by dragging components onto the logical diagram of the mapping editor.
4. Connect and configure components by dragging connections between the components, dragging attributes across those connections, and modifying the properties of the components using the property inspector.

5. Use the physical diagram to define where and how the integration process is going to run on your physical infrastructure.
6. Run the mapping.

See Creating a Mapping in *Developing Integration Projects with Oracle Data Integrator*.

See Creating a Reusable Mapping in *Developing Integration Projects with Oracle Data Integrator*.

11
Knowledge Modules

Learn about knowledge modules in Oracle Data Integrator.

Topics

11.1 What are Knowledge Modules

Learn about knowledge modules in Oracle Data Integrator.

A knowledge module (KM) in Oracle Data Integrator (ODI) is a code template for a given integration task. Knowledge modules implement how the integration processes occur. Each knowledge module type refers to a specific integration task:

- Reverse-engineering Knowledge Modules (RKM) reverse-engineer metadata from the heterogeneous systems for Oracle Data Integrator.
- Jounalizing Knowledge Modules (JKM) handle Changed Data Capture (CDC) on a given system.
- Loading Knowledge Modules (LKM) load data from one system to another, using system-optimized methods. These KMs are used in mappings.
- Integration Knowledge Models (IKM) integrate data in a target system, using specific strategies (insert/update, slowly changing dimensions). These KMs are used in mappings.
- Check Knowledge Modules (CKM) control Data Integrity on the data flow. These KMs are used in data model's static check and mappings flow checks.
- Service Knowledge Modules (SKM) expose data in the form of web services.

Each Knowledge Module's code can be edited to improve performance, implement new methods, or comply with new regulations and corporate standards.

You can import the KMs into each project or make use of Global Knowledge Modules. These enable you to share specific Knowledge Modules across multiple projects. A best practice is to import as Global Knowledge Modules those KMs that are used frequently in multiple projects; you then only need import a Knowledge Module once, rather than import it into each project using it. Similarly, if you need to modify the Global Knowledge Module, the modification propagates to all projects using the Knowledge Module, rather than having to update each knowledge module in each project.

See Introduction to Knowledge Modules in *Developing Knowledge Modules with Oracle Data Integrator*.

11.2 Find the Knowledge Modules in a Project

Learn how to find the knowledge modules in a project in Oracle Data Integrator.

Follow these steps to find out the knowledge modules (KMs) in a project.

1. Expand the project in the Designer navigator and expand Knowledge Modules.
2. Expand the Knowledge Modules node to see nodes for the above listed KM types.
3. Expand each to see the KMs already in the project. If you expand each KM, you can see where they are used; if you double-click each KM you can see an Overview.

11.3 Add a Knowledge Module to a Project

Learn how to add a knowledge module to a project in Oracle Data Integrator.

Follow these steps to add a knowledge module to a project:

1. Expand the project in the Designer navigator and right-click **Knowledge Modules**.
2. Select **Import knowledge Modules**. The Import Knowledge Modules (XML File) dialog box appears.
3. Click the Search icon next to File import directory, and navigate to the folder `.../<Oracle_Home>/odi/sdk/xml-reference`.
4. Click **Open** to display the Import Knowledge Modules (XML Files) dialog box. Select the knowledge modules that you want to use in this project. Be very careful in making your selections because there are several knowledge modules that have similar names.
5. Click **OK**. Review the Import Report, then close it. Check the imported knowledge modules by expanding the relevant nodes under Knowledge Modules.

See Importing and Replacing Knowledge Modules in *Developing Integration Projects with Oracle Data Integrator*.

11.4 Import a Global Knowledge Module

Learn how to import a global knowledge module in Oracle Data Integrator.

Follow these steps to import a global knowledge module:

1. In the Designer Navigator, expand the Global Objects tab.
2. Right-click **Integration (IKM)** and select **Import Knowledge Modules**. The Import Knowledge Modules (XML File) dialog box appears.
3. Select the knowledge modules that you want to use. Be very careful in making your selections because there are several knowledge modules that have similar names.
4. Click **OK**. Review the Import Report, then close it.
5. Check the imported knowledge modules by expanding the relevant nodes under Global Knowledge Modules.

12
Scenarios

Learn about scenarios in Oracle Data Integrator, what they are, how to create them, and how to run them.

