Oracle® Cloud

Using Oracle Data Integrator Cloud

E93719-01
March 2018

Oracle Cloud Using Oracle Data Integrator Cloud,

E93719-01

Primary Author: Oracle Corporation

Contents

Preface

Topics

- Audience
- Documentation Accessibility
- Conventions
- Related Resources

Audience

Using Oracle Data Integrator Cloud Guide is intended for administrators who perform ongoing configuration tasks in Oracle Data Integrator Cloud.

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`.

Accessible Access to Oracle Support

Oracle customers who 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.

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.

Related Resources

See these Oracle resources:

- Getting Started with Oracle Cloud
- Oracle Public Cloud

  `http://cloud.oracle.com`

1

Getting Started with Oracle Data Integrator Cloud

Learn about Oracle Data Integrator Cloud, what it is and how to use it.

Topics:

- What is Oracle Data Integrator Cloud
- Before you Begin with Oracle Data Integrator Cloud
- How to Begin with Oracle Data Integrator Cloud
- Accessing Oracle Data Integrator Cloud
- Users

What is Oracle Data Integrator Cloud

Data integration ensures that information is timely, accurate, and consistent across complex systems. This section provides an introduction to data integration and describes how Oracle Data Integrator Cloud provides support for data integration.

- What is Data Integration?
- About Oracle Data Integrator Cloud
- What is E-LT?

What is Data Integration?

Integrating data and applications throughout the enterprise, and presenting them in a unified view is a complex proposition. Not only are there broad disparities in technologies, data structures, and application functionality, but there are also fundamental differences in integration architectures. Some integration needs are Data Oriented, especially those involving large data volumes. Other integration projects lend themselves to an Event Driven Architecture (EDA) or a Service Oriented Architecture (SOA), for asynchronous or synchronous integration.

Data Integration ensures that information is timely, accurate, and consistent across complex systems. Although it is still frequently referred as Extract-Transform-Load (ETL), data integration was initially considered as the architecture used for loading Enterprise Data Warehouse systems. Data integration now includes data movement, data synchronization, data quality, data management, and data services.

About Oracle Data Integrator Cloud

Oracle Data Integrator Cloud provides a fully unified cloud-based solution for building, deploying, and managing complex data warehouses or as part of data-centric architectures in a SOA or business intelligence environment.

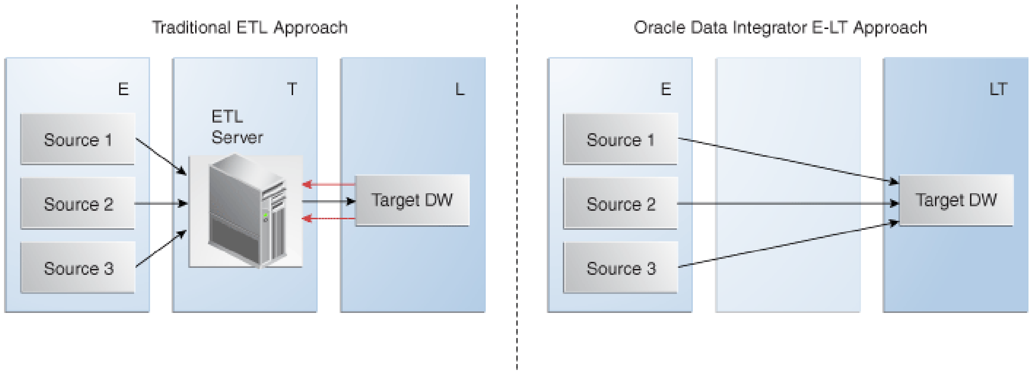
It combines all the elements of data integration — data movement, data synchronization, data quality, data management, and data services—to ensure that information is timely, accurate, and consistent across complex systems.

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

What is E-LT?

Traditional ETL tools operate by first *E*xtracting the data from various sources, *T*ransforming the data in a proprietary, middle-tier ETL engine that is used as the staging area, and then *L*oading 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, as shown in Figure 1-1.

Figure 1-1 Traditional ETL versus ODI E-LT



The data transformation step of the ETL process is by far the most compute-intensive, and is performed entirely by the proprietary ETL engine on a dedicated server. The ETL engine performs data transformations (and sometimes data quality checks) on a row-by-row basis, and hence, can easily become the bottleneck in the overall process. In addition, the data must be moved over the network twice – once between the sources and the ETL server, and again between the ETL server and the target data warehouse or Hadoop cluster. Moreover, if one wants to ensure referential integrity by comparing data flow references against values from the target data warehouse, the referenced data must be downloaded from the target to the engine, thus further increasing network traffic, download time, and leading to additional performance issues.

In response to the issues raised by ETL architectures, a new architecture has emerged, which in many ways incorporates the best aspects of manual coding and automated code-generation approaches. Known as *E-LT*, this new approach changes where and how data transformation takes place, and leverages existing developer skills, RDBMS and Big Data engines, and server hardware to the greatest extent possible. In essence, E-LT moves the data transformation step to the target RDBMS, changing the order of operations to: Extract the data from the source tables, Load the tables into the destination server, and then Transform the data on the target RDBMS using native SQL operators. Note, with E-LT there is no need for a middle-tier engine or server as shown in Figure 1-1.

Oracle Data Integrator supports both ETL- and E-LT-Style data integration. See the Designing E-LT and ETL-Style Mappings section in *Developing Integration Projects with Oracle Data Integrator* for more information.

Before you Begin with Oracle Data Integrator Cloud

There are a few prerequisites you need before you can use Oracle Data Integrator Cloud.

Before you begin with Oracle Data Integrator Cloud, you should have:

1. An Oracle Cloud account.
2. Your Oracle Cloud user name, password, and identity domain. You can locate your account details in the post-activation mail that you received from Oracle Cloud. For additional information, see *Getting Started with Oracle Cloud*.
3. Service Administrator role for your Oracle Cloud services. When the service is activated, Oracle sends the sign-in credentials and URL to the designated Account Administrator. The Account Administrator then creates an account for each user who needs access to the service.
4. A supported browser, such as:
   - Microsoft Internet Explorer 11.x+
   - Mozilla Firefox ESR 38+
   - Google Chrome 42+
   - Apple Safari 8.x and 7.x

How to Begin with Oracle Data Integrator Cloud

Here’s how to get started with free Oracle Data Integrator Cloud promotions and subscriptions:

1. Sign up for a free credit promotion or purchase a subscription.

   See Requesting and Managing Free Oracle Cloud Promotions or Buying an Oracle Cloud Subscription in *Getting Started with Oracle Cloud*
2. Access the Oracle Data Integrator Cloud service.

   See Accessing Oracle Data Integrator Cloud.

To grant access to others:

- Learn about users.

  See Users.
- Create accounts for your users and assign them appropriate privileges and roles.

  See Managing Users, User Accounts, and Roles in *Getting Started with Oracle Cloud*

Accessing Oracle Data Integrator Cloud

You can access Oracle Data Integrator Cloud through emails that you receive after subscribing or through the service web console.

To access Oracle Data Integrator Cloud:

1. Open your web browser and go to http://cloud.oracle.com.
2. Click **Sign In**.
3. From the Cloud Account menu, select **Cloud Account with Identity Cloud Service**.
4. Enter the name of your Cloud Account in the **Cloud Account Name** field.
5. Click **My Services**.
6. On the Sign In to Oracle Cloud page, enter your sign-in credentials.

If you don’t have your welcome mail to the **My Services** application, then contact your administrator.

Users

Administrators, Developers and Operators use the Oracle Data Integrator Studio to access the repositories. This Fusion Client Platform (FCP) based UI is used for administering the infrastructure (security and topology), reverse-engineering the metadata, developing projects, scheduling, operating and monitoring executions.

Business users (as well as developers, administrators and operators), can have read access to the repository, perform topology configuration and production operations through a web based UI called *Oracle Data Integrator Console*. This Web application can be deployed in a Java EE application server such as Oracle WebLogic.

ODI Studio provides four Navigators for managing the different aspects and steps of an ODI integration project:

- Topology Navigator
- Designer Navigator
- Operator Navigator
- Security Navigator

Topology Navigator

Topology Navigator is used to manage the data describing the information system's physical and logical architecture. Through Topology Navigator you can manage the topology of your information system, the technologies and their datatypes, 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 will enable Oracle Data Integrator to execute the same mappings in different environments.

Designer Navigator

Designer Navigator is used to design data integrity checks and to build transformations such as for example:

- 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
- Customization of the generated code

The main objects you handle through Designer Navigator are Models and Projects.

Operator Navigator

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

Security Navigator

Security Navigator is the tool for managing the security information in Oracle Data Integrator. Through Security Navigator you can create users and profiles and assign user rights for methods (edit, delete, etc) on generic objects (data server, datatypes, etc), and fine-tune these rights on the object instances (Server 1, Server 2, etc).

Part I

Oracle Data Integrator Cloud Architecture, Components, and Workflows

Here you can learn about Oracle Data Integrator Cloud Services architecture, components and workflows.

Topics:

2

Oracle Data Integrator Cloud Architecture

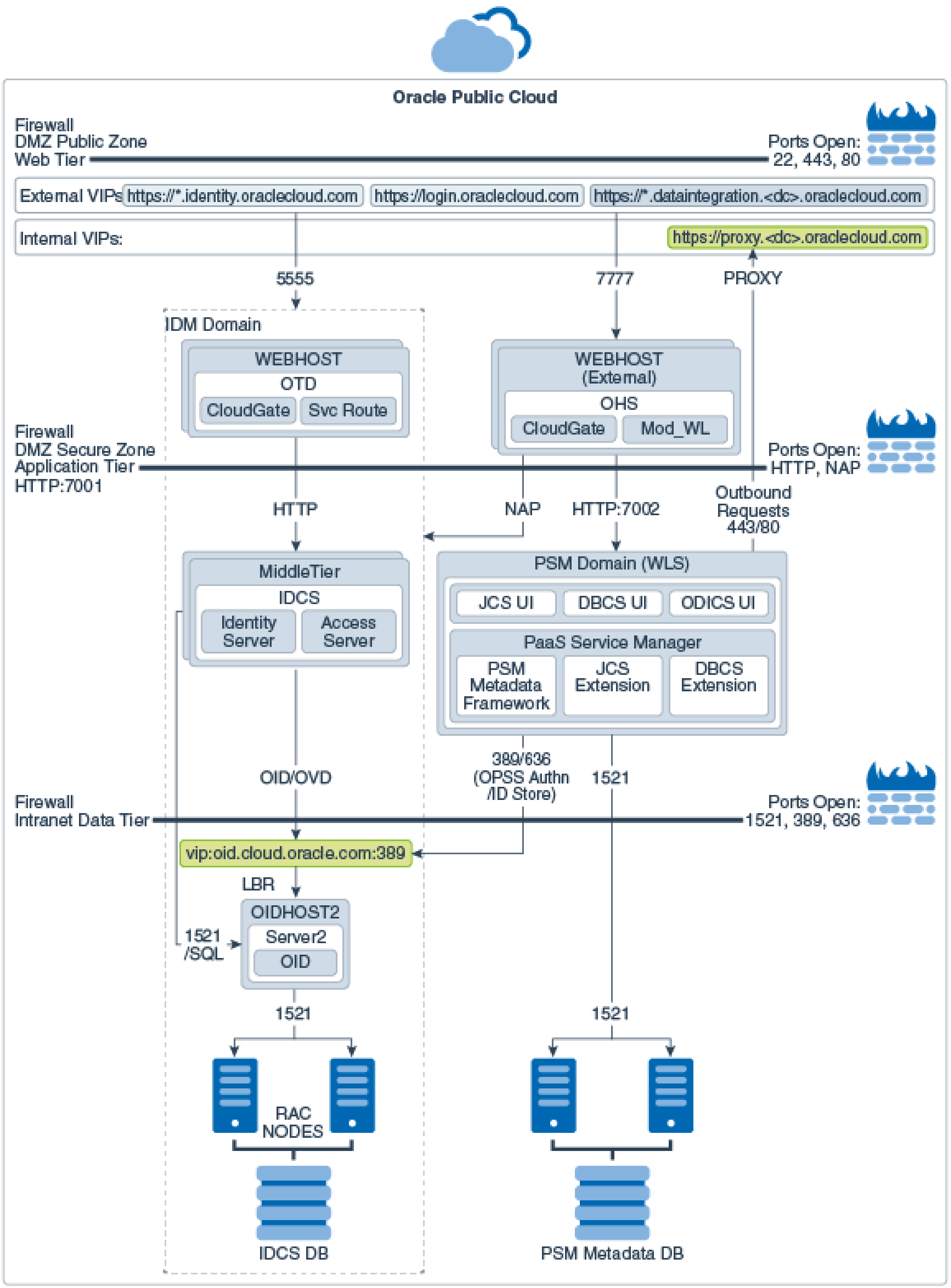
Review Oracle Data Integrator Cloud’s architecture to get a better understanding of how all the pieces work.

Topics:

Oracle Data Integrator Cloud Architecture

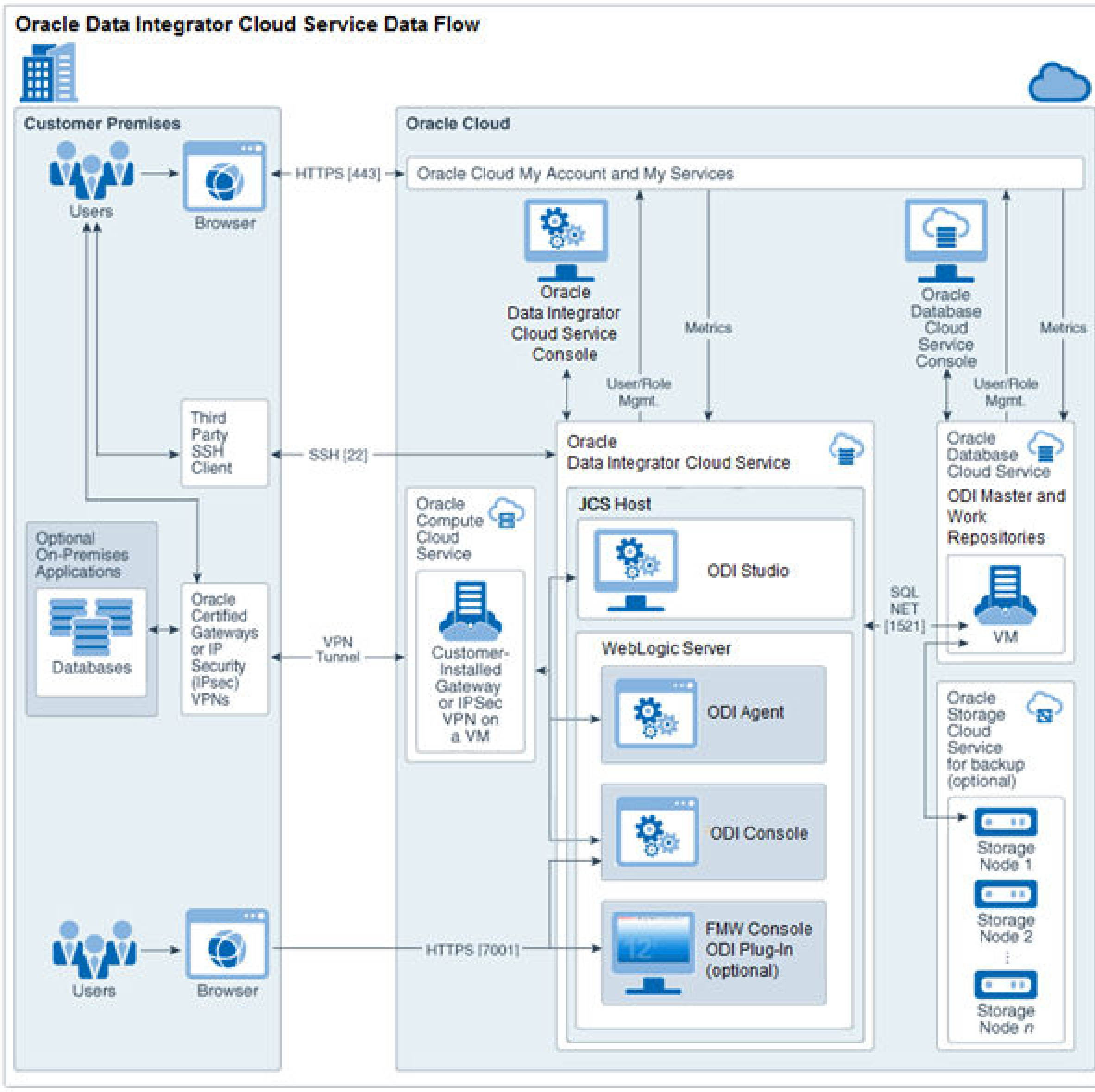
Review Oracle Data Integrator Cloud’s architecture to get a better understanding of how all the pieces work.

Oracle Data Integrator Cloud is built on top of the infrastructure and functionality provided by Oracle Public Cloud. The work involved in setting up and managing the virtual machines and storage resources for Oracle Data Integrator Cloud is done for you when you provision these services. With Oracle Public Cloud, you have the added benefit of Identity Management Cloud to manage users and roles across all your Oracle Cloud services.



Oracle Data Integrator Cloud removes the complexity in creating a real-time operation reporting schema, as well as a near real-time analytical data warehouse, enabling a novice user to perform the creation, initial load, and real-time synchronization in just a few clicks. Historically, these types of data integration tasks required assistance from ELT developers and Database administrators, as well as a detailed understanding of the source schema. Built using proven technologies, Oracle Data Integrator Cloud provides enterprise data integration and data governance features for today's enterprise.

Here, you can take a closer look at the Oracle Data Integrator Cloud architecture. This diagram shows the relationship between Oracle Data Integrator Cloud and your on-premise database. Oracle Data Integrator Cloud communicates to your on-premise agent through a SOCKS PROXY Tunnel in order to synchronize data between it and your on-premise database. Once your data is synchronized, you can use the power of Oracle Data Integrator Cloud's E-LT and tools in the cloud.


Oracle Data Integrator Cloud Service Data Flow
Customer Premises
Users
Browser
HTTPS [443]
Oracle Cloud
Oracle Cloud My Account and My Services
Oracle Data Integrator Cloud Service Console
Metrics
User/Role Mgmt.
Oracle Database Cloud Service Console
Metrics
User/Role Mgmt.
Third Party SSH Client
SSH [22]
Oracle Data Integrator Cloud Service
Oracle Database Cloud Service
ODI Master and Work Repositories
Optional On-Premises Applications
Databases
Oracle Certified Gateways or IP Security (IPsec) VPNs
VPN Tunnel
Oracle Compute Cloud Service
Customer-Installed Gateway or IPSec VPN on a VM
JCS Host
ODI Studio
SQL NET [1521]
VM
WebLogic Server
ODI Agent
Oracle Storage Cloud Service for backup (optional)
ODI Console
Storage Node 1
Storage Node 2
Storage Node n
Users
Browser
HTTPS [7001]
FMW Console ODI Plug-In (optional)

3

Oracle Data Integrator Cloud Components

The Oracle Data Integrator (ODI) distribution incudes several products and feature sets described here.

Topics:

- Oracle Data Integrator Components

Oracle Data Integrator Components

The Oracle Data Integrator (ODI) distribution incudes several products and feature sets.

The ODI distribution contains the products and feature sets described in Table 3-1. Unless otherwise noted, all products and feature sets are available with both the **Standalone Installation** and the **Enterprise Installation** types.

Table 3-1 Oracle Data Integrator Components and Features

Product	Feature Set	Description
Oracle Data Integrator features	ODI SDK	The ODI Software Development Kit (SDK) is a Java API for performing run-time and design-time operations.
	ODI Studio	Oracle Data Integrator Studio is used for administering the infrastructure (security and topology), reverse-engineering the metadata, developing projects, scheduling, operating and monitoring executions.
	ODI Standalone Agent	The standalone agent runs as its own process and is deployed in a domain as a system component. It can be configured in a standalone domain and managed by WebLogic Management Framework, or it can be colocated in a WebLogic domain and managed by Fusion Middleware Control.
	ODI J2EE	This is the Java EE agent, which is a Java EE application that is deployed and runs on a Managed Server configured in a WebLogic domain. This feature set is only available with the **Enterprise Installation** type.
	ODI Standalone Agent Template	This template provides the domain files required when your Oracle Data Integrator installation is not being managed by Oracle WebLogic Server. This feature set is only available with the **Standalone Installation** type.
	ODI Console	Oracle Data Integrator Console is a web-based console available for different types of users.
Internal Features	Apache Ant	This is a software tool used for automating build processes.
	FMW Upgrade	This is the upgrade assistant that can be used to upgrade your 11*g* Oracle Data Integrator software to 12*c*. For more information about FMW upgrade, see Upgrading Oracle Data Integrator from 11*g* in *Upgrading Oracle Data Integrator*.
	OPatch	The OPatch utility is a tool that allows the application and rollback of interim patches to Oracle products.

Table 3-1 (Cont.) Oracle Data Integrator Components and Features

Product	Feature Set	Description
	Repository Creation Utility	The Repository Creation Utility (RCU) is used for creating database schemas. This feature set is included with the **Standalone Installation** type. The **Enterprise Installation** type does not include RCU since Oracle Fusion Middleware Infrastructure is a requirement and RCU is included with the Oracle Fusion Middleware Infrastructure distribution.

4 Oracle Data Integrator Cloud Service Workflows

Example workflows for Oracle Data Integrator Cloud workflows.

Topics:

- Typical Workflow for using Oracle Data Integrator Cloud on a Single Instance
- Typical Workflow for using Oracle Data Integrator Cloud on a Clustered Instance

Typical Workflow for using Oracle Data Integrator Cloud on a Single Instance

Single server and clustered installations of Oracle Data Integrator Cloud offer you different capabilities. Learn about the workflows for each installation.

Task	Description	More Information
Purchase a subscription to Oracle Data Integrator Cloud Service	Provide your information and purchase a subscription to Oracle Data Integrator Cloud Service	How to Begin with Oracle Data Integrator Cloud Service
Create an Oracle Database Cloud Service Instance	Use the wizard to create a new Oracle Database Cloud service instance	Creating an Oracle Database Cloud Service Instance
Provision and Access an Oracle Data Integrator Cloud Instance	Use the wizard to create and configure new Oracle Java Cloud service and Oracle Data Integrator Cloud instances.	Provisioning and Accessing Oracle Data Integrator Cloud Service Instance

Typical Workflow for using Oracle Data Integrator Cloud on a Clustered Instance

Here is a typical workflow for installing and configuring Oracle Data Integrator Cloud on a clustered instance.

Task	Description	More Information
Purchase a subscription to Oracle Data Integrator Cloud Service	Provide your information and purchase a subscription to Oracle Data Integrator Cloud Service	How to Begin with Oracle Data Integrator Cloud Service
Create an Oracle Database Cloud Service Instance	Use the wizard to create a new Oracle Database Cloud service instance	Creating an Oracle Database Cloud Service Instance

Task	Description	More Information
Provision and Access a Clustered Oracle Data Integrator Cloud Instance	Use the wizard to create and configure new clustered Oracle Java Cloud service and Oracle Data Integrator Cloud instances.	Provisioning and Accessing the Clustered Oracle Data Integrator Cloud Instance

Part II

Creating Service Instances for Oracle Data Integrator Cloud

To use an instance of Oracle Data Integrator Cloud, you need to create an object storage classic container, provision a database, and then provision and access the Oracle Data Integrator Cloud instance.

Topics:

5

Creating an Object Storage Classic Container

A container is a storage compartment that provides a way to organize the data stored in Oracle Cloud Infrastructure Object Storage Classic.

Any user with the Service Administrator role can create containers. You should create at least one container for your account. Containers are similar to a directory structure but with a key distinction: unlike directories, containers cannot be nested. By default, all containers are of the standard storage class (as opposed to the archive storage class). You can also create containers when provisioning an Oracle Database Cloud Service deployment or Data Integration Platform Cloud instance.

Note:

Before you create your first container, make sure that the replication policy has been set for your account. See About Replication Policy for Your Account in *Using Oracle Cloud Infrastructure Object Storage Classic.*

Interface	Resources
Web Console (Not available on Oracle Cloud Machine)	Creating a Container Using the Web Console
RESTful API	Creating a Container Using the REST API
Java Library	See `createContainer` in *Java API Reference for Oracle Cloud Infrastructure Object Storage Classic*.
File Transfer Manager API	See `createContainer` in *Java API Reference forOracle Cloud Infrastructure Object Storage Classic File Transfer Manager*.

To create an archive container, you must set the X-Storage-Class header to Archive. For more information, see Creating Archive Containers in *Using Oracle Cloud Infrastructure Object Storage Classic.* (Not available on Oracle Cloud Machine)

Creating a Container Using the Web Console

(Not available on Oracle Cloud Machine)

If this is your first time using Oracle Cloud, make sure that you customize your My Services dashboard to show Storage Classic. To show Storage Classic on your Dashboard:

1. On your My Services Dashboard, click **Customize Dashboard**.
2. In the Customize Dashboard dialog, locate **Storage Classic** under Infrastructure, and then click **Show**.

Storage Classic then appears in your Dashboard. You can close the Customize Dashboard dialog to return to your Dashboard.

3. From the Storage Classic Action menu, select **Open Service Console**.
4. If this is your first time accessing Storage Classic, you'll be asked to set a **Geolocation Policy**. Select the appropriate policy from the menu, and then click **Set Policy**.
5. Click **Create Container.**

   The **Create Container** dialog box is displayed.
6. Enter a name for the container.

   > **Note:**
   >
   > Ensure that the container name complies with the input restrictions mentioned in Character Restrictions in *Using Oracle Cloud Infrastructure Object Storage Classic.*

7. Select `Standard` in the **Storage Class** field.
8. Click **Create.**

   The container is created and displayed in the console.

Creating a Container Using the REST API

cURL Command Syntax

```
curl -v -X PUT \
     -H "X-Auth-Token: token" \
     accountURL/containerName
```

- `token` is the authentication token obtained earlier from Oracle Cloud Infrastructure Object Storage Classic. See Authenticating Access When Using the REST API in *Using Oracle Cloud Infrastructure Object Storage Classic.*
- For the syntax of `accountURL`, see About REST URLs for Oracle Cloud Infrastructure Object Storage Classic Resources in *Using Oracle Cloud Infrastructure Object Storage Classic.*
- `containerName` is the name of the container to be created.

> **Note:**
>
> Ensure that the container name complies with the input restrictions mentioned in Character Restrictions in *Using Oracle Cloud Infrastructure Object Storage Classic.*

Note:

When you send a REST API request to Oracle Cloud Infrastructure Object Storage Classic, all non-ASCII characters in container names, object names and metadata values must be URL-encoded. For example, `my container` should be encoded as `my%20container`, where `%20` is the HTML encoding for the space character. Similarly, `my Über Container` should be encoded as `my%20%C3%9Cber%20Container`, where `%20` represents the space character and `%C3%9C` is the `Ü` character.

HTTP Response Codes

- Success: `201 Created`
- Failure: See A Error Code Reference for Object Storage Classic in *Using Oracle Cloud Infrastructure Object Storage Classic.*

cURL Command Example

```
curl -v -X PUT \
     -H "X-Auth-Token: AUTH_tkb4fdf39c92e9f62cca9b7c196f8b6e6b" \
     https://foo.storage.oraclecloud.com/v1/myservice-bar/FirstContainer
```

The following is an example of the output of this command:

```
> PUT /v1/myservice-bar/FirstContainer HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/
1.2.3 libidn/1.18 libssh2/1.4.2
> Host: foo.storage.oraclecloud.com
> Accept: */*
> X-Auth-Token: AUTH_tkb4fdf39c92e9f62cca9b7c196f8b6e6b
>
< HTTP/1.1 201 Created
< Date: Fri, 06 Mar 2015 10:34:20 GMT
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< X-Trans-Id: tx23a1084b8c674fdeae8d4-0054f982ac
< Cache-Control: no-cache
< Pragma: no-cache
< Content-Language: en
```

For information about getting details of a container, see Getting Container Metadata.

6

Provisioning Oracle Database Cloud Service Instances

Topics:

- Creating an Oracle Database Cloud Service Instance

Creating an Oracle Database Cloud Service Instance

Using Oracle Data Integrator Cloud requires a subscription to Oracle Database Cloud Service and instance created. Learn to create an Oracle Database Cloud Service instance for use with Oracle Data Integrator Cloud.

To create a Database Cloud Service instance, you'll use the Create Service wizard as described in the following procedure.

Before you begin

When you create a Database Cloud Service instance, you provide information used to create the instance itself and the Oracle database it hosts. Additionally, you may need to provide information about other resources, such as:

- An Oracle Database Cloud Service subscription
- Enough CPUs to create an Oracle Database Cloud Service instance
- An Oracle Cloud Infrastructure Object Storage Classic subscription

Procedure

To create Database Cloud Service instance:

1. Open the Oracle Database Cloud Service console.

   For detailed instructions, see Access the Oracle Database Cloud Service Console.

2. Click **Create Instance.**

   The Create Service wizard starts and the Service page is displayed.

3. On the Service page, provide a name and description for the database service instance, and provide information about its high-level characteristics. When you're finished, click **Next** to advance to the Service Details page.

Field Name	Description
Instance Name	The name of your database service instance. The name: • Must start with a letter. • Can't be more than 50 characters. • Can't contain any special characters except for a hyphen.

Field Name	Description
Description	A description for your new database service instance.
Notification Email	Enter your email address to receive notifications and status updates about this service instance.
Region	Select your region from the drop down menu or select No Preference.
Tags	Assign a tag from the list of tags or create a new tag.
Bring Your Own License	Check this box if you want to use your non cloud license.
Software Release	Select the release number from the drop down menu.
Software Edition	You can choose from: • Standard Edition • Enterprise Edition • Governance Edition
Database Type	You can choose from: • Single Instance • Database Clustering with RAC • Single Instance with Data Guard Standby • Data Guard Standby for Hybrid DR • Database Clustering with RAC and Data Guard Standby

4. On the Service Detail page, complete the Database Configuration section by providing information about the Oracle Database you want to create.

Field Name	Description
DB Name (SID)	The name of the database instance. The name: • Must not exceed 8 characters. • Must start with a letter. • Must contain only letters, numbers, or these symbols: _ (underscore), # (hash sign), or $ (dollar sign).
PDB Name	**(Available only for Oracle Database 12c.)** The name of the default pluggable database (PDB). The name: • Must not exceed 8 characters. • Must start with a letter. • Must contain only letters, numbers, or these symbols: _ (underscore), # (hash sign), or $ (dollar sign).

Field Name	Description
Administration Password **Confirm Password**	The password for the following administrative users: • Oracle Database administrative users • Oracle Application Express admin user The password: • Must be 8 to 30 characters in length. • Must contain at least one lowercase letter • Must contain at least one uppercase letter • Must contain at least one number • Must contain at least one of these symbols: _ (underscore), # (hash sign), or $ (dollar sign). • Must not contain the word “oracle”.
Usable Database Storage (GB)	The amount of storage in GB for actual database data.
Total Data File Storage (GB)	The computed amount of storage in GB that will be allocated to the deployment, including space for operating system and product binaries, supporting files, database data and configuration files, and so on.
Compute Shape	The number of Oracle Compute Units (OCPUs) and amount of memory (RAM) for each compute node of the new database deployment.
SSH Public Key **Edit**	The SSH public key to be used for authentication when using an SSH client to connect to a compute node that is associated with your database deployment. To display the public key input for VM access, click **Edit** and specify the public key using one of the following methods: • Select **Key File Name** to select a file that contains a Secure Shell (SSH) public key to assign to this service instance • Select **Key Value,** then copy and paste the value to associate the value of an existing SSH public key with this service instance. • Click **Create a New Key** to create a public and private key file, zipped, and ready for you to save both files on your computer. This option saves the public key information for the service instance. For more information, see Creating SSH Keys for Use with Oracle Cloud Services

5. On the Service Details page, complete the Advanced Settings fields by providing information about the following:

Field Name	Description
Listener Port	The port number for the Oracle Net Listener. The port number must be between 1521 and 5499 (inclusive).
Timezone	The time zone for the new database deployment. The default is Coordinated Universal Time (UTC).
Character Set	The database character set for the database. The database character set is used for: • Data stored in SQL CHAR data types (CHAR, VARCHAR2, CLOB, and LONG) • Identifiers such as table names, column names, and PL/SQL variables • Entering and storing SQL and PL/SQL source code This option is not available if Create Instance from Existing Backup is set to Yes.
National Character Set	The national character set for the database. The national character set is used for data stored in SQL NCHAR data types (NCHAR, NCLOB, and NVARCHAR2). This option is not available if Create Instance from Existing Backup is set to Yes.
Enable Oracle GoldenGate	Configures the database for use as the replication database of an Oracle GoldenGate Cloud Service instance.
Include “Demos” PDB	You can deselect this option to save some space. Selecting this option configures several PDB demos in the database with the service instance creation. These demos are for 12*c* new features.

6. On the Service Details page, complete the Backup and Recovery Configuration section, choosing a backup option for the database service instance and, depending on your choice, providing information about the Oracle Cloud Service container where cloud backups are to be stored.