Topics

12.1 What are Scenarios

Learn what scenarios are in Oracle Data Integrator.

Once you have built and tested a component in Oracle Data Integrator (ODI), you can create a scenario corresponding to its state. Scenarios can only be created in a development work repository. If you modify the component at a later date, the scenario code does not change. You can generate scenarios for packages, procedures, mappings or variables. Scenarios that execute a procedure, mapping, or variable are only single-step. Those for packages can be single-step or multi-step.

Scenarios can be exported and then imported into other work repositories, which can be development or execution repositories.

See Introduction to Scenarios in *Developing Integration Projects with Oracle Data Integrator*.

12.2 Create and Run Scenarios

Learn how to create and run scenarios in Oracle Data Integrator.

1. Start generating a scenario using one of the following:
    a. Double-click the package, mapping, procedure or variable for which you want to create a scenario to open the object editor. From the **ODI** menu, select **Generate**, and then **Scenario**.
    b. In the Designer Navigator, right-click the package, mapping, procedure or variable for which you want to create a scenario, and select **Generate Scenario**.
2. Enter the scenario name and a version.
3. Click **OK** twice and the scenario appears on the Scenarios tab and under the Scenarios node of the source object under the project. If there are variables in the scenario, then after clicking the first **OK**, you can define the variables that are parameters for the scenario.

See Generating a Scenario in *Developing Integration Projects with Oracle Data Integrator*.

See Generating a Group of Scenarios in *Developing Integration Projects with Oracle Data Integrator*.

You can execute a scenario from within Oracle Data Integrator Studio, or from the command line.

In Oracle Data Integrator Studio:

1. Right-click the scenario and select **Run**.
2. In the Run dialog box, set the execution parameters and Agent. Click **OK**. If there are any variables used as parameters, select the appropriate values when the Variable Values dialog box is displayed.

To start a scenario from the command line:

1. Change the directory to `<DOMAIN_HOME>/bin/` directory
2. Run the following command:
   - On UNIX:

```
./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>]*
```

   - On Windows:

```
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>"]*
```

See Executing a Scenario in *Administering Oracle Data Integrator*.

13
Procedures

Learn about procedures in Oracle Data Integrator, what they are and how you create them.

Topics

13.1 What is a Procedure

Learn about procedures in Oracle Data Integrator.

A procedure in Oracle Data Integrator (ODI) consists of a series of commands executed in sequence. Commands contain code that can be executed by database engines, the operating system where the Agent is running, or directly by ODI. You can also define options in the procedure to control its behavior at run time. A very useful property of procedures is that they are reusable and can be inserted into packages. Thus, just as procedures comprise commands, a package can contain several procedures as steps.

You use procedures in a package or a scenario, or you can run them directly from the Designer Navigator.

Here are some examples of the uses of procedures:

- The Email Administrator procedure contains a single command that calls the OdiSendMail email tool. You can then add an option to specify the email address. Thus, to change the email address of the administrator, you do not have to modify the procedure. You can even reuse the procedure in several places, each with different parameters to send emails to different people.
- You can create an ODI procedure to create an RDBMS table and populate it with data.

See Introduction to Procedures in *Developing Integration Projects with Oracle Data Integrator*.

See Using Procedures in *Developing Integration Projects with Oracle Data Integrator*.

13.2 Create a Procedure

Learn the steps to create a procedure in Oracle Data Integrator.

Use the following steps to create a procedure in ODI:

1. Create a Blank Procedure
   Like mappings and packages, procedures are contained within project folders.
2. Add Commands to a Procedure
   Now you have created a procedure that performs a series of tasks. However, it always performs the same tasks with the same parameters. To make the procedure more flexible, you can add options to it.
3. Add New Options to a Procedure

Add options to a procedure to make it customizable. You do not need to define any options.

These options function as parameters to the procedure. This means that you can use the same procedure in different environments with different parameters, without having to change any code.

Procedure options function as parameters.

Options can be used in two ways to control the way procedures are executed.

- Check box options serve to skip individual commands in the procedure. For example, you can have an option that, if false, completely skips a command.
- Value or text options are used in the text of commands to influence the behavior of the procedure at a smaller level. For example, the subject of an email can be a value option, which is then passed to the OdiSendMail tool.