Field Name	Description
Backup Destination	Controls how backups for the deployment are to be configured. Select Both Cloud Storage and Local Storage. All other options are incompatible with Data Integration Platform Cloud. For more information about backups and backup configurations, see About Backing Up Database Deployments on Database Cloud Service

Field Name	Description
Cloud Storage Container	The name of an existing Oracle Storage Cloud Service container or a new one to be created in the format: `instance-id_domain/container` where `instance` is the name of the Oracle Storage Cloud Service instance, `id_domain` is the id of the identity domain, and `container` is the name of the container. Note: Do not use the Oracle Storage Cloud container that you are using to back up Database Cloud Service deployments to cloud storage for any other purpose. For example, do not also use it to back up Oracle Java Cloud Service instances to cloud storage. Using the container for multiple purposes can result in billing errors.
Username	The user name of a user who has read/write access to the specified Oracle Storage Cloud Service container.
Password	The password of the user specified in Cloud Storage User Name .
Create Cloud Storage Container	Create a new Oracle Storage Cloud Service container as part of the database deployment creation. Specify the container name and the Cloud Storage user name and password in the preceding fields.
Total Estimated Monthly Storage (GB)	Storage for data files and backups.

7. On the Service Details page, complete the Initialize Data From Backup section if you are having the new database populated, or "instantiated," from the data stored in the Database Backup Cloud Service backup. When you're finished, click **Next** to advance to the Confirmation page.
8. On the Confirmation page, review the information listed. If you're satisfied with the information, click **Create.**

   If you need to change the information, use the navigation bar or **Previous** button at the top of the wizard to step back through the pages in the wizard. Click **Cancel** to cancel out of the wizard without creating a new Database Cloud service instance.

You can follow the steps in this tutorial to provision and access Oracle Data Integrator Cloud Service Instance, Provisioning and Accessing Oracle Data Integrator Cloud Service Instance. If you have a clustered Oracle Data Integrator Cloud Service Instance, then follow the steps in this tutorial to provision and access it, Provisioning and Accessing the Clustered Oracle Data Integrator Cloud Instance

7

Provisioning and Accessing Oracle Data Integrator Cloud Service Instance

Steps required to provision and then access an instance of Oracle Data Integrator Cloud

Topics:

Provisioning the Oracle Java Cloud Instance

Learn how to configure the Oracle Java Cloud instance on which you want to install and run Oracle Data Integrator Cloud

To begin, you must create a new Oracle Java Cloud instance that's linked to your database. This provision also include the Oracle Data Integrator installation software as part of its virtual machine set up.

1. Log on to your Oracle Cloud service and select *Java* from the main services menu.
2. From the Oracle Java Cloud Instances page, click **Create Instance**. Select *Java* from the drop-down list.
3. Enter the service details:
   a. **Instance Name** — Assign a unique name to your service.
   b. **Description** — Describe the service.
   c. **Notification Email** — Enter an email address for alerts and notifications.
   d. **Service Level** — From the dropdown, select `Oracle Java Cloud Service for Fusion Middleware`.
   e. **Software Release** — Select `Oracle Weblogic Server 12c(12.2.1.2)`
   f. **Software Edition** — Select `Enterprise Edition`

g. **Metering Frequency** — From the dropdown, select `Monthly`.

Click **Next**.

4. Click **Advanced** to enter the parameters for your new instance.

Enter parameter details for these sections:

- **WebLogic Configuration**

Parameter	Description
WebLogic Clusters	Click **Edit**. In the Manage Clusters dialog box, leave all the parameter values at the default values, except for **Compute Shape**. Set this to `OC5 - 4 OCPU, 30 GB RAM`
Compute shape	Shows as OC5 - 4 OCPU, 30 GB RAM
Server Count	This is a single server installation so leave as `1`
Fusion Middleware	Select `Oracle Data Integrator` from the drop-down list.
Domain Partitions	Leave it as `0`
Enable access to Administration Consoles	Select this.
Deploy Sample Application	Select if you want a sample application.

- **WebLogic Access**

Parameter	Description
SSH Public Key	You must supply a public SSH key for accessing the service securely. You can generate one using a third-party utility, or you can have the service create one for you. Press **Edit** to create the key, or supply one of your own.
Local Administrative Username	Assign an administrator user name for the Weblogic server
Password	Assign a password for the Weblogic administrator

- **Database Configuration**

Parameter	Description
Database Instance Name	Select the database you want to use to store the ODI repository from the dropdown list.
PDB Name	Leave as the default.
Administrator Username	Enter the database cloud service administrator user name, usually `sys`.
Password	Enter the database cloud service administrator password.

Parameter	Description
Add Application Schema	Leave as `No Application Schema added`.

- **Backup and Recovery Configuration**

Parameter	Description
Backup Destination	Select `Both Remote and Disk Storage` from the drop down list.
Cloud Storage Container	Enter the URL for your cloud storage container.
Username	Enter the user name for your cloud storage container.
Password	Enter the password for your cloud storage container user name.
Create Cloud Storage Container	Leave unchecked as the cloud storage container already exists.

- **Load Balancer**

Parameter	Description
Provision Local Load Balancer	Leave as `No`.

Click `Next`.

5. On the instance creation confirmation page, click Create.
6. After the service has been provisioned, the Oracle Java Cloud services page redisplays. Ensure the new service has been added to the list.

   To be able to access WLS console, you must follow the steps here in the documentation to enable control access in an Oracle Java Cloud Service.

Connecting to the Oracle Java Cloud Instance Environment Using SSH

Steps to use SSH to connect to Oracle Java Cloud Instance.

When you create an instance of the Oracle Java Cloud Service, all the Oracle Compute Cloud Service VMs required to support the service are provisioned and configured for you. You can access the services and resources provided by the VMs by logging in to the machine through an SSH tunnel. You'll access the VMs from your local machine by using the ssh command in a UNIX command shell.

1. To connect to the Oracle Java Cloud Service instance through SSH, you must find the IP address of the Administration Server VM hosting the instance.
   a. Sign in to the My Services application at http://cloud.oracle.com.
   b. In the Oracle Java Cloud Service section, click **Java**.
   c. On the Oracle Java Cloud Service page, click **Open Service Console**.
   d. On the Oracle Java Cloud Service Console, click the instance.

   e. The public IP address of the Administration Server is available under the **Virtual Machines** section. Make a note of the IP address.

> **Note:**
>
> You must wait for the instance creation to complete before you can see the IP address.

2. Start a Linux terminal shell and run the SSH utility.

   ```
   ssh -i path_to_private_key opc@IP_of_JCS_Instance_Admin_Server
   ```

   In the preceding command:

Parameter	Description
path_to_private_key	is the path to the SSH private key file that matches the public key used when your instance was created
IP_of_JCS Instance_Admin_Server	is the public IP address of the Admin Server VM in n.n.n.n format.
opc	is the user account.

   For example:

   ```
   ssh -i keys/id_rsa opc@192.0.2.1
   ```

   You are now logged in as user **opc**.

3. Switch to user oracle on the JCS Admin Server VM.

   ```
   sudo su oracle
   ```

4. Turn off the lock screen on the JCS Admin server.

   ```
   gconftool-2 -s -t bool /apps/gnome-screensaver/lock_enabled false
   ```

5. Start the VNC Server. If prompted for a password, enter one. Make a note of it.

   ```
   vncserver -nolisten local -geometry 1680x1050
   ```

6. Open a **new local terminal** to create an SSH tunnel on the VNC server port on the Administration Server VM.

   ```
   ssh -i path_to_private_key -L 5901:IP_of_Tunnel_Server:5901
   opc@IP_of_Admin_Server -N
   ```

   In the preceding command:

Parameter	Description
path_to_private_key	Path to your private key file.
IP_of_Tunnel_Server	The address of the server through which the tunnel is run.
IP_of_Admin_Server	The destination server (the address of the administration server for your Oracle Java Cloud instance).

For example:

```
ssh -i keys/id_rsa -L  5901:127.0.0.1:5901 opc@10.10.10.10 -N
```

7. To get desktop access for the Oracle Java Cloud Server instance,use the VNC Viewer on your local machine to connect to localhost:5901. You will be prompted for the password you entered in step 4.

   ```
   vncviewer
   ```

   In the Window that opens, enter **localhost:5901** and click OK.

   Once VNC connects you to the remote machine, you want to turn off the screen saver, otherwise you could find yourself locked out of the remote machine. Select **System/Preferences/Screensaver** menu item. In the resulting Screensaver Preferences dialog box, make sure **Activate screensaver when computer is idle** is not selected.

8. If you don't manage to turn off the screen saver in time, you will be locked out and the system will prompt you for the (unknown) oracle user password. If this happens, you need to reset the server and log back in.

   If you're not locked out, you can ignore this step.

   a. Using the same terminal session as in step 4, enter the following command to make sure you are the oracle user:

      ```
      sudo su oracle
      ```

   b. Find the number of the session which is now locked. You can use this number to kill the session. For example:

      ```
      ps –ef | grep vnc

      oracle         28596     1  0 16:07 pts/0    00:00:01 /usr/bin/Xvnc :1 -
      desktop test-w
      ls-1:1 (oracle) -auth /u01/app/oracle/tools/paas/state/homes/
      oracle/.Xauthority
      -geometry 1680x1050 -rfbwait 30000 -rfbauth /u01/app/oracle/tools/paas/state/
      homes/oracle/.vnc/passwd -rfbport 5901
      -fp catalogue:/etc/X11/fontpath.d -pn -nolisten local
      oracle   28601 28600  0 16:07 pts/0    00:00:00 vncconfig -iconic
      ```

      In this example, the number we want is **28596**. The session number returned when you run the command most likely will be different.

   c. Kill the session.

      ```
      kill -9 28596
      ```

   d. Delete the server temp files and folders relating to this session. Answer **yes** when prompted.

      ```
      rm –R /tmp/.X*
      ```

   e. Start the vncserver again.

      ```
      vncserver -nolisten local -geometry 1680x1050
      ```

   f. Repeat steps 5 and 6, but with a different port number. The simplest thing to do, is increase the port number by one. Previously the port number was given as 5901, so this time use 5902. Make sure the port is free, before you use it. Open a **new local terminal** to create an SSH tunnel the VNC server port on the Administration Server VM.

```
ssh -i path_to_private_key -L 5902:IP_of_Tunnel_Server:5902
opc@IP_of_Admin_Server -N
```

To get desktop access for the Oracle Java Cloud Server instance, use the VNC Viewer on your local machine to connect to localhost:5902. You will be prompted for the password you entered in step 4.

Installing Oracle Integrator

The steps to install Oracle Data Integrator.

As with all things Linux, directories are case sensitive.

Note:

Since Oracle WebLogic Server is already installed in your Java Cloud Service instance, there is no need to install the Fusion Middleware Infrastructure. It is already there.

1. From the **vncviewer**, open a terminal session and navigate to the /tmp directory. Check there are no temporary files present. If there are any, delete them using the rm command, so removing any possiblity of running out of disk space whilst installing ODI.

   Then navigate to the /u01/zips/upperstack directory

   ```
   cd /u01/zips/upperstack
   ```

   Inside the folder you will find the **odi** installer archive file. You can check by executing the ls command.
2. Unpack the archive file using the unzipcommand.

   ```
   unzip ODI.zip
   ```
3. This will unpack two jar files into the current directory. These files make up the ODI installer. To run the installer. Use the Java command:
4. To run the sintaller, use the Java command:

   ```
   java -jar fmw_12.2.1.2.6_odi_generic.jar
   ```

   to initiate the installation.
5. When prompted, set the **Inventory Directory** to /u01/app/oraInventory, then click OK to create the Central Inventory Directory.
6. When the Welcome screen opens, click Next.
7. Select Skip Auto Updates and then click Next.
8. For the installation location enter

   ```
   /u01/app/oracle/middleware
   ```

   , then click View to check the features that will be installed at this location. Click Next to move to the next step.
9. In the **Installation Type** form, select **Enterprise Installation**. Click Next.

> **Note:**
>
> The Standalone Installation is not supported in this configuration. Do not select it.

10. Click Next on the **Prerequisite Check** screen.

> **Note:**
>
> Depending on the configuration of instance, the ODI installer may display a warning stating that the JDK version is not certified. You can ignore the message.

11. On the **Installation Summary** form, click Install.
12. The Installation Progress screen will show the name and status of each component as it copied and configured. When the operation has completed, click Next.
13. Click Finish on the **Installation Complete** screen.

    You have successfully installed the Oracle Data Integrator.

Locating your Database Prefix

Steps to locate the prefix fo your database cloud service you will use with Oracle Data Integrator Cloud.

You will need to locate the user prefix of your database before moving on to the next section. Once you have located the prefix then note it down.

1. Start an SSH terminal session and connect to your Oracle Java Cloud instance.
2. Switch to the oracle user by executing the following command:

   ```
   sudo su oracle
   ```

3. The database prefix is located in the JDBC setting file for your server domain. Use the more command to examine the mds-owsm-jdbc.xml file using the following form:

   ```
   more /u01/data/domains/server_domain/config/jdbc/mds-owsm-jdbc.xml
   ```

   where the server_domain is the name assiged to your provisioned weblogic server. In this example, our server domain is called MyJCSODI_domain, so the command would be:

   ```
   more /u01/data/domains/MyJCSODI_domain/config/jdbc/mds-owsm-jdbc.xml
   ```

4. The terminal window will display the file, which will look similar to this:

   ```
   <name>mds-owsm</name>
     <jdbc-driver-params>
       <url>jdbc:oracle:thin:@MyJCSODI:1521/PDB1.fmwcert.ucfc2z3a.usdv1.cloud.com</
   url>
       <driver-name>oracle.jdbc.OracleDriver</driver-name>
       <properties>
         <property><name>user</name>
   ```

```
        <value>SP435951274_MDS</value>
      </property>
      <property>
        <name>oracle.net.CONNECT_TIMEOUT</name>
        <value>120000</value>
      </property>
      <property>
        <name>SendStreamAsBlob</name>
        <value>true</value>
      </property>
    </properties>
    <password-encrypted></password-encrypted>
  </jdbc-driver-params>
  <jdbc-connection-pool-params>
    <initial-capacity>0</initial-capacity>
    <connection-creation-retry-frequency-seconds>10</connection-creation-retry-
frequency-seconds>
    <test-frequency-seconds>300</test-frequency-seconds>
    <test-connections-on-reserve>true</test-connections-on-reserve>
    <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
    <seconds-to-trust-an-idle-pool-connection>0</seconds-to-trust-an-idle-pool-
connection>
  </jdbc-connection-pool-params>
  <jdbc-data-source-params>
    <jndi-name>jdbc/mds/owsm</jndi-name>
    <global-transactions-protocol>None</global-transactions-protocol>
  </jdbc-data-source-params>
</jdbc-data-source>
```

You will find the prefix under the **user** property. (All the characters before the underscore).

Creating Oracle Data Integrator Repository using the Repository Creation Utility (RCU)

Steps to create the database repository for Oracle Data Integrator Cloud, using the Repository Creation Utility.

This section describes how to use the Repository Creation Utility (RCU) to create the required Oracle Data Integrator schemas.

1. Run the following commands to start the Repository Creation Utility (RCU) and create the ODI schema:

   ```
   cd /u01/app/oracle/middleware/oracle_common/bin
   ./rcu
   ```

   Click Next when the opening screen is displayed.

2. On the **Create Repository** panel, accept the defaults and click Next.

3. You must use the Database you seleted when creating you ODI Cloud Service instance. The database details you define here will be designated as the supervisor database:

Field	Description
Database Type	At the time of writing, only Oracle is supported for the repository.
Host Name	This is the instance name of main Oracle database you attached to your Java Cloud instance.
Port	The designated port the database instance communicates through. For Oracle databases this is usually port 1521.
Service Name	The external name of the service can be found in the service details page of your Oracle Java Cloud instance.
Username	The system administrator user name of the database.
Password	The database administrator password
Role	This will be SYSDBA and will be selected automatically when you set the Username to SYS.

To get the dbhost:dbport/dbservice details, review the DBaaS properties. When you have filled in the details click Next.

4. The Repository Creation Utility will check the entries by connecting to the specified database. When the checks have been completed click OK.
5. Click the **Select existing prefix**. Select the correct user prefix from the drop down list. It should match the prefix you found in the previous section from the datasource in the Weblogic Administrator Console. Select Oracle Data Integrator component. Click Next.
6. The Repository Creation Utility will display a message window while it is checking prerequisites for creating the database. Press OK when it has finished.
7. Enter a new password for schema users.For a proof-of-concept, you can use the same passwords for all schemas. In a production environment, you might not want to do that. Note that RCU has different password complexity rules than the VM images, in particular about special characters. Click when you have entered and confirmed the password
8. On this page you will enter details for the ODI master and work repositories in the database you attached to your Oracle Java Cloud instance.

Custom Variable	Value
Supervisor password	The password of the SUPERVISOR user.
Work Repository Name	Use the default name WORKREP.
Work Repository Password	The password for the work repository.

Note:

Make a note of the **Supervisor password**. It will be required for step 23 of the next section, “Updating the Java Cloud Service Domain”

9. The default tablespaces are fine, just click Next.
10. You will now receive a warning stating that the RCU is about to create any missing table spaces. Press OK to continue.
11. The Repository Creation Utility will now proceed to create the tablespaces. Press OK when it's finished.
12. Make a note of the **Service Name** and the **Schema Owner**; you will need them when are running the ODI Studio later on in this tutorial. When you have noted them down, click Create to continue.
13. On the Summary screen, click Close.

    You now have the tablespaces and schemas needed to support Oracle Data Integrator Repositories in the database.

Updating the Java Cloud Instance Domain

Steps to update the Java Cloud Instance domain. on which you have installed Oracle Data Integrator Cloud.

Before you begin updating the Java Cloud Service Domain, use the WebLogic Administration Console to shutdown all Managed Servers and the Administration Server.

1. Access the Administration Server Console by using the following link:

    ```
    http://admin_server_host:admin_server_port/console
    ```

    .

2. In the **Domain Structure**, expand [+] Environment and click **Servers**. Click the **Control** tab. Select the managed server (not the adminserver), and click **Shutdown > Force Shutdown Now**.

    Click Yes to continue. Wait a minute.

3. On the same panel, select the adminserver and shut it down in the same way. This also kills the Console.

    Click Yes.

    Just ignore the web browser for the moment.

4. To update your domain, use the **Configuration Wizard** to extend (update) the Java Cloud Service domain with ODI. Open a terminal session and enter the following commmands:

    ```
    cd /u01/app/oracle/middleware/oracle_common/common/bin
    ./config.sh
    ```

5. In the **Update Domain** form, select **Update an existing domain**. In the **Domain Location**, select the following WebLogic domain directory. The domain_name will automatically point to the default domain created when provisioning this JCS instance: u01/data/domains/domain_name. Click to continue on to the next screen.

6. In the **Templates** form, do the following:
    a. Select **Update Domain Using Product Templates**.
    b. Select the following items (selecting one will cause others to be selected with green checks):

- Oracle Data Integrator - Agent
- Oracle Data Integrator - Agent Libraries
- Oracle Data Integrator SDK Shared Library Template
- Oracle Data Integrator - Console
- Oracle Data Integrator – Standalone Collocated Agent
- Oracle Enterprise Manager Plugin for ODI

  The following products should already be selected, and greyed out and in italics. They were selected when you created the domain in your Java Cloud Service instance:

  – Basic WebLogic Server Domain
  – Oracle Enterprise Manager
  – Oracle WSM Policy Manager
  – Oracle JRF
  – WebLogic Coherence Cluster Extension

Click Next.

7. The next screen is the for configuring the database type that the Repository Creation Utility requires for internal use. Again, the installer can fill in the details for you.

   Click Get RCU configuration, then press Next.

8. On the **Datasources** form, click the checkbox next to your ODI repositories configuration details. The installer will fill in the rest of the details for you. Then click Next.

9. The **JDBC Test** screen is used to make sure that your database connections are working correctly. Select your database connection and then press Test Selected Connections. When the test has completed, press Next.

10. In the **Credentials** form, enter the ODI Supervisor username and password for the SUPERVISOR key. (The password for the supervisor should be the variable you entered in the Repository Creation Utility section). Then add a new domain key.

    a. Click **Add** to add a new credential.
    b. In **Key Name**, enter the name of this domain as the key. This is the domain created for the JCS.
    c. In **Username** and **Password**, provide the Administrator user's Username and case-sensitive Password.
    d. In **Store Name**, select oracle.odi.credmap from the pull-down.

    Click Next.

11. In the **Advanced Configuration** form, select **Topology**, **System Components** and **Deployments and Services**. On many of the panels you will just accept the defaults.

12. In the **Managed Servers** form, update the required managed servers.

    a. To avoid targeting related libraries and applications to the default ODI server instead of the ODI cluster, delete the default ODI server from the list (you don't physically delete it) by selecting **ODI_server1** and click Delete.

   b. By default, the remaining managed server will be pre-populated. You can either rename it or use it as it is. For the purpose of this documentation, the Managed Server has been left with the default entry, however select Enable SSL if it is unchecked.

   Click Next.

13. In the **Clusters** form, just note the Cluster Name. It will be the first eight letters of your domain name (which is the same as the first eight letters of your JCS instance name) and the suffix of _cluster. WARNING: If you try to change the Cluster Name, it appears to work but will generate a Coherence error later, do not change the cluster name. Click Next.
14. Leave the **Server Templates** settings as they are and click Next.
15. Leave the **Dynamic Servers** settings as they are and click Next.
16. In the **Assign Servers to Clusters** form, ensure that the servers are assigned to the cluster. Click Next.
17. In the **Coherence Clusters** form, accept the defaults and click Next.
18. In the **Machines** form, click the **Unix Machine** tab to verify that it is populated. Give the machine a suitable name. Leave all other fields at their default values. Click Next.
19. In the **Assign Servers to Machines** form, ensure that the servers are assigned as shown in the image below. then click Next.

    Click Next.
20. Leave the **Virtual Targets** settings as they are and click Next.
21. Do not change anything on the **Partitions** screen. Click Next.
22. Do not change anything on the **System Components** page. This screen is used to configure a standalone collocated agent which can be optionally defined in the Topology. Click Next.
23. On the **ODI Configuration** page, set a SUPERVISOR password for Oracle Data Integrator standalone collocated agent. This screen is used to configure a standalone collocated agent which can be optionally defined in the Topology.
24. Do not change anything on the **Assign System Components** screen. Click Next.
25. The ODI applications are deployed to the cluster. Check that they have been deployed correctly. On the right side under Targets, confirm that you can see odiconsole and oraclediagent under your cluster > Application.

    By default, the ODI libraries are under the Admin Server. Move them into the cluster and out of Admin Server (these are two separate steps).

    a. In the **Deployments Targeting** form, on the left side under **Deployments > Library**, select oracle.odi-agent#2.0@12.2.1.2.6 and oracle.odi-sdk#2.0@12.2.1.2.6.
    b. On the right side under **Targets**, select the cluster name (your cluster name will be different).
    c. Click the middle **>** to move the items from the left to the right. They will go in alphabetically under Library.

    Do not click Next yet.

    Last you move the Libraries out of the Administration Server:

   a. On the right side under **Targets**, under **Server > domain_adminserver > Library** (your adminserver name will be different), select oracle.odi-agent#2.0@12.2.1.2.6 and oracle.odi-sdk#2.0@12.2.1.2.6.
   b. Click the middle **<** to move the items from the right. They don't actually go to the left, they just go away from the right.

   Now click Next.

26. Confirm that odiMasterRepository and odiWorkRepository are in the right side under **Targets/cluster/JDBC/JDBC System Resource**.
27. In the **Configuration Summary** form, leave the defaults and click Update.

    You can safely ignore the warning about No system component instance.
28. Click Next once the domain is configured, then click Finish.

    You have successfully updated the JCS domain to support ODI JEE Agents.

Creating Agents in the Master Repository Using Oracle Data Integrator Studio

Steps to create physical and logical Oracle Data Integrator agents

If there is no physical agent entry for an agent, then the agent startup will fail when starting the WLS Administration Server.

To create an agent using ODI Studio, see Creating an Agent in the Master Repository Using ODI Studio in *Installing and Configuring Oracle Data Integrator*.

Note:

Running ODI Studio on-premises on your local laptop or desktop is not supported as it may slow down the user interface to the point of being unusable. You should use VNC connected to the JCS instance to run ODI Studio.

1. Start ODI Studio from the command line by using the following commands:

   ```
   cd /u01/app/oracle/middleware/odi/studio
   ./odi.sh
   ```
2. You will be asked if you would like to import preferences from a previous ODI installation. Click No.
3. Log in to the Repository you created earlier using the RCU.
   a. Create a Login by clicking on **Connect to Repository** and then click the green plus.
   b. Fill in all of the connection information that match the entries you made in the RCU. These tables show how to complete the connection information:
      - Oracle Data Integrator Connector

Field	Description
Login Name	Choose a suitable login name for the connection.
User	This is the name you picked when creating the tablespaces in the Repository Creation Utility. Creating Oracle Data Integrator Repository using the Repository Creation Utility (RCU).
Password	The password you created for the SUPERVISOR

- Database Connection (Master Repository)

Field	Description
User	The user name created for the master repository in Repository Creation Utility. You should already have made a note of this value earlier on. Creating Oracle Data Integrator Repository using the Repository Creation Utility (RCU).
Password	The password for the master repository user.
Driver List	The list contains the JDBC drivers supported by the ODI Studio. Select Oracle JDBC driver.
Driver Name	The name of the JDBC driver will be filled in when you select from the Driver List.
URL	This is the location of the database given as a standard JDBC connection string.

c. Select **Work Repository** and click Browse to display the Work Repository selection window.
d. Select WORKREP and click OK.
e. Click Test. (If the selection worked, then the Test should work…)
f. If successful, click OK
g. Enter a Wallet password twice. Click OK.
h. All the information should be saved and pre-populated. Just click OK and the **Designer**, **Operator**, and **Topology** tabs should appear on the left.

4. Create a Physical Agent. On the **Topology** tab, expand **Physical Architecture**, right-click **Agents**, and select **New Agent**.

   Fill in the information for for Oracle Data Integrator J2EE agent:

   - **Name** — This is unique name given for the agent, and it must match the name of the J2EE agent application running on your server.
   - **Host** — The IP address, or host name, of the server that the J2EE agent is running on.

- **Port** — The port of the server that the J2EE agent is running on. This is the same value (9073 in our example) as the listen port set in step 12 of Updating the Java Cloud Service Domain section.

Click Save.

5. Create a Logical Agent. On the **Topology** tab, expand **Logical Architecture**, right-click **Agents**, and select **New Agent**.
6. For simplicity, keep all the names the same: OracleDIAgent. Click Save. Close the OracleDIAgent tab.
7. Optionally, you can repeat steps 4, 5 and 6, but for the standalone collocated agent.

   Create a Physical Agent. On the **Topology** tab, expand **Physical Architecture**, right-click **Agents**, and select **New Agent**.

   Fill in the information for for Oracle Data Integrator standalone collocated agent:

   - **Name** — This is unique name given for the agent, and it must match the name of the standalone collocated agent application running on your server.
   - **Host** — The IP address, or host name, of the server that the standalone collocated agent is running on.
   - **Port** — This is the same value as the server listen port, 20910, set in step 23 of Updating the Java Cloud Service Domain section.

   Click Save.
8. Create a Logical Agent. On the **Topology** tab, expand **Logical Architecture**, right-click **Agents**, and select **New Agent**.
9. For simplicity, keep all the names the same: OracleDIAgent1. Click Save. Close the OracleDIAgent1 tab.

   ODI is now provisioned on the JCS instance. The agents are running and available for ad hoc and scheduled work.

Starting the Administration Server

Once you've created the physical and logical agents in Oracle Data Integrator, you need to re-start the WLS Administration server.

The Administration Server normally does not host any user applications, it simply controls the other Managed Servers. Once the Managed Servers are running, the Administration Server only collects statistics and logs

Weblogic servers in the cloud environment are usually monitored and controlled by the Node Manager, so it is better to use the Node Manager to restart the Administration Server.

1. From your **vncviewer** console, open a new terminal window and run the Weblogic Scripting Tool using the following commands:

   ```
   cd /u01/app/oracle/middleware/oracle_common/common/bin
   ./wlst.sh
   ```

   The command prompt should change to show that you are now in the WLST environment.
2. Now connect to the Node manager using the **nmConnect** command.

```
nmConnect('weblogic_admin_user', 'admin_password', 'cloud_service_ip', 'port',
'domain','domain_path')
```

where

Parameter	Meaning
weblogic_admin_user	is the administrator user name for your Oracle Cloud Service
admin_password	is the administrator password.
cloud_service_ip	is the ip address of your Oracle Java Cloud service instance
port	is the listener port for the Node Manager. This is usually 5556
domain	is the name of your Weblogic domain. This can be found in the service details page of the Oracle Java Cloud service console.
domain_path	is the fully qualified path where the domain resides on your virtual machine.

For example:

```
nmConnect('weblogic', 'gTY78Hcv', '10.10.10.10', '5556', 'MyJCSODI_domain','/u01/
data/domains/MyJCSODI_domain')
```

Note:

All the parameters are enclosed in single quotes.

3. Now you are connected to the Node Manager, you can start your administration server. Use the following command:

   ```
   nmStart('MyJCSODI_adminserver')
   ```

   Remember to replace the parameter with the name of your own administration server domain.

Post-Configuration Tasks

Steps to update Oracle Coherence parameters

Use the WLS Administration Console to update the Oracle Coherence parameters. To configure Oracle Coherence:

1. Log into the Oracle WebLogic Server Administration Console.
2. In the **Domain Structure** panel, expand the **Environment** node. Click **Coherence Clusters**.
3. In the **Coherence Clusters** table, click **DataGridConfig**.
4. Select the **Configuration** tab and ensure that the **Cluster Listen port** has been set. If there is no value then click Lock & Edit and set **Cluster Listen port** to 9000. (If this port is in use then you may have to change it later on).

5. Click Save , then change to the **Members** tab. Select **Part of the cluster** under the **Clusters** section, and then select your managed server.

   Now click Save and Activate Changes.

The post-configuration tasks pertaining to Coherence membership are done. When you start the servers (next section), you may need to come back here to adjust the Coherence membership again.

Starting the Managed Server

To get your agents up and running, use these steps to start the Managed Server that was created during domain extension.

Start (or if necessary restart) the servers. Use the command line script to restart the Admin Server, and use the WebLogic Server console to start the Managed Servers. The Admin Server may already be running!

1. You can now access the WLS Administration Server Console by using the following link:

   ```
   http://admin_server_host:admin_server_port/console
   ```

2. In the **Domain Structure** panel, expand the **Environment** node. Click **Servers**.
3. Click the **Control** tab, select your managed server (click the checkbox, not the server name), click Start. (This assumes your node manager is running.)
4. Click Yes to continue. Wait a minute...
5. The agent should now be deployed in JCS. You can see it by going to **Domain Structure > Deployments** then in Summary of Deployments panel click **Deployment Order** twice to get it sorted low-to-high. You should see oraclediagent near the top, and it should be State=Active and Health=OK.
6. Go back to ODI Studio to test the agent. In **Topology > Physical Architecture > Agents > OracleDIAgent**, click Test.
7. If you chose to configure the standalone collocated agent, then you can test it by repeating step 6, but for **OracleDIAgent1**.

> **Note:**
>
> This Agent would need to be started first, which you do from the command line. For more information on how to do this, see: section 7.5 - Starting a Standalone or Standalone Collocated Agent Using Node Manager - of the Oracle Fusion Middleware Installing and Configuring Oracle Data Integrator.

The WLS admin server and managed servers are now running. All deployed applications are running.

ODI is now provisioned on the JCS instance. The agents are running and available for ad hoc and scheduled work.

8

Provisioning and Accessing the Clustered Oracle Data Integrator Cloud Instance

Instructions on how to install and configure Oracle Data Integrator Cloud on a clustered Oracle Java Cloud Instance, and then provision a Java enterprise agent.

Topics:

Provisioning the Clustered Oracle Java Cloud Instance

How to configure the clustered Oracle Java Cloud instance on which you want to install and run Oracle Data Integrator Cloud

To start with you will create a new clustered Oracle Java Cloud instance that is linked to your database. This provision will also include the Oracle Data Integrator installation software as part of its virtual machine set up.

1. Log on to your Oracle Cloud service and select Java from the main services menu.
2. From the Oracle Java Cloud Instances page, click the Create Instance button. Select Java from the drop-down list.
3. On the first page you will fill out the service details:
    a. **Instance Name** — This is the unique name you will assign to your service.
    b. **Description** — This is optional text you can use to describe the service.
    c. **Notification Email** — This is optional. You can enter an email address for alerts and notifications.
    d. **Service Level** — From the dropdown, select Oracle Java Cloud Service for Fusion Middleware.

e. **Software Release** — Select Oracle Weblogic Server 12c(12.2.1.2)
f. **Software Edition** — Select Enterprise Edition
g. **Metering Frequency** — Select Monthly from the dropdown.

Press Next when you have completed your entries.

4. The instance details page is where you fill in the parameters for your new instance. Click the Advanced button and then populate the form as described in the following table:

   Enter parameter details for these sections:

   - **WebLogic Configuration**

Parameter	Description
WebLogic Clusters	Click the Edit icon. In the Manage Clusters dialog box, leave all the parameter values at the default values, except for Compute Shape. Set this to OC5 - 4 OCPU, 30 GB RAM
Compute shape	Shows as OC5 - 4 OCPU, 30 GB RAM
Server Count	Set to 2, so you can load balance.
Fusion Middleware	Select Oracle Data Integrator from the drop-down list.
Domain Partitions	Since we're not interested in using partitions, leave it as 0
Enable access to Administration Consoles	Select this.
Deploy Sample Application	This is not necessary, but you can select it if you want.