In both cases, you define a default value for the option. This is the value that is used if no other value is specified when the procedure is used. When you call the procedure in a package, you can override any of the default values to customize the procedure for that particular context.

4. Save and run the procedure.
One of the most common ways to execute an ODI procedure is to run it from within a package. However, you can also execute a procedure manually for testing. You click the **Execute** button in the procedure window to do this. Default values for all options are used. You can, however, override the default values for the options. Click the **procedure** step, and then click the **Options** tab from the Properties pane. Here, you can select the values of options that are used when the procedure step is executed.

See Creating Procedures in *Developing Integration Projects with Oracle Data Integrator*.

An example of creating, populating and running a procedure is in Section 2 - Creating and Running a Procedure in the Oracle By Example tutorial Oracle Data Integrator 12c - Creating Procedures and Scenarios.

13.2.1 Create a Blank Procedure

Learn how to create a blank procedure in Oracle Data Integrator

Use the following steps to create a blank procedure in ODI:

1. Navigate to the Project and Project Folder where you want to create the procedure. Right-click the **Procedures** node and select **New Procedure**.
2. Provide the procedure a meaningful name and a description. You may want to include any limitations the procedure has in the description.
3. (optional) Set the default target and source technologies. If you don't set these technologies, you must set them individually for each command. You can always override these technologies for each individual step.

13.2.2 Add Commands to a Procedure

Learn how to add commands to a procedure in Oracle Data Integrator

Add one or more commands to make the procedure do something useful. You can create a procedure with only one command, but you often want more than one. Remember that commands are always performed in order.

1. Click the Tasks tab of the procedure window.

2. Click the **Add Command** (green plus) icon. A window for the command appears.
3. Enter a name for the command. This name appears in the Operator navigator when you execute the procedure, so you should try to make it specific and meaningful.
4. Select the **Ignore Errors** check box so that the procedure runs even if this command fails.
5. Select the appropriate options for the Command on Target and Command on Source options. (Note that a command can execute code in two places (Source and Target) within the same command.) Click **Command** on Target first.
6. Select the technology used in the command. This affects the code that you can use and how ODI generates its code.
7. Keep the Context as <Execution Context>, so that the user can select the context at the time the procedure is executed. The Schema and Context fields represent the logical schema and execution context for database queries.

13.2.3 Add New Options to a Procedure

Learn how to add options to new procedures in Oracle Data Integrator

Add new options to make the procedure customizable. You do not need to define any options.

1. Click the Options tab.
2. In Properties, enter the name as it appears in the list of options when you execute the procedure. Enter a description.

   The value that you enter in the Description field is displayed when you select the options that trigger a command in a procedure. It should be a short reminder of what the option does. Complete the Help field only when creating a Knowledge Module. It enables you to provide a longer description of the option, including any side effects or special notes.
3. Select the option type.

   This can be Boolean for an option that determines whether a given step is executed or not, Value for a numeric value, or Text for a text string. Complete the Default Value field. The default value specified for an option is the value that is used if you do not specifically set it in a package.
4. Save and run the procedure.

14
Packages

Learn about packages in Oracle Data Integrator and how to create them.

Topics

- What is a Package
- Create and Run a Package

14.1 What is a Package

Find out what are packages in Oracle Data Integrator.

An Oracle Data Integrator (ODI) package defines a complete data integration job. A job is made up of many smaller steps. Normally, you design these steps first, such as the ODI procedures and mappings. Other steps are created in the package.

See Introduction to Packages in *Developing Integration Projects with Oracle Data Integrator*.

14.2 Create and Run a Package

Learn how to create and run a package in Oracle Data Integrator.

Other ways can be used to create a package in ODI, but here we focus on just two ways: mappings and tools. The basic process for creating a package is shown in the following three tasks:

- Create a blank package
  You create a blank package and name it.
- Add steps to the diagram
  You can now load your package with the steps that it later executes. You can drag mappings from the **Project** view onto the package's **Diagram** tab. This creates a link rather than a copy to the mapping. Thus, you can keep working on your mapping, and your changes update the package. You can also add ODI tools into the package. Tools do useful things, such as sending email, copying files, or waiting for the data to arrive.

  The most useful package steps for most situations are either mappings or ODI tools. Mappings, as you know by now, transfer data from one or more source datastores into a target datastore. ODI tools perform a much wider variety of tasks, including sending email or waiting for data.

  Steps are created by dragging objects onto the package diagram, which is found on the Diagram tab. They are then sequenced by creating links from one step to the next.

  Mappings are reusable. When you create a mapping, you can use it several times in the same package. You can even copy it into several different packages simultaneously. Although you can use mappings from other projects in your package. It's best practice to avoid doing this, because it makes it difficult to keep your project organized.
- Arrange the package steps
  Finally, you arrange the package steps in a meaningful order. You begin by defining the first step to be executed. Then, you tell ODI what to do next when the first step succeeds.

You can also tell ODI what to do if any particular step in the package fails. Thus, you can link complex sequences of operations with error handling or error recovery.

Every package must have a first step. This is where the execution of a package always begins. After that, the sequence of steps splits in two directions: If the step executes successfully, the execution follows the green “ok” link. If the step fails, it follows the red “ko” link. Success is defined by the step returning a code 0. Any return code other than 0 is treated as a failure.

The first step you drag into the package diagram is always (initially) the First Step. You can of course designate a step you subsequently drag as the first one to execute. If you delete the step marked First Step, be sure to designate another step as the first one to execute. ODI displays an error message if no step is labeled First Step.

All the steps in the diagram must be either on the success path or connected by a failure path to it. If there are unlinked steps, the yellow triangle is highlighted. You can click the yellow triangle to check for the problem.

Be sure to test each mapping individually before using it in a package. Similarly, make sure you test individual steps in the package. To do this, right-click a step and select Execute Step. This is called unit testing. In some cases, it may not be possible to test an individual step - for example, when it requires a variable that is defined in the package. However, it is a good practice to test each step individually, whenever possible.

You should save your work frequently when working on packages.

To execute the package, click the **Execute** button in the package window. As with mappings, you can specify the context and the agent to run the package with. When you receive a notification that the session has started, open the Operator Navigator. The package represents a session in the Operator Navigator. Mapping and tool steps are represented as steps. Mapping steps are then subdivided into tasks that complete the individual data transfer operations. Tool steps, however, have only one task.

See Creating a new Package in *Developing Integration Projects with Oracle Data Integrator*.

See Running a Package in *Developing Integration Projects with Oracle Data Integrator*.

14.2.1 Create a Blank Package

Learn how to create a blank package in Oracle Data Integrator

To create a blank package, perform the following:

1. Locate the project and folder in the Projects view where you want to create the package.

   A package always belongs to a project, but you can use mappings from other projects in the package.

2. Right-click the **Packages** node and select **New Package**.
3. Provide a meaningful name to your package in the **Name** field.
4. Describe what the package does, by entering details in the **Description** field.

   This description appears in the package documentation and is useful when performing changes or maintenance.

5. To begin adding steps, click the **Diagram** tab.

14.2.2 Create a Step

Learn how to create a package step using a procedure in Oracle Data Integrator

To create a step, using a procedure:

1. Find the mapping, ODI tool, or procedure that you want to add. To add a procedure, expand the project and folder nodes in the Projects view. Then expand the Procedures node.
2. Drag the procedure onto the package diagram. You see an icon representing the procedure step in the package.
3. (optional) Change the name of the procedure step.

14.2.3 Arrange Package Steps

Learn how to arrange package steps in Oracle Data Integrator

To arrange the steps in a package:

1. Define the first step. Right-click a step and select **First Step**.
2. Define the success path for your package by connecting the normal sequence of steps into a continuous sequence. Click the **Next step on success** tool, which resembles a green arrow with the word **ok** above it.
3. Click all the steps in the sequence from the first to the last.
4. Click the **Select** tool (which resembles a cursor), to stop sequencing.
5. Add error handling by clicking the red **ko** button representing “Next step on failure.” Now click a step and the corresponding error recovery step. If this step fails, then report the error into a log file.
6. Delete unwanted links on steps, by selecting the step or link with the Select tool. Then use **Delete selection** on the toolbar, or right-click and select **Delete**. You can also use the keyboard by pressing the **Delete** key.

15

Load Plans

Learn about load plans in Oracle Data Integrator and how to create them.