   - **WebLogic Access**

Parameter	Description
Local Administrative Username	Assign an administrator user name for the Weblogic server
Password	Assign a password for the Weblogic administrator
Password	Assign a password for the Weblogic administrator

   - **Database Configuration**

Parameter	Description
Database Instance Name	Select in the drop down list the database you want to use to store the Oracle Data Integrator repository.
PDB Name	Leave as the default.
Administrator Username	Enter the database cloud service administrator user name, usually sys.
Password	Enter the database cloud service administrator password.

Parameter	Description
Add Application Schema	Leave as No Application Schema added.

- **Backup and Recovery Configuration**

Parameter	Description
Backup Destination	Select Both Remote and Disk Storage from the drop down list.
Cloud Storage Container	Enter the URL for your cloud storage container.
Username	Enter the user name for your cloud storage container.
Password	Enter the password for the above user name.
Create Cloud Storage Container	Leave unchecked as the cloud storage container already exists.

- **Load Balancer**

Parameter	Description
Provision Local Load Balancer	Select Yes.
Compute Shape	Select OC3 - 1.0 OCPU, 7.5GB RAM
Add Another Active OTD Node	Leave it unselected
Load Balancer Policy	Leave it as Least Connection Count

When you have entered all the details, click Next.

5. On the instance creation confirmation page, click Create.
6. After the service has been provisioned, you will be returned to the Oracle Java Cloud services page. Ensure the new service has been added to the list.

   To be able to access WLS console, you need to follow the steps here in the documentation to enable control access in an Oracle Java Cloud Service.

Connecting to the Oracle Java Cloud Instance Environment Using SSH

Steps to use SSH to connect to Oracle Java Cloud Instance.

When you create an instance of the Oracle Java Cloud Service, all the Oracle Compute Cloud Service VMs required to support the service are provisioned and configured for you. You can access the services and resources provided by the VMs by logging in to the machine through an SSH tunnel. You'll access the VMs from your local machine by using the ssh command in a UNIX command shell.

1. To connect to the Oracle Java Cloud Service instance through SSH, you must find the IP address of the Administration Server VM hosting the instance.
   a. Sign in to the My Services application at http://cloud.oracle.com.
   b. In the Oracle Java Cloud Service section, click **Java**.

c. On the Oracle Java Cloud Service page, click **Open Service Console**.
d. On the Oracle Java Cloud Service Console, click the instance.
e. The public IP address of the Administration Server is available under the **Virtual Machines** section. Make a note of the IP address.

> **Note:**
>
> You must wait for the instance creation to complete before you can see the IP address.

2. Start a Linux terminal shell and run the SSH utility.

```
ssh -i path_to_private_key opc@IP_of_JCS_Instance_Admin_Server
```

In the preceding command:

Parameter	Description
path_to_private_key	is the path to the SSH private key file that matches the public key used when your instance was created
IP_of_JCS Instance_Admin_Server	is the public IP address of the Admin Server VM in n.n.n.n format.
opc	is the user account.

For example:

```
ssh -i keys/id_rsa opc@192.0.2.1
```

You are now logged in as user **opc**.

3. Switch to user oracle on the JCS Admin Server VM.

```
sudo su oracle
```

4. Turn off the lock screen on the JCS Admin server.

```
gconftool-2 -s -t bool /apps/gnome-screensaver/lock_enabled false
```

5. Start the VNC Server. If prompted for a password, enter one. Make a note of it.

```
vncserver -nolisten local -geometry 1680x1050
```

6. Open a **new local terminal** to create an SSH tunnel the VNC server port on the Administration Server VM.

```
ssh -i path_to_private_key -L 5901:IP_of_Tunnel_Server:5901
opc@IP_of_Admin_Server -N
```

In the preceding command:

Parameter	Description
path_to_private_key	Path to your private key file.
IP_of_Tunnel_Server	The address of the server through which the tunnel is run.

Parameter	Description
IP_of_Admin_Server	The destination server (the address of the administration server for your Oracle Java Cloud instance).

For example:

```
ssh -i keys/id_rsa -L  5901:127.0.0.1:5901 opc@10.10.10.10 -N
```

7. To get desktop access for the Oracle Java Cloud Server instance,use the VNC Viewer on your local machine to connect to localhost:5901. You will be prompted for the password you entered in step 4.

   ```
   vncviewer
   ```

   In the Window that opens, enter **localhost:5901** and click OK.

   Once VNC connects you to the remote machine, you want to turn off the screen saver, otherwise you could find yourself locked out of the remote machine. Select **System/Preferences/Screensaver** menu item. In the resulting Screensaver Preferences dialog box, make sure **Activate screensaver when computer is idle** is not selected.

8. If you don't manage to turn off the screen saver in time, you will be locked out and the system will prompt you for the (unknown) oracle user password. If this happens, you need to reset the server and log back in.

   If you're not locked out, you can ignore this step.

   a. Using the same terminal session as in step 4, enter the following command to make sure you are the oracle user:

      ```
      sudo su oracle
      ```

   b. Find the number of the session which is now locked. You can use this number to kill the session. For example:

      ```
      ps –ef | grep vnc

      oracle        28596     1  0 16:07 pts/0    00:00:01 /usr/bin/Xvnc :1 -
      desktop test-w
      ls-1:1 (oracle) -auth /u01/app/oracle/tools/paas/state/homes/
      oracle/.Xauthority
      -geometry 1680x1050 -rfbwait 30000 -rfbauth /u01/app/oracle/tools/paas/state/
      homes/oracle/.vnc/passwd -rfbport 5901
      -fp catalogue:/etc/X11/fontpath.d -pn -nolisten local
      oracle   28601 28600  0 16:07 pts/0    00:00:00 vncconfig -iconic
      ```

      In this example, the number we want is **28596**. The session number returned when you run the command most likely will be different.

   c. Kill the session.

      ```
      kill -9 28596
      ```

   d. Delete the server temp files and folders relating to this session. Answer **yes** when prompted.

      ```
      rm –R /tmp/.X*
      ```

   e. Start the vncserver again.

      ```
      vncserver -nolisten local -geometry 1680x1050
      ```

f. Repeat steps 5 and 6, but with a different port number. The simplest thing to do, is increase the port number by one. Previously the port number was given as 5901, so this time use 5902. Make sure the port is free, before you use it. Open a **new local terminal** to create an SSH tunnel the VNC server port on the Administration Server VM.

```
ssh -i path_to_private_key -L 5902:IP_of_Tunnel_Server:5902
opc@IP_of_Admin_Server -N
```

To get desktop access for the Oracle Java Cloud Server instance, use the VNC Viewer on your local machine to connect to localhost:5902. You will be prompted for the password you entered in step 4.

Installing Oracle Data Integrator

The steps to install Oracle Data Integrator.

As with all things Linux, directories are case sensitive.

Note:

Since Oracle WebLogic Server is already installed in your Java Cloud Service instance, there is no need to install the Fusion Middleware Infrastructure. It is already there.

1. From the **vncviewer**, open a terminal session and navigate to the /tmp directory. Check there are no temporary files present. If there are any, delete them using the rm command, so removing any possiblity of running out of disk space whilst installing ODI.

   Then navigate to the /u01/zips/upperstack directory

   ```
   cd /u01/zips/upperstack
   ```

   Inside the folder you will find the **odi** installer archive file. You can check by executing the ls command.
2. Unpack the archive file using the unzipcommand.

   ```
   unzip ODI.zip
   ```
3. This will unpack two jar files into the current directory. These files make up the ODI installer. To run the installer. Use the Java command:
4. To run the sintaller, use the Java command:

   ```
   java -jar fmw_12.2.1.2.6_odi_generic.jar
   ```

   to initiate the installation.
5. When prompted, set the **Inventory Directory** to /u01/app/oraInventory, then click OK to create the Central Inventory Directory.
6. When the Welcome screen opens, click Next.
7. Select Skip Auto Updates and then click Next.
8. For the installation location enter

   ```
   /u01/app/oracle/middleware
   ```

, then click View to check the features that will be installed at this location. Click Next to move to the next step.

9. In the **Installation Type** form, select **Enterprise Installation**. Click Next.

> **Note:**
>
> The Standalone Installation is not supported in this configuration. Do not select it.

10. Click Next on the **Prerequisite Check** screen.

> **Note:**
>
> Depending on the configuration of instance, the ODI installer may display a warning stating that the JDK version is not certified. You can ignore the message.

11. On the **Installation Summary** form, click Install.
12. The Installation Progress screen will show the name and status of each component as it copied and configured. When the operation has completed, click Next.
13. Click Finish on the **Installation Complete** screen.

    You have successfully installed the Oracle Data Integrator.

Installing Oracle Data Integrator on the Second Oracle Java Cloud Instance Node

The steps to install Oracle Data Integrator on the second Oracle Java Cloud Instance.

As with all things Linux, directories are case sensitive.

1. Get the host name and private IP address of VM2 and of the Load Balancer:
   a. Sign in to the My Services application and go to the My Services Dashboard.
   b. In the **Oracle Compute Cloud Service** section, click the **Open Service Console** link.
   c. On the instance details page, make a note of the host name and the public IP address of the second node, and the Public IP address of the load balancer.
2. Copy the private key file to the Administration server.
   a. Close the VNC connection to the Administration server, and then close the two terminal sessions you used to create the SSH tunnel.
   b. Copy the private key file to the **/tmp** directory on the Administration Server. Open a terminal session and enter the command:

      ```
      scp -i path_to_private_key_file path_to_private_key_file opc@IP_of_JCS
      Instance_Admin_Server:/tmp
      ```

      In the preceding command:

Parameter	Description
path_to_private_key	is the path to the SSH private key file that matches the public key used when your instance was created
IP_of_JCS Instance_Admin_Server	is the public IP address of the Admin Server VM in n.n.n.n format.
opc	is the user account.

For example:

```
scp -i keys/key-20170605 keys/key-20170605 opc@192.0.2.1:/tmp
```

c. Now you need to make a copy of the file so that it can be accessed by the **oracle** user as you will be SSH tunnelling from the Administration server to the managed server (on the second node).

   Start by connecting to the Administration server using SSH:

```
ssh -i path_to_private_key_file opc@IP_of_JCS Instance_Admin_Server
```

   For example:

```
ssh -i keys/key-20170605 opc@192.0.2.1
```

d. Change to the **/tmp** directory:

```
cd /tmp
```

e. Make a copy of the private key file on the Administration server and give anyone full access to it:

```
cp key-20170605 key-20170605-1
chmod 777 key-20170605-1
```

f. Delete the private key file you originally copied over:

```
rm keyfile
```

   For example:

```
rm key-20170605
```

g. Change to the **oracle** user:

```
sudo su oracle
```

h. Make a copy the copy of the private key file and give it the original name:

```
cp keyfile-1 keyfile
```

   For example:

```
cp key-20170605-1 key-20170605
```

i. Now you have the private key file owned by the **oracle** user. All that needs doing is to set the permissions on it to be limited as before:

```
chmod 600 keyfile
```

   For example:

```
chmod 600 key-20170605
```

j. Change back from the oracle user to opc.

```
exit
```

k. Delete the private key file copy owned by opc:

```
rm keyfile-1
```

For example:

```
rm key-20170605-1
```

l. Close the terminal session.

3. Now, reconnect to the Administration server and start VNC again.

   Use the ssh command as in step 3 - Connecting to the Oracle Java Cloud Service Environment Using SSH - to log in to the second node as oracle user.

   a. Open a terminal session and enter the following command:

   ```
   $> ssh –i /tmp/private_key_file opc@host_name_of_VM2
   ```

   Where private_key_file represents the name of the private key file you just copied over, and host_name_of_VM2 represents the host name of the second node. **Note:** Use the host name of the second node, not its public IP address.

   For example

   ```
   ssh -i /tmp/key-20170605 opc@myjcsodi-wls-2
   ```

   Enter the passcode for the private key file, if prompted.

   b. Switch to the **oracle** user

   ```
   sudo su oracle
   ```

   c. Disable the screen saver and start the VNC server

   ```
   gconftool-2 -s -t bool /apps/gnome-screensaver/lock_enabled false
   vncserver -nolisten local -geometry 1680x1050
   ```

   d. Open a second terminal session and enter the following command to open a tunnel to the second node:

   ```
   ssh -i /tmp/private_key_file -L 5901:127.0.0.1:5901 opc@host_name_of_VM2 -N
   ```

   Where host_name_of_VM2 represents the host name of the second node. As before, it is not an IP address!

   For example:

   ```
   ssh -i /tmp/key-20170605 -L 5901:127.0.0.1:5901 opc@myjcsodi-wls-2 -N
   ```

   You can ignore any messages about the bind address already being in use. Enter the password for VNC when prompted.

   e. To get desktop access for the Oracle Java Cloud Server instance, use the VNC Viewer on the Administration server to connect to localhost:5901.

   ```
   vncviewer
   ```

   In the Window that opens, enter **localhost:5901** and click OK.

   f. Enter the password when prompted.

   g. Once VNC connects you to the second node, as before, you want to turn off the screen saver, otherwise you could find yourself locked out of it. Select **System/Preferences/Screensaver** menu item. In the resulting Screensaver

Preferences dialog box, make sure **Activate screensaver when computer is idle** is not selected.

4. Install the Oracle Data Integrator software as shown in the Installing Oracle Data Integrator section.
5. Once Oracle Data Integrator is installed, close the VNC Viewer and then shut the two terminal sessions that you opened on the Administration server for the SSH tunnel.

Locating your Database Prefix

Steps to locate the prefix fo your database cloud service you will use with Oracle Data Integrator Cloud.

You will need to locate the user prefix of your database before moving on to the next section. Once you have located the prefix then note it down.

1. Start an SSH terminal session and connect to your Oracle Java Cloud instance.
2. Switch to the oracle user by executing the following command:

   ```
   sudo su oracle
   ```

3. The database prefix is located in the JDBC setting file for your server domain. Use the more command to examine the mds-owsm-jdbc.xml file using the following form:

   ```
   more /u01/data/domains/server_domain/config/jdbc/mds-owsm-jdbc.xml
   ```

   where the server_domain is the name assiged to your provisioned weblogic server. In this example, our server domain is called MyJCSODI_domain, so the command would be:

   ```
   more /u01/data/domains/MyJCSODI_domain/config/jdbc/mds-owsm-jdbc.xml
   ```

4. The terminal window will display the file, which will look similar to this:

   ```
   <name>mds-owsm</name>
     <jdbc-driver-params>
       <url>jdbc:oracle:thin:@MyJCSODI:1521/PDB1.fmwcert.ucfc2z3a.usdv1.cloud.com</
   url>
       <driver-name>oracle.jdbc.OracleDriver</driver-name>
       <properties>
         <property><name>user</name>
           <value>SP435951274_MDS</value>
         </property>
         <property>
           <name>oracle.net.CONNECT_TIMEOUT</name>
           <value>120000</value>
         </property>
         <property>
           <name>SendStreamAsBlob</name>
           <value>true</value>
         </property>
       </properties>
       <password-encrypted></password-encrypted>
     </jdbc-driver-params>
     <jdbc-connection-pool-params>
       <initial-capacity>0</initial-capacity>
       <connection-creation-retry-frequency-seconds>10</connection-creation-retry-
   frequency-seconds>
   ```

```
    <test-frequency-seconds>300</test-frequency-seconds>
    <test-connections-on-reserve>true</test-connections-on-reserve>
    <test-table-name>SQL SELECT 1 FROM DUAL</test-table-name>
    <seconds-to-trust-an-idle-pool-connection>0</seconds-to-trust-an-idle-pool-
connection>
  </jdbc-connection-pool-params>
  <jdbc-data-source-params>
    <jndi-name>jdbc/mds/owsm</jndi-name>
    <global-transactions-protocol>None</global-transactions-protocol>
  </jdbc-data-source-params>
</jdbc-data-source>
```

You will find the prefix under the **user** property. (All the characters before the underscore).

Creating Oracle Data Integrator Repository using the Repository Creation Utility (RCU)

Steps to create the database repository for Oracle Data Integrator Cloud, using the Repository Creation Utility.

This section describes how to use the Repository Creation Utility (RCU) to create the required Oracle Data Integrator schemas.

1. Run the following commands to start the Repository Creation Utility (RCU) and create the ODI schema:

   ```
   cd /u01/app/oracle/middleware/oracle_common/bin
   ./rcu
   ```

   Click Next when the opening screen is displayed.
2. On the **Create Repository** panel, accept the defaults and click Next.
3. You must use the Database you seleted when creating you ODI Cloud Service instance. The database details you define here will be designated as the supervisor database:

Field	Description
Database Type	At the time of writing, only Oracle is supported for the repository.
Host Name	This is the instance name of main Oracle database you attached to your Java Cloud instance.
Port	The designated port the database instance communicates through. For Oracle databases this is usually port 1521.
Service Name	The external name of the service can be found in the service details page of your Oracle Java Cloud instance.
Username	The system administrator user name of the database.
Password	The database administrator password

Field	Description
Role	This will be SYSDBA and will be selected automatically when you set the Username to SYS.