Topics

15.1 What are Load Plans

Understand what an Oracle Data Integrator load plan is.

A load plan is an executable object in Oracle Data Integrator (ODI) that contains a hierarchy of steps. The steps can be executed conditionally, in parallel, or in series. The leaves of this hierarchy are scenarios. The set of scenarios that makes up a load plan can involve the execution of packages, mappings, variables, and procedures. Load plans also support exception handling strategies for scenarios that end in error.

Load plans can be started, stopped, and restarted from the command line, from Oracle Data Integrator Studio, Oracle Data Integrator Console, or a web service interface. They can also be scheduled by using the run-time agent's built-in scheduler or by using an external scheduler. When a load plan is executed, a load plan instance is created. Each attempt to run this load plan instance is a separate load plan run.

A load plan can be modified in production environments and steps can be enabled or disabled according to production needs. Load plan objects can be designed and viewed in the Designer and Operator Navigators. Various design operations (such as create, edit, delete, and so forth) can be performed on a load plan object if a user connects to a development Work Repository, but some design operations are not be available in an execution work repository. After it is created, a load plan is stored in the work repository. The load plan can be exported, and then imported to another repository and executed in different contexts.

A load plan is made up of a sequence of several types of steps. Each step can contain several child steps. Depending on the step type, the steps can be executed conditionally, in parallel, or sequentially. By default, a load plan contains an empty root serial step. This root step is mandatory and the step type cannot be changed.

The step types are:

- Serial - This defines a serial execution of its child steps. Child steps are ordered and a child step is executed only when the previous one is terminated.
- Parallel - This defines a parallel execution of its child steps. Child steps are started immediately in their order of priority.
- Run Scenario - This launches the execution of a scenario.
- Case - The combination of these steps allows conditional branching based on the value of a variable.
- When - If you have several When steps under a Case step, only the first enabled When step that satisfies the condition is executed.

- Else - If no When step satisfies the condition or the Case step does not contain any When steps, the Else step is executed.

Load plan instances and load plan runs are similar to sessions. The difference is that when a session is restarted, the existing session is overwritten by the new execution. The new load plan run does not overwrite the existing load plan run; it's added after the previous load plan runs for this load plan instance. The load plan instance cannot be modified at run time.

A scenario is an atomic run time component that is given to production. The production administrator can orchestrate the various scenarios by using load plans in order to optimize the time taken to execute the ETL jobs in the given time frame and ease the administration of all the ETL jobs.

See Introduction to Load Plans in *Developing Integration Projects with Oracle Data Integrator*.

See Running Load Plans in *Developing Integration Projects with Oracle Data Integrator*.

See Using Load Plans in Production in *Developing Integration Projects with Oracle Data Integrator*.

See Executing a Load Plan in *Administering Oracle Data Integrator*.

15.2 Create Load Plans

Learn the steps to create load plans in Oracle Data Integrator.

To create a simple load plan in ODI that executes two steps in parallel (in this example we load two dimensions, PRODUCT and TIME):

1. Go to the Designer Navigator and click the **Load Plans and Scenarios** tab. From the tab's menu, select **New Load Plan**.
2. Enter a suitable Name.
3. Click the **Steps** tab.
   The load plan editor opens with the root step node.
4. Add ODI objects to the load plan as parallel steps. Click **Add Step** (the green plus), and select **Parallel Step**. The Parallel step node appears under root_step.
5. Define two new procedures and place them under the parallel node so that they execute in parallel.
   Note that the default restart type for Parallel steps is "Restart all children," and the restart type for root_step is "Restart from failure."
6. Save your load plan.
7. Define two procedures to load dimensions, PRODUCT and TIME, and add them to the load plan Parallel step node.
   a. On the Projects tab of the Designer Navigator, expand the project and folder. Right-click **Procedures** and select **New Procedure**.
   b. Enter a suitable name for the procedure to load the PRODUCT dimension. Update the other fields as appropriate for your environment. Click **Save** and close the editing tab.
   c. Repeat the previous two steps to create and save the procedure to load the TIME dimension.
   d. Return to the Data Warehouse Load Plan editor. With the Parallel step highlighted, click **Add Step** (green plus) and select **Run Scenario Step**.
   e. Select an existing scenario for a step, or choose other objects (packages, interfaces, variables, and procedures), for which ODI automatically generates a scenario to run

the step. You now add the two procedures that you just defined as steps under the Parallel node of the load plan. On the Add Run Scenario Step screen, click **Lookup Scenario** (magnifying glass).