To get the dbhost:dbport/dbservice details, review the DBaaS properties. When you have filled in the details click Next.

4. The Repository Creation Utility will check the entries by connecting to the specified database. When the checks have been completed click OK.
5. Click the **Select existing prefix**. Select the correct user prefix from the drop down list. It should match the prefix you found in the previous section from the datasource in the Weblogic Administrator Console. Select Oracle Data Integrator component. Click Next.
6. The Repository Creation Utility will display a message window while it is checking prerequisites for creating the database. Press OK when it has finished.
7. Enter a new password for schema users.For a proof-of-concept, you can use the same passwords for all schemas. In a production environment, you might not want to do that. Note that RCU has different password complexity rules than the VM images, in particular about special characters. Click when you have entered and confirmed the password
8. On this page you will enter details for the ODI master and work repositories in the database you attached to your Oracle Java Cloud instance.

Custom Variable	Value
Supervisor password	The password of the SUPERVISOR user.
Work Repository Name	Use the default name WORKREP.
Work Repository Password	The password for the work repository.

Note:

Make a note of the **Supervisor password**. It will be required for step 23 of the next section, “Updating the Java Cloud Service Domain”

9. The default tablespaces are fine, just click Next.
10. You will now receive a warning stating that the RCU is about to create any missing table spaces. Press OK to continue.
11. The Repository Creation Utility will now proceed to create the tablespaces. Press OK when it's finished.
12. Make a note of the **Service Name** and the **Schema Owner**; you will need them when are running the ODI Studio later on in this tutorial. When you have noted them down, click Create to continue.
13. On the Summary screen, click Close.

You now have the tablespaces and schemas needed to support Oracle Data Integrator Repositories in the database.

Updating the Clustered Oracle Java Cloud Instance Domain

Steps to update the clustered Oracle Java Cloud Instance domain. on which you have installed Oracle Data Integrator Cloud.

Before you begin updating the clustered Oracle Java Cloud Service Domain, use the WebLogic Administration Console to shutdown all Managed Servers and the Administration Server.

1. Access the Administration Server Console by using the following link:

    ```
    http://admin_server_host:admin_server_port/console
    ```

    .

2. In the **Domain Structure**, expand [+] Environment and click **Servers**. Click the **Control** tab. Select the managed server (not the adminserver), and click **Shutdown > Force Shutdown Now**.

    Click Yes to continue. Wait a minute.

3. On the same panel, select the adminserver and shut it down in the same way. This also kills the Console.

    Click Yes.

    Just ignore the web browser for the moment.

4. To update your domain, use the **Configuration Wizard** to extend (update) the Java Cloud Service domain with ODI. Open a terminal session and enter the following commmands:

    ```
    cd /u01/app/oracle/middleware/oracle_common/common/bin
    ./config.sh
    ```

5. In the **Update Domain** form, select **Update an existing domain**. In the **Domain Location**, select the following WebLogic domain directory. The domain_name will automatically point to the default domain created when provisioning this JCS instance: u01/data/domains/domain_name. Click to continue on to the next screen.

6. In the **Templates** form, do the following:

    a. Select **Update Domain Using Product Templates**.

    b. Select the following items (selecting one will cause others to be selected with green checks):

        - Oracle Data Integrator - Agent
        - Oracle Data Integrator - Agent Libraries
        - Oracle Data Integrator SDK Shared Library Template
        - Oracle Data Integrator - Console
        - Oracle Data Integrator – Standalone Collocated Agent
        - Oracle Enterprise Manager Plugin for ODI

        The following products should already be selected, and greyed out and in italics. They were selected when you created the domain in your Java Cloud Service instance:

   – Basic WebLogic Server Domain
   – Oracle Enterprise Manager
   – Oracle WSM Policy Manager
   – Oracle JRF
   – WebLogic Coherence Cluster Extension

   Click Next.

7. The next screen is the for configuring the database type that the Repository Creation Utility requires for internal use. Again, the installer can fill in the details for you.

   Click Get RCU configuration, then press Next.

8. On the **Datasources** form, click the checkbox next to your ODI repositories configuration details. The installer will fill in the rest of the details for you. Then click Next.

9. The **JDBC Test** screen is used to make sure that your database connections are working correctly. Select your database connection and then press Test Selected Connections. When the test has completed, press Next.

10. In the **Credentials** form, enter the ODI Supervisor username and password for the SUPERVISOR key. (The password for the supervisor should be the variable you entered in the Repository Creation Utility section). Then add a new domain key.

    a. Click **Add** to add a new credential.

    b. In **Key Name**, enter the name of this domain as the key. This is the domain created for the JCS.

    c. In **Username** and **Password**, provide the Administrator user's Username and case-sensitive Password.

    d. In **Store Name**, select oracle.odi.credmap from the pull-down.

    Click Next.

11. In the **Advanced Configuration** form, select **Topology**, **System Components** and **Deployments and Services**. On many of the panels you will just accept the defaults.

12. In the **Managed Servers** form, update the required managed servers.

    a. To avoid targeting related libraries and applications to the default Oracle Data Integrator server instead of the Oracle Data Integrator cluster, delete the default Oracle Data Integrator server from the list (you don't physically delete it) by selecting **ODI_server1** and click Delete.

    b. By default, the remaining managed server will be pre-populated. You can either rename it or use it as it is. For the purpose of this documentation, the Managed Server has been left with the default entry, however select Enable SSL if it is unchecked.

    c. Make a note of the Listen Port number, you will need this later on when creating an Agent in Oracle Data Integrator

    Click Next.

13. In the **Clusters** form, just note the Cluster Name. It will be the first eight letters of your domain name (which is the same as the first eight letters of your JCS instance name) and the suffix of _cluster. WARNING: If you try to change the

Cluster Name, it appears to work but will generate a Coherence error later, do not change the cluster name. Click Next.

14. Leave the **Server Templates** settings as they are and click Next.
15. Leave the **Dynamic Servers** settings as they are and click Next.
16. In the **Assign Servers to Clusters** form, ensure that the servers are assigned to the cluster. Click Next.
17. In the **Coherence Clusters** form, accept the defaults and click Next.
18. In the **Machines** form, click the **Unix Machine** tab to verify that it is populated. Give the machine a suitable name. Leave all other fields at their default values. Click Next.
19. In the **Assign Servers to Machines** form, ensure that the servers are assigned as shown in the image below. then click Next.

    Click Next.
20. Leave the **Virtual Targets** settings as they are and click Next.
21. Do not change anything on the **Partitions** screen. Click Next.
22. Do not change anything on the **System Components** page. This screen is used to configure a standalone collocated agent which can be optionally defined in the Topology. Click Next.
23. Check server listen address. On the **ODI Configuration** page, set a SUPERVISOR password for Oracle Data Integrator standalone collocated agent. This screen is used to configure a standalone collocated agent which can be optionally defined in the Topology. (Although we are not going to use a collocated agent, it is sensible to specify a password and make a note if it.)
24. Do not change anything on the **Assign System Components** screen. Click Next.
25. The ODI applications are deployed to the cluster. Check that they have been deployed correctly. On the right side under Targets, confirm that you can see odiconsole and oraclediagent under your cluster > Application.

    By default, the ODI libraries are under the Admin Server. Move them into the cluster and out of Admin Server (these are two separate steps).

    a. In the **Deployments Targeting** form, on the left side under **Deployments > Library**, select oracle.odi-agent#2.0@12.2.1.2.6 and oracle.odi-sdk#2.0@12.2.1.2.6.
    b. On the right side under **Targets**, select the cluster name (your cluster name will be different).
    c. Click the middle **>** to move the items from the left to the right. They will go in alphabetically under Library.

    Do not click Next yet.

    Last you move the Libraries out of the Administration Server:

    a. On the right side under **Targets**, under **Server > domain_adminserver > Library** (your adminserver name will be different), select oracle.odi-agent#2.0@12.2.1.2.6 and oracle.odi-sdk#2.0@12.2.1.2.6.
    b. Click the middle **<** to move the items from the right. They don't actually go to the left, they just go away from the right.

    Now click Next.

26. Confirm that odiMasterRepository and odiWorkRepository are in the right side under **Targets/cluster/JDBC/JDBC System Resource**.
27. In the **Configuration Summary** form, leave the defaults and click Update.

    You can safely ignore the warning about No system component instance.
28. Click Next once the domain is configured, then click Finish.

    You have successfully updated the JCS domain to support ODI JEE Agents.

Creating Agents in the Master Repository Using Oracle Data Integrator Studio

Steps to create physical and logical Oracle Data Integrator agents

If there is no physical agent entry for an agent, then the agent startup will fail when starting the WLS Administration Server.

To create an agent using Oracle Data Integrator Studio, see Creating an Agent in the Master Repository Using ODI Studio in *Installing and Configuring Oracle Data Integrator*.

Note:

Running Oracle Data Integrator Studio on-premises on your local laptop or desktop is not supported as it may slow down the user interface to the point of being unusable. You should use VNC connected to the JCS instance to run Oracle Data Integrator Studio.

1. Start ODI Studio from the command line by using the following commands:

```
cd /u01/app/oracle/middleware/odi/studio
./odi.sh
```

2. You will be asked if you would like to import preferences from a previous ODI installation. Click No.
3. Log in to the Repository you created earlier using the RCU.
   a. Create a Login by clicking on **Connect to Repository** and then click the green plus.
   b. Fill in all of the connection information that match the entries you made in the RCU.

Field	Description
Login Name	Choose a suitable login name for the connection.
User	This is the name you picked when creating the tablespaces in the Repository Creation Utility. .
Password	The password you created for the SUPERVISOR

Field	Description
User	The user name created for the master repository in Repository Creation Utility. You should already have made a note of this value earlier on. .
Password	The password for the master repository user.
Driver List	The list contains the JDBC drivers supported by the ODI Studio. Select Oracle JDBC driver.
Driver Name	The name of the JDBC driver will be filled in when you select from the Driver List.
URL	This is the location of the database given as a standard JDBC connection string.

c. Select **Work Repository** and click Browse to display the Work Repository selection window.

d. Select WORKREP and click OK.

e. Click Test. (If the selection worked, then the Test should work…)

f. If successful, click OK

g. Enter a Wallet password twice. Click OK.

h. All the information should be saved and pre-populated. Just click OK and the **Designer**, **Operator**, and **Topology** tabs should appear on the left.

4. Create a Physical Agent. On the **Topology** tab, expand **Physical Architecture**, right-click **Agents**, and select **New Agent**.

   Fill in the information for for Oracle Data Integrator J2EE agent:

   - **Name** — This is unique name given for the agent, and it must match the name of the J2EE agent application running on your server.
   - **Host** — The IP address, or host name, of the server that the J2EE agent is running on.
   - **Port** — The port of the server that the J2EE agent is running on. This is the same value (9073 in our example) as the listen port set in step 12 of Updating the Java Cloud Service Domain section.

   Click Save.

5. Create a Logical Agent. On the **Topology** tab, expand **Logical Architecture**, right-click **Agents**, and select **New Agent**.

6. For simplicity, keep all the names the same: OracleDIAgent. Click Save. Close the OracleDIAgent tab.

7. Optionally, you can repeat steps 4, 5 and 6, but for the standalone collocated agent.

   Create a Physical Agent. On the **Topology** tab, expand **Physical Architecture**, right-click **Agents**, and select **New Agent**.

   Fill in the information for for Oracle Data Integrator standalone collocated agent:

   - **Name** — This is unique name given for the agent, and it must match the name of the standalone collocated agent application running on your server.

- **Host** — The IP address, or host name, of the server that the standalone collocated agent is running on.
- **Port** — This is the same value as the server listen port, 20910, set in step 23 of Updating the Java Cloud Service Domain section.

Click Save.

8. Create a Logical Agent. On the **Topology** tab, expand **Logical Architecture**, right-click **Agents**, and select **New Agent**.
9. For simplicity, keep all the names the same: OracleDIAgent1. Click Save. Close the OracleDIAgent1 tab.

   ODI is now provisioned on the JCS instance. The agents are running and available for ad hoc and scheduled work.

Starting the Administration Server

Once you've created the physical and logical agents in Oracle Data Integrator, you need to re-start the WLS Administration server.

The Administration Server normally does not host any user applications, it simply controls the other Managed Servers. Once the Managed Servers are running, the Administration Server only collects statistics and logs

Weblogic servers in the cloud environment are usually monitored and controlled by the Node Manager, so it is better to use the Node Manager to restart the Administration Server.

1. From your **vncviewer** console, open a new terminal window and run the Weblogic Scripting Tool using the following commands:

```
cd /u01/app/oracle/middleware/oracle_common/common/bin
./wlst.sh
```

   The command prompt should change to show that you are now in the WLST environment.

2. Now connect to the Node manager using the **nmConnect** command.

```
nmConnect('weblogic_admin_user', 'admin_password', 'cloud_service_ip', 'port',
'domain','domain_path')
```

   where

Parameter	Meaning
weblogic_admin_user	is the administrator user name for your Oracle Cloud Service
admin_password	is the administrator password.
cloud_service_ip	is the ip address of your Oracle Java Cloud service instance
port	is the listener port for the Node Manager. This is usually 5556
domain	is the name of your Weblogic domain. This can be found in the service details page of the Oracle Java Cloud service console.

Parameter	Meaning
domain_path	is the fully qualified path where the domain resides on your virtual machine.

For example:

```
nmConnect('weblogic', 'gTY78Hcv', '10.10.10.10', '5556', 'MyJCSODI_domain','/u01/
data/domains/MyJCSODI_domain')
```

Note:

All the parameters are enclosed in single quotes.

3. Now you are connected to the Node Manager, you can start your administration server. Use the following command:

   ```
   nmStart('MyJCSODI_adminserver')
   ```

   Remember to replace the parameter with the name of your own administration server domain.

Post-Configuration Tasks

Steps to update Oracle Coherence parameters

Use the WLS Administration Console to update the Oracle Coherence parameters. To configure Oracle Coherence:

1. Log into the Oracle WebLogic Server Administration Console.
2. In the **Domain Structure** panel, expand the **Environment** node. Click **Coherence Clusters**.
3. In the **Coherence Clusters** table, click **DataGridConfig**.
4. Select the **Configuration** tab and ensure that the **Cluster Listen port** has been set. If there is no value then click Lock & Edit and set **Cluster Listen port** to 9000. (If this port is in use then you may have to change it later on).
5. Click Save , then change to the **Members** tab. Select **Part of the cluster** under the **Clusters** section, and then select your managed server.

   Now click Save and Activate Changes.

The post-configuration tasks pertaining to Coherence membership are done. When you start the servers (next section), you may need to come back here to adjust the Coherence membership again.

Starting the Managed Servers

To get your agents up and running, use these steps to start the Managed Servers that were created during domain extension.

Start (or if necessary restart) the servers. Use the command line script to restart the Admin Server, and use the WebLogic Server console to start the Managed Servers. The Admin Server may already be running!

1. You can now access the WLS Administration Server Console by using the following link:

   ```
   http://admin_server_host:admin_server_port/console
   ```

2. In the **Domain Structure** panel, expand the **Environment** node. Click **Servers**.
3. Click the **Control** tab, select your managed server (click the check box, not the server name), click Start. (This assumes your node manager is running.)
4. Click Yes to continue. Wait a minute...
5. The agent should now be deployed in JCS. You can see it by going to **Domain Structure > Deployments** then in Summary of Deployments panel click **Deployment Order** twice to get it sorted low-to-high. You should see oraclediagent near the top, and it should be State=Active and Health=OK.
6. Go back to ODI Studio to test the agent. In **Topology > Physical Architecture > Agents > OracleDIAgent**, click Test.
7. If you chose to configure the standalone collocated agent, then you can test it by repeating step 6, but for **OracleDIAgent1**.

   Note:

   This Agent would need to be started first, which you do from the command line. For more information on how to do this, see: section 7.5 - Starting a Standalone or Standalone Collocated Agent Using Node Manager - of the Oracle Fusion Middleware Installing and Configuring Oracle Data Integrator.

The WLS admin server and managed servers are now running. All deployed applications are running.

ODI is now provisioned on the JCS instance. The agents are running and available for ad hoc and scheduled work.

Part III
Integrating Data

Understand Oracle Data Integrator and how to use it.