f. Select **Procedure** as the Executable Object Type. Select the procedure to load the PRODUCT dimension. Click **OK**.

g. Go to the New Scenario window, and add a name for the scenario so that ODI can create it to run this procedure, and click **OK**.
ODI returns to the Add Run Scenario Step window. By default, the name of the new scenario that ODI generates for the procedure, and the step name that you are adding to the load plan are the same as the procedure name.

h. Click **Finish**.
The first of two parallel steps appears.

i. Click the **Parallel** node of the Steps Hierarchy and click **Add Step** (the green plus sign) again. Select **Run Scenario Step**.

j. Repeat steps f to h to add your other procedure to load the TIME dimension to the Parallel node of the Steps Hierarchy.

To create a simple load plan to execute two steps serially:

1. Go to the Designer Navigator, click the **Load Plans and Scenarios** tab. From the tab's menu, select **New Load Plan**.
2. Enter a suitable Name.
3. Click the **Steps** tab.
The load plan editor opens with the root step node.
4. Add ODI objects to the load plan as serial steps.

a. Click **Add Step** (the green plus), and select **Serial Step**. The Serial node appears under root_step.

b. With Serial node selected, click **Add Step** and select **Run Scenario Step**.

c. Click **Lookup Scenario** in the Add Run Scenario Step window.

d. Select **Procedure** as the Executable Object Type. Select the procedure name. Click **OK**. In the New Scenario window, accept the default as the name of the scenario so that ODI creates it to run this procedure, and click **OK**.
ODI returns to the Add Run Scenario Step window.

e. Click **Finish**.
The first serial step appears.

f. Click the **Serial** node of the Steps Hierarchy, and click **Add Step** again. Select **Run Scenario Step**.

g. Click **Lookup Scenario**, and select **Mappings**.

h. Select the mapping name and click **OK**. In the New Scenario window, accept the default as the name of the scenario so that ODI can create it to run this mapping, and click **OK**.
ODI returns to the Add Run Scenario Step window.

i. Click **Finish**.

To create a simple load plan to execute two steps in parallel and to execute five steps serially (this example includes two procedures to load dimensions, TIME and PRODUCT):

1. Go to the Designer Navigator, click the **Load Plans and Scenarios** tab. From the tab's menu, select **New Load Plan**.

2. Enter a suitable Name.
3. Click the **Steps** tab.
   The load plan editor opens with the root step node.
4. Add ODI objects to the load plan as parallel steps. Click **Add Step** (the green plus), and select **Parallel Step**. The Parallel step node appears under root_step.
5. You define two new procedures and place them under the parallel node so that they execute in parallel.
   Note that the default restart type for Parallel steps is “Restart all children,” and the restart type for root_step is “Restart from failure.”
6. Click **Save** to save your load plan.
7. Define two procedures to load dimensions, PRODUCT and TIME, and add them to the load plan Parallel step node.
   a. Go to the Projects tab of the Designer Navigator and expand the **HandsOnLoads** > **HandsOn** folder. Right-click **Procedures** and select **New Procedure**.
   b. Enter a suitable name for the procedure to load the PRODUCT dimension. Update the other fields as appropriate for your environment. Click **Save** and close the editing tab.
   c. Repeat the previous two steps to create and save the procedure to load the TIME dimension.
   d. Return to the Data Warehouse Load Plan editor. With the **Parallel** step highlighted, click **Add Step** (green plus) and select **Run Scenario Step**.
   e. Select an existing scenario for a step, or choose other objects (packages, interfaces, variables, and procedures), for which ODI automatically generates a scenario to run the step. Add the two procedures that you just defined as steps under the Parallel node of the load plan. On the Add Run Scenario Step screen, click **Lookup Scenario** (magnifying glass).
   f. Select **Procedure** as the Executable Object Type. Select the procedure to load the product dimension. Click **OK**.
   g. Go to the New Scenario window and add a name for the scenario that ODI creates to run this procedure. Click **OK**.
      ODI returns to the Add Run Scenario Step window. By default, the name of the new scenario that ODI generates for the procedure, and the step name that you added to the load plan are the same as the procedure name.
   h. Click **Finish**.
      The first of two parallel steps appears.
   i. Click the **Parallel** node of the Steps Hierarchy, and click **Add Step** (the green plus sign) again. Select **Run Scenario Step**.
   j. Repeat steps e to h to add your other procedure to load the time dimension to the Parallel node of the Steps Hierarchy.
8. Add ODI objects to the load plan as serial steps.
   a. Go to the Steps Hierarchy, and highlight **root_step**. Click **Add Step** (the green plus sign) and select **Serial Step**.
      The Serial node appears in the Steps Hierarchy.
   b. Select the **Serial** node, click **Add Step** and select **Run Scenario Step**.
   c. Go to the Add Run Scenario Step window, and click **Lookup Scenario**.