Topics:

9

Understanding Oracle Data Integrator Concepts

This chapter provides an introduction to the main concepts of Oracle Data Integrator. This chapter includes the following topics:

Introduction to Declarative Design

To design an integration process with conventional ETL systems, a developer needs to design each step of the process: Consider, for example, a common case in which sales figures must be summed over time for different customer age groups. The sales data comes from a sales management database, and age groups are described in an age distribution file. In order to combine these sources then insert and update appropriate records in the customer statistics systems, you must design each step, which includes:

1. Load the customer sales data in the engine
2. Load the age distribution file in the engine
3. Perform a lookup between the customer sales data and the age distribution data
4. Aggregate the customer sales grouped by age distribution
5. Load the target sales statistics data into the engine
6. Determine what needs to be inserted or updated by comparing aggregated information with the data from the statistics system
7. Insert new records into the target
8. Update existing records into the target

This method requires specialized skills, depending on the steps that need to be designed. It also requires significant efforts in development, because even repetitive succession of tasks, such as managing inserts/updates in a target, need to be developed into each task. Finally, with this method, maintenance requires significant effort. Changing the integration process requires a clear understanding of what the process does as well as the knowledge of how it is done. With the conventional ETL method of design, the logical and technical aspects of the integration are intertwined. Declarative Design is a design method that focuses on "What" to do (the Declarative Rules) rather than "How" to do it (the Process). In our example, "What" the process does is:

- Relate the customer age from the sales application to the age groups from the statistical file
- Aggregate customer sales by age groups to load sales statistics

"How" this is done, that is the underlying technical aspects or technical strategies for performing this integration task – such as creating temporary data structures or calling loaders – is clearly separated from the declarative rules.

Declarative Design in Oracle Data Integrator uses the well known relational paradigm to declare in the form of a mapping the declarative rules for a data integration task, which includes designation of sources, targets, and transformations.

Declarative rules often apply to metadata to transform data and are usually described in natural language by business users. In a typical data integration project (such as a Data Warehouse project), these rules are defined during the specification phase in documents written by business analysts in conjunction with project managers. They can very often be implemented using SQL expressions, provided that the metadata they refer to is known and qualified in a metadata repository.

Declarative rules are implemented using Mappings, which connect sources to targets through a flow of components such as Join, Filter, Aggregate, Set, Split, and so on.

Table 9-1 gives examples of declarative rules.

Table 9-1   Examples of declarative rules

Declarative Rule	Type	SQL Expression
Sum of all amounts or items sold during October 2005 multiplied by the item price	Aggregate	`SUM( CASE WHEN SALES.YEARMONTH=200510 THEN SALES.AMOUNT*PRODUCT.ITEM_PRICE ELSE 0 END )`
Products that start with 'CPU' and that belong to the hardware category	Filter	`Upper(PRODUCT.PRODUCT_NAME)like 'CPU%' And PRODUCT.CATEGORY = 'HARDWARE'`
Customers with their orders and order lines	Join	`CUSTOMER.CUSTOMER_ID = ORDER.CUSTOMER_ID And ORDER.ORDER_ID = ORDER_LINE.ORDER_ID`
Reject duplicate customer names	Unique Key Constraint	`Unique key (CUSTOMER_NAME)`
Reject orders with a link to an non-existent customer	Reference Constraint	`Foreign key on ORDERS(CUSTOMER_ID) references CUSTOMER(CUSTOMER_ID)`

Introduction to Knowledge Modules

Knowledge Modules (KM) implement "how" the integration processes occur. Each Knowledge Module type refers to a specific integration task:

- Reverse-engineering metadata from the heterogeneous systems for Oracle Data Integrator (RKM). See the Creating and Using Data Models and Datastores section in *Developing Integration Projects with Oracle Data Integrator* for more information on how to use the RKM.

- Handling Changed Data Capture (CDC) on a given system (JKM). See the Using Journalizing section in *Developing Integration Projects with Oracle Data Integrator* for more information on how to use the Journalizing Knowledge Modules.
- Loading data from one system to another, using system-optimized methods (LKM). These KMs are used in mappings. See the Creating and Using Mappings section in *Developing Integration Projects with Oracle Data Integrator* for more information on how to use the Loading Knowledge Modules.
- Integrating data in a target system, using specific strategies (insert/update, slowly changing dimensions) (IKM). These KMs are used in mappings. See the Creating and Using Mappings section in *Developing Integration Projects with Oracle Data Integrator* for more information on how to use the Integration Knowledge Modules.
- Controlling Data Integrity on the data flow (CKM). These KMs are used in data model's static check and mappings flow checks. See the Creating and Using Data Models and Datastores and Creating and Using Mappings sections in *Developing Integration Projects with Oracle Data Integrator* for more information on how to use the Check Knowledge Modules.
- Exposing data in the form of web services (SKM). See the Creating and Using Data Services section in *Administering Oracle Data Integrator* for more information on how to use the Service Knowledge Modules.

A Knowledge Module is a code template for a given integration task. This code is independent of the Declarative Rules that need to be processed. At design-time, a developer creates the Declarative Rules describing integration processes. These Declarative Rules are merged with the Knowledge Module to generate code ready for runtime. At runtime, Oracle Data Integrator sends this code for execution to the source and target systems it leverages in the E-LT architecture for running the process.

Knowledge Modules cover a wide range of technologies and techniques. Knowledge Modules provide additional flexibility by giving users access to the most-appropriate or finely tuned solution for a specific task in a given situation. For example, to transfer data from one DBMS to another, a developer can use any of several methods depending on the situation:

- The DBMS loaders (Oracle's SQL*Loader, Microsoft SQL Server's BCP, Teradata TPump) can dump data from the source engine to a file then load this file to the target engine
- The database link features (Oracle Database Links, Microsoft SQL Server's Linked Servers) can transfer data directly between servers

These technical strategies amongst others correspond to Knowledge Modules tuned to exploit native capabilities of given platforms.

Knowledge modules are also fully extensible. Their code is open and can be edited through a graphical user interface by technical experts willing to implement new integration methods or best practices (for example, for higher performance or to comply with regulations and corporate standards). Without having the skill of the technical experts, developers can use these custom Knowledge Modules in the integration processes.

For more information on Knowledge Modules, see the *Connectivity and Modules Guide for Oracle Data Integrator* and the *Knowledge Module Developer's Guide for Oracle Data Integrator*.

Introduction to Mappings

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 will be used to generate the integration process.

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
- an API that returns data in the form of an array of records

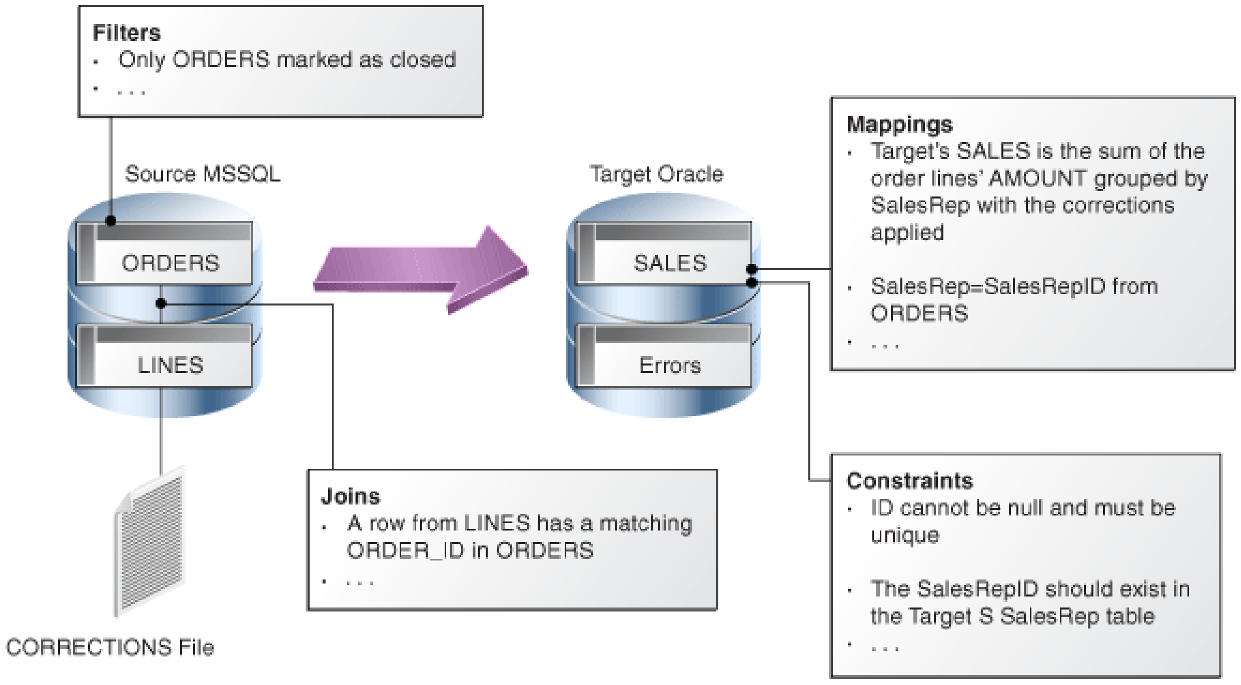
Regardless of the underlying technology, all data sources appear in Oracle Data Integrator in the form of datastores that can be manipulated and integrated in the same way. The datastores are grouped into data models. These models contain all the declarative rules –metadata - attached to datastores such as constraints.

Declarative Rules

The declarative rules that make up a mapping can be expressed in human language, as shown in the following example: Data is coming from two Microsoft SQL Server tables (ORDERS joined to LINES) and is combined with data from the CORRECTIONS file. The target SALES Oracle table must match some constraints such as the uniqueness of the ID column and valid reference to the SALES_REP table.

Data must be transformed and aggregated according to some mappings expressed in human language as shown in Figure 9-1.

Figure 9-1 Example of a business problem


Filters
Only ORDERS marked as closed
...
Source MSSQL
ORDERS
LINES
CORRECTIONS File
Target Oracle
SALES
Errors
Mappings
Target's SALES is the sum of the order lines' AMOUNT grouped by SalesRep with the corrections applied
SalesRep=SalesRepID from ORDERS
...
Joins
A row from LINES has a matching ORDER_ID in ORDERS
...
Constraints
ID cannot be null and must be unique
The SalesRepID should exist in the Target S SalesRep table
...

Translating these business rules from natural language to SQL expressions is usually straightforward. In our example, the rules that appear in Figure 9-1 could be translated as shown in Table 9-2.

Table 9-2 Business rules translated

Type	Rule	SQL Expression/Constraint
Filter	Only ORDERS marked as closed	`ORDERS.STATUS = 'CLOSED'`
Join	A row from LINES has a matching ORDER_ID in ORDERS	`ORDERS.ORDER_ID = LINES.ORDER_ID`
Aggregate	Target's SALES is the sum of the order lines' AMOUNT grouped by sales rep, with the corrections applied	`SUM(LINES.AMOUNT + CORRECTIONS.VALUE)`
Mapping	Sales Rep = Sales Rep ID from ORDERS	`ORDERS.SALES_REP_ID`
Constraint	ID must not be null	ID is set to `not null` in the data model
Constraint	ID must be unique	A unique key is added to the data model with (ID) as set of columns
Constraint	The Sales Rep ID should exist in the Target SalesRep table	A reference (foreign key) is added in the data model on `SALES.SALES_REP = SALES_REP.SALES_REP_ID`

Implementing this business problem using Oracle Data Integrator is a very easy and straightforward exercise. It is done by simply translating the business rules into a mapping.

Data Flow

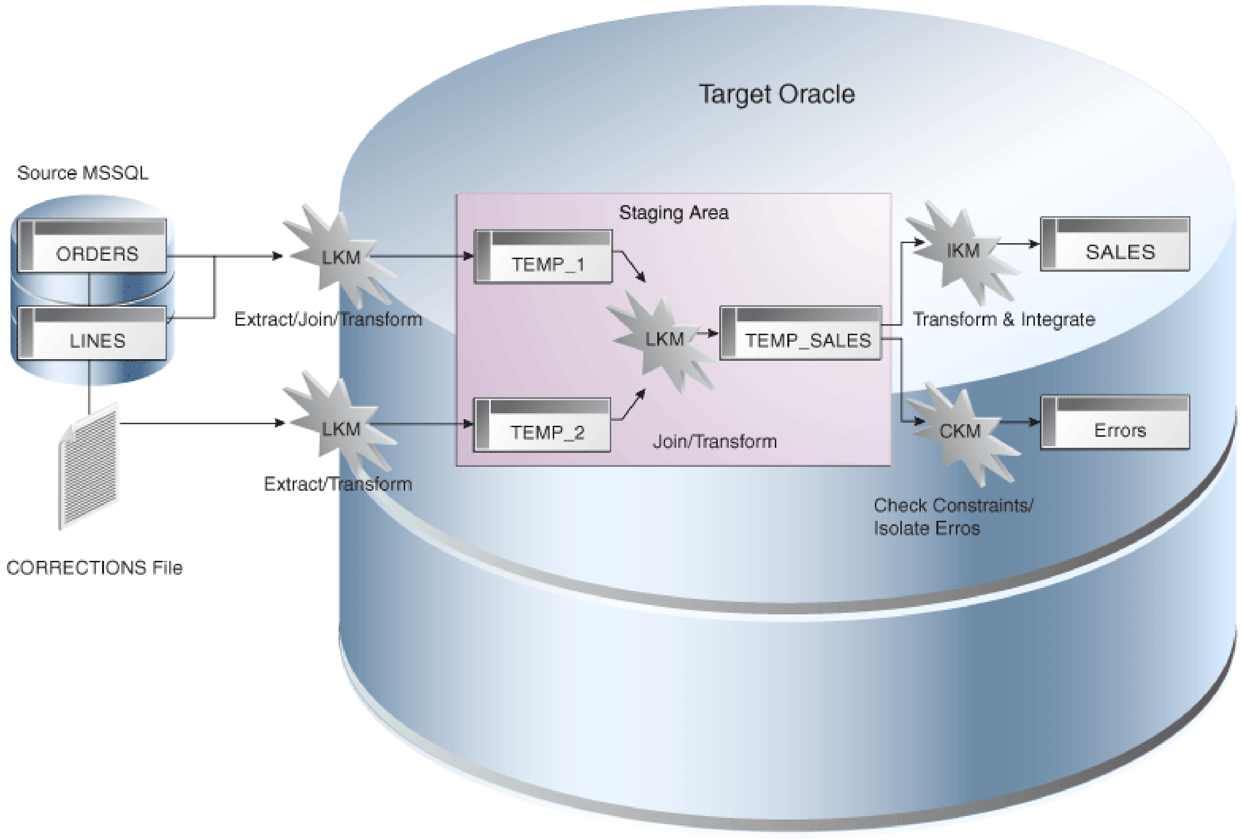
Business rules defined in the mapping are automatically converted into a data flow that will carry out the joins, filters, mappings, and constraints from source data to target tables.

By default, Oracle Data Integrator will use 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 Oracle Data Integrator creates its temporary objects and executes some of the rules (mapping, joins, final filters, aggregations etc.). When performing the operations this way, Oracle Data Integrator uses an E-LT strategy as it first extracts and loads the temporary tables and then finishes the transformations in the target RDBMS.

In some particular cases, when source volumes are small (less than 500,000 records), this staging area can be located in memory in Oracle Data Integrator's in-memory relational database – In-Memory Engine. Oracle Data Integrator would then behave like a traditional ETL tool.

Figure 9-2 shows the data flow automatically generated by Oracle Data Integrator to load the final SALES table. The business rules will be transformed into code by the Knowledge Modules (KM). The code produced will generate several steps. Some of these steps will extract and load the data from the sources to the staging area (Loading Knowledge Modules - LKM). Others will transform and integrate the data from the staging area to the target table (Integration Knowledge Module - IKM). To ensure data quality, the Check Knowledge Module (CKM) will apply the user defined constraints to the staging data to isolate erroneous records in the Errors table.

Figure 9-2 Oracle Data Integrator Knowledge Modules in action



Oracle Data Integrator Knowledge Modules contain the actual code that will be executed by the various servers of the infrastructure. Some of the code contained in the Knowledge Modules is generic. It makes calls to the Oracle Data Integrator Substitution API that will be bound at run-time to the business-rules and generates the final code that will be executed.

At design time, declarative rules are defined in the mappings and Knowledge Modules are only selected and configured.

At run-time, code is generated and every Oracle Data Integrator API call in the Knowledge Modules (enclosed by <% and %>) is replaced with its corresponding object name or expression, with respect to the metadata provided in the Repository. The generated code is orchestrated by Oracle Data Integrator run-time component - the Agent – on the source and target systems to make them perform the processing, as defined in the E-LT approach.

See the Creating and Using Mappings section in *Developing Integration Projects with Oracle Data Integrator* for more information on how to work with mappings.

10

Typical Integration Projects

Oracle Data Integrator provides a wide range of integration features. This chapter introduces the most typical ODI Integration Projects.
This chapter includes the following topics:

- Design-time Projects
- Batch-Oriented Integration
- Event-Oriented Integration
- Service-Oriented Architecture
- Data Quality with ODI

Design-time Projects

A typical project is composed of several steps and milestones.