d. Select **Procedure** as the Executable Object Type. Select the procedure name. Click **OK**. In the New Scenario window, accept the default as the name of the scenario that ODI can create it to run this procedure, and click **OK**.
ODI returns to the Add Run Scenario Step window.

e. Click **Finish**.
The first serial step appears.

f. Click the **Serial** node of the Steps Hierarchy, and click **Add Step** again. Select **Run Scenario Step**.

g. Click **Lookup Scenario**, and select **Mappings**.

h. Select the mapping name and click **OK**. In the New Scenario window, accept the default as the name of the scenario so that ODI can create it to run this mapping, and click **OK**.
ODI returns to the Add Run Scenario Step window.

i. Click **Finish**.

j. Repeat the previous four steps to add the remaining serial mappings.

9. Execute the load plan and examine the list of executed steps in the Operator Navigator.

a. Click the green arrow to execute the load plan. Click **OK** in the Start Load Plan window, and then click **OK** when the “Load Plan started” message appears.

b. Open the Operator Navigator, click **Refresh**, and examine the results of executing the load plan.
The steps in the load plan appear as a grouping of executions within seconds of each other. For each load plan step, ODI generated a scenario for execution. You can see that steps running in parallel.

See Creating a Load Plan *Developing Integration Projects with Oracle Data Integrator*.

16
Console

Learn about the console of Oracle Data Integrator and how to use it.

Topics

16.1 What is Oracle Data Integrator Console

Learn what Oracle Data Integrator Console is.

Business users (as well as developers, administrators and operators), can have read access to the Oracle Data Integrator (ODI) repository, perform topology configuration and production operations through a web based UI called ODI Console. This web application can be deployed in a Java EE application server such as Oracle WebLogic. ODI provides a plug-in, as well as the ODI Console to manage and monitor the Java EE and Standalone Agents, that integrates with Oracle Enterprise Manager Cloud Control and Oracle Fusion Middleware Control Console. It allows you to drill down into the details of ODI components and sessions. It is required for the Fusion Middleware Control Extension for ODI. It must be installed and configured for this extension to discover and display the ODI components in a domain.

Different types of user can use ODI Console in different ways:

- Administrators use it to create and import repositories and to configure the Topology.
- Production operators use it to manage scenarios and Load Plans, monitor sessions and Load Plan runs, and manage the content of the error tables generated by ODI.
- Business users and developers browse development artifacts in it, using, for example, the Data Lineage and Flow Map features.

See Introduction to Oracle Data Integrator Console in *Administering Oracle Data Integrator*.

16.2 Use Oracle Data Integrator Console

Learn how to use Oracle Data Integrator Console.

To connect to ODI Console:

1. Open a web browser, and connect to the URL where ODI Console is installed. For example:

   ```
   http://server_name:8001/odiconsole/
   ```

   where server_name represents the name or IP address of the server on which ODI Console is installed.
2. Go to the Repository list, and select the Repository connection corresponding to the master or work repository to which you want to connect.
3. Provide a User ID and a Password.

4. Click **Sign In**.

You can create, edit, delete, or view objects. You can import/export and run scenarios, and stop and re-start sessions. You can import/export, run, and stop and start load plans. you can also perform general administrative operations for ODI.

See Using Oracle Data Integrator Console in *Administering Oracle Data Integrator*.