Some of these are:

- Define the business needs
- Identify and declare the sources and targets in the Topology
- Design and Reverse-engineer source and target data structures in the form of data models
- Implement data quality rules on these data models and perform static checks on these data models to validate the data quality rules
- Develop mappings using datastores from these data models as sources and target
- Develop additional components for tasks that cannot be achieved using mappings, such as Receiving and sending e-mails, handling files (copy, compress, rename and such), executing web services
- Integrate mappings and additional components for building Package workflows
- Version your work and release it in the form of scenarios
- Schedule and operate scenarios.

Oracle Data Integrator will help you cover most of these steps, from source data investigation to metadata lineage, and through loading and data quality audit. With its repository, Oracle Data Integrator will centralize the specification and development efforts and provide a unique architecture on which the project can rely to succeed.

The Overview of an Integration Project section in *Developing Integration Projects with Oracle Data Integrator* introduces you to the basic steps of creating an integration project with Oracle Data Integrator. The Creating an Integration Project section in *Developing Integration Projects with Oracle Data Integrator* gives you more detailed information on the several steps of creating an integration project in ODI.

Batch-Oriented Integration

ODI is a comprehensive data integration platform with a built-in connectivity to all major databases, Big Data clusters, data warehouse and analytic applications providing high-volume and high-performance batch integration.

The main goal of a data warehouse is to consolidate and deliver accurate indicators to business users to help them make decisions regarding their everyday business. A typical project is composed of several steps and milestones. Some of these are:

- Defining business needs (Key Indicators)
- Identifying source data that concerns key indicators; specifying business rules to transform source information into key indicators
- Modeling the data structure of the target warehouse to store the key indicators
- Populating the indicators by implementing business rules
- Measuring the overall accuracy of the data by setting up data quality rules
- Developing reports on key indicators
- Making key indicators and metadata available to business users through adhoc query tools or predefined reports
- Measuring business users' satisfaction and adding/modifying key indicators

With its repository, ODI will centralize the specification and development efforts and provide a unique architecture on which the project can rely to succeed.

Scheduling and Operating Scenarios

Scheduling and operating scenarios is usually done in the Test and Production environments in separate Work Repositories. Any scenario can be scheduled by an ODI Agent or by any external scheduler, as scenarios can be invoked by an operating system command.

When scenarios are running in production, agents generate execution logs in an ODI Work Repository. These logs can be monitored either through the Operator Navigator or through any web browser when Oracle Data Integrator Console is setup. Failing jobs can be restarted and ad-hoc tasks submitted for execution.

E-LT

ODI uses a unique E-LT architecture that leverages the power of existing RDBMS and Big Data engines by generating native SQL and bulk loader control scripts to execute all transformations.

Event-Oriented Integration

Capturing events from a Message Oriented Middleware or an Enterprise Service Bus has become a common task in integrating applications in a real-time environment. Applications and business processes generate messages for several subscribers, or they consume messages from the messaging infrastructure.

Oracle Data Integrator includes technology to support message-based integration and that complies with the Java Message Services (JMS) standard. For example, a transformation job within Oracle Data Integrator can subscribe and source messages from any message queue or topic. Messages are captured and transformed in real time and then written to the target systems.

Other use cases of this type of integration might require capturing changes at the database level. Oracle Data Integrator Changed Data Capture (CDC) capability identifies and captures inserted, updated, or deleted data from the source and makes it available for integration processes.

ODI provides two methods for tracking changes from source datastores to the CDC framework: triggers and RDBMS log mining. The first method can be deployed on most RDBMS that implement database triggers. This method is optimized to minimize overhead on the source systems. For example, changed data captured by the trigger is not duplicated, minimizing the number of input/output operations, which slow down source systems. The second method involves mining the RDBMS logs—the internal change history of the database engine. Log-based CDC is supported using Oracle GoldenGate.

The CDC framework used to manage changes, based on Knowledge Modules, is generic and open, so the change-tracking method can be customized. Any third-party change provider can be used to load the framework with changes.

Changes frequently involve several data sources at the same time. For example, when an order is created, updated, or deleted, both the orders table and the order lines table are involved. When processing a new order line, it is important that the new order, to which the line is related, is taken into account too. ODI provides a mode of change tracking called Consistent Set CDC. This mode allows for processing sets of changes for which data consistency is guaranteed.

For example, incoming orders can be detected at the database level using CDC. These new orders are enriched and transformed by ODI before being posted to the appropriate message queue or topic. Other applications such as Oracle BPEL or Oracle Business Activity Monitoring can subscribe to these messages, and the incoming events will trigger the appropriate business processes.

For more information on how to use the CDC framework in ODI, see the Using Journalizing section in *Developing Integration Projects with Oracle Data Integrator*.

Service-Oriented Architecture

Oracle Data Integrator can be integrated seamlessly in a Service-Oriented Architecture (SOA) in several ways:

Data Services are specialized Web services that provide access to data stored in database tables. Coupled with the Changed Data Capture capability, data services can also provide access to the changed records for a given subscriber. Data services are automatically generated by Oracle Data Integrator and deployed as Web services to a Web container, usually a Java application server. For more information on how to set up, generate and deploy data services, see the Creating and Using Data Services section in *Administering Oracle Data Integrator*.

Oracle Data Integrator can also expose its transformation processes as Web services to enable applications to use them as integration services. For example, a LOAD_SALES batch process used to update the CRM application can be triggered as a Web service from any service-compliant application, such as Oracle BPEL, Oracle Enterprise Service Bus, or Oracle Business Activity Monitoring. Transformations developed using ODI can therefore participate in the broader Service Oriented Architecture initiative.

Third-party Web services can be invoked as part of an ODI workflow and used as part of the data integration processes. Requests are generated on the fly and responses processed through regular transformations. Suppose, for example, that your company subscribed to a third-party service that exposes daily currency exchange rates as a Web service. If you want this data to update your multiple currency data warehouse, ODI automates this task with a minimum of effort. You would simply invoke the Web service from your data warehouse workflow and perform any appropriate transformation to the incoming data to make it fit a specific format. For more information on how to use web services in ODI, see the Using Web Services section in *Developing Integration Projects with Oracle Data Integrator*.

Data Quality with ODI

With an approach based on declarative rules, Oracle Data Integrator is the most appropriate tool to help you build a data quality framework to track data inconsistencies.

Oracle Data Integrator uses declarative data integrity rules defined in its centralized metadata repository. These rules are applied to application data to guarantee the integrity and consistency of enterprise information. The Data Integrity benefits add to the overall Data Quality initiative and facilitate integration with existing and future business processes addressing this particular need.

Oracle Data Integrator automatically retrieves existing rules defined at the data level (such as database constraints) by a reverse-engineering process. ODI also allows developers to define additional, user-defined declarative rules that may be inferred from data discovery and profiling within ODI, and immediately checked.

Oracle Data Integrator provides a built-in framework to check the quality of your data in two ways:

- Check data in your data servers, to validate that this data does not violate any of the rules declared on the datastores in Oracle Data Integrator. This data quality check is called a static check and is performed on data models and datastores. This type of check allows you to profile the quality of the data against rules that are not enforced by their storage technology.
- Check data while it is moved and transformed by a mapping, in a flow check that checks the data flow against the rules defined on the target datastore. With such a check, correct data can be integrated into the target datastore while incorrect data is automatically moved into error tables.

Both static and flow checks are using the constraints that are defined in the datastores and data models, and both use the Check Knowledge Modules (CKMs). For more information see the Flow and Static Control section in *Developing Integration Projects with Oracle Data Integrator*.

These checks check the integrity of the data and validate constraints. For advanced data quality projects (for example for name and address cleansing projects) as well as advanced data profiling, the Oracle Enterprise Data Quality products can be used along with Oracle Date Integrator.

Tip:

Oracle Enterprise Data Quality (EDQ) can augment ODI's Data Quality capabilities. For more information about EDQ, refer to the Oracle Enterprise Data Quality documentation.

11

Setting up a Topology

This chapter describes how to set up the topology in Oracle Data Integrator Cloud.

This chapter includes the following sections:

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

the Overview of Oracle Data Integrator Topology section in {Varref: ODIDG}

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 *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 *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:

   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`
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 Balancing for 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 11-1 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

Table 11-1 (Cont.) Working with Big Data

Task	Documentation
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 Oracle Data Integrator
- 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 Oracle Data Integrator
- 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.

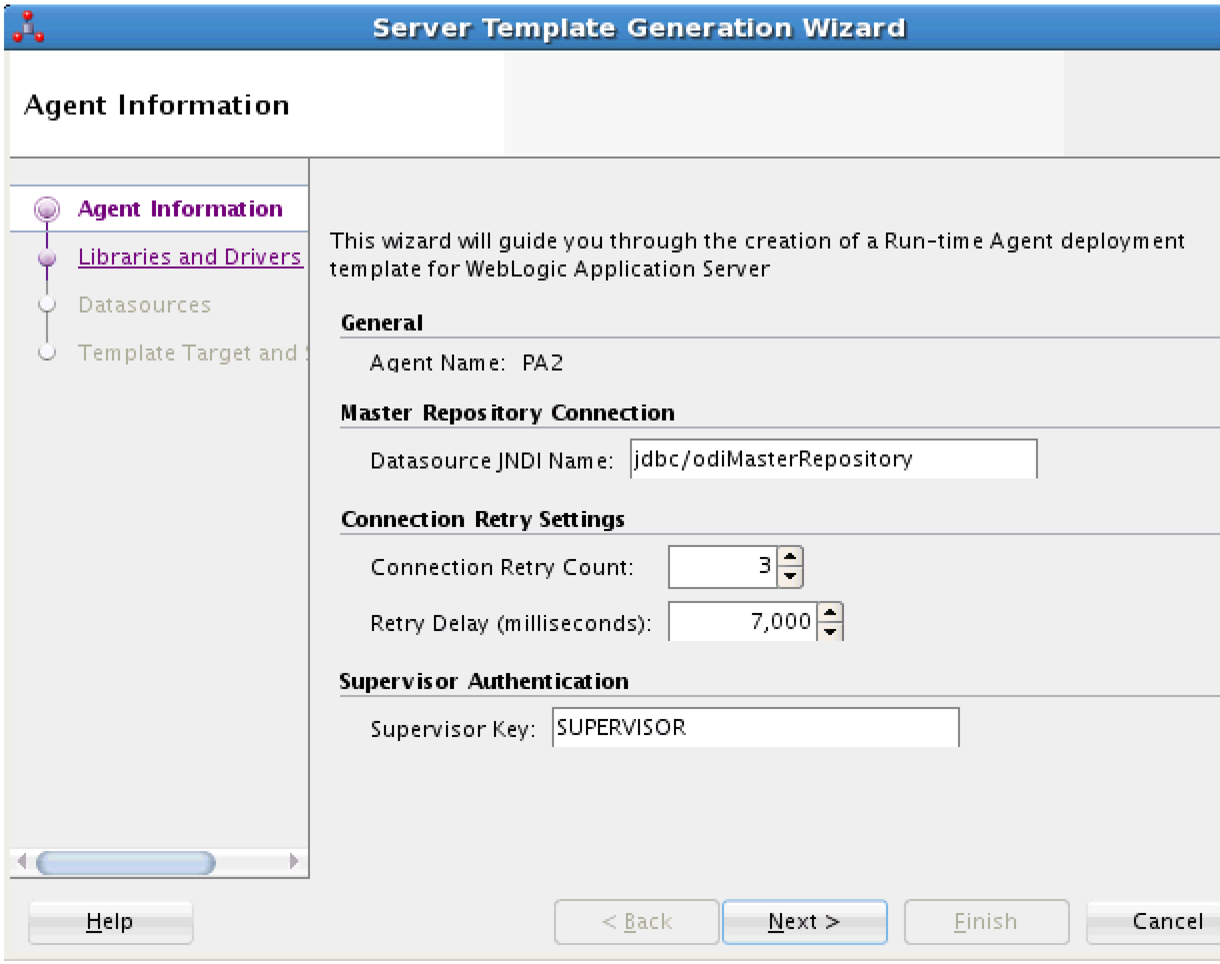
> **Note:**
>
> To use the generate server template feature, ODI Studio must be installed using the Enterprise Installation option. For more information on ODI Studio and installation types, see the Installing Oracle Data Integrator chapter in *Installing and Configuring Oracle Data Integrator*.

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 11-1.

   > **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 11-1 Server Template Generation Wizard - Agent Information Tab



This wizard will guide you through the creation of a Run-time Agent deployment template for WebLogic Application Server

General

Agent Name: PA2

Master Repository Connection

Datasource JNDI Name: jdbc/odiMasterRepository

Connection Retry Settings

Connection Retry Count: 3

Retry Delay (milliseconds): 7,000

Supervisor Authentication

Supervisor Key: SUPERVISOR

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.

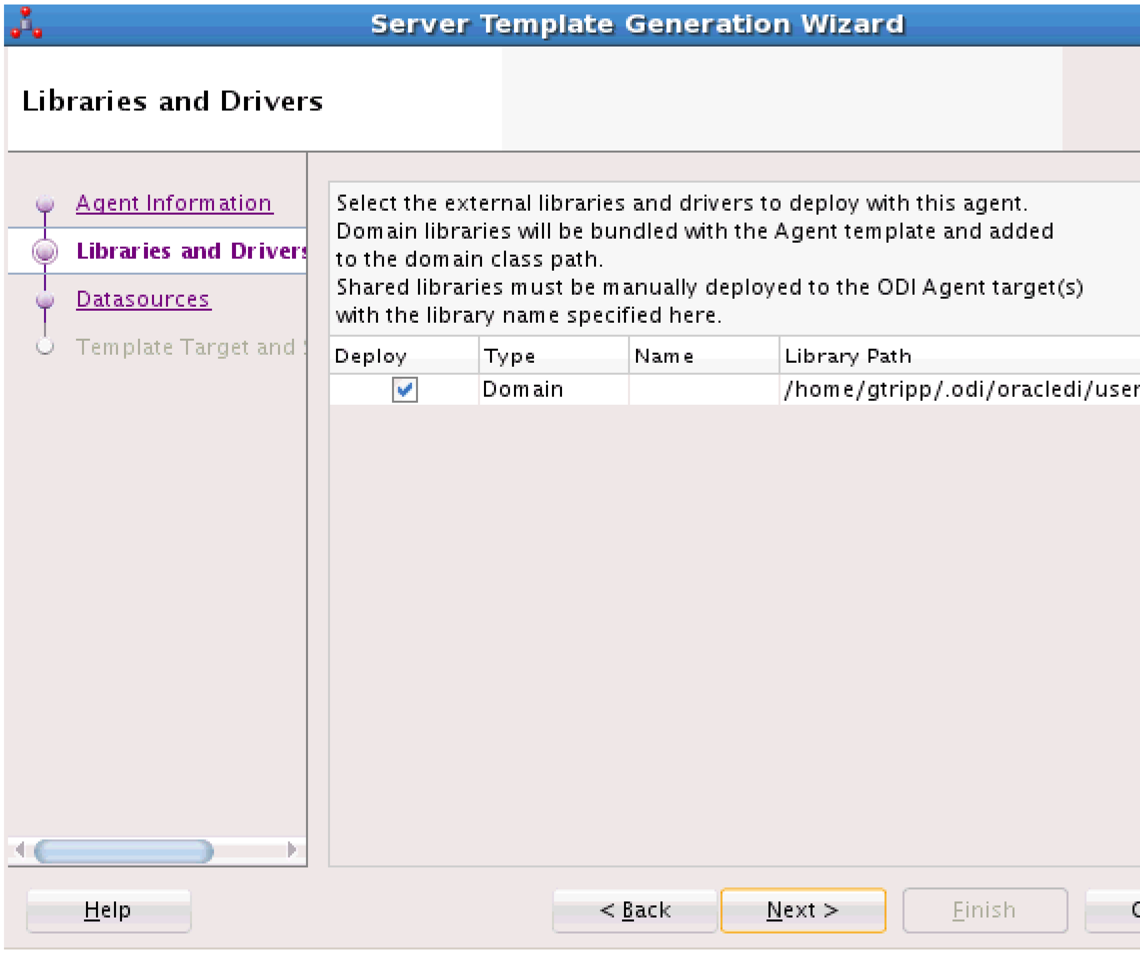
     Retry Delay (milliseconds): Interval (in milliseconds) between each connection retry attempt.

- **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 11-2.

Figure 11-2 Server Template Generation Wizard - Libraries and Drivers Tab



Select the external libraries and drivers to deploy with this agent.
Domain libraries will be bundled with the Agent template and added to the domain class path.
Shared libraries must be manually deployed to the ODI Agent target(s) with the library name specified here.

Deploy	Type	Name	Library Path
✓	Domain		/home/gtripp/.odi/oracledi/user

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.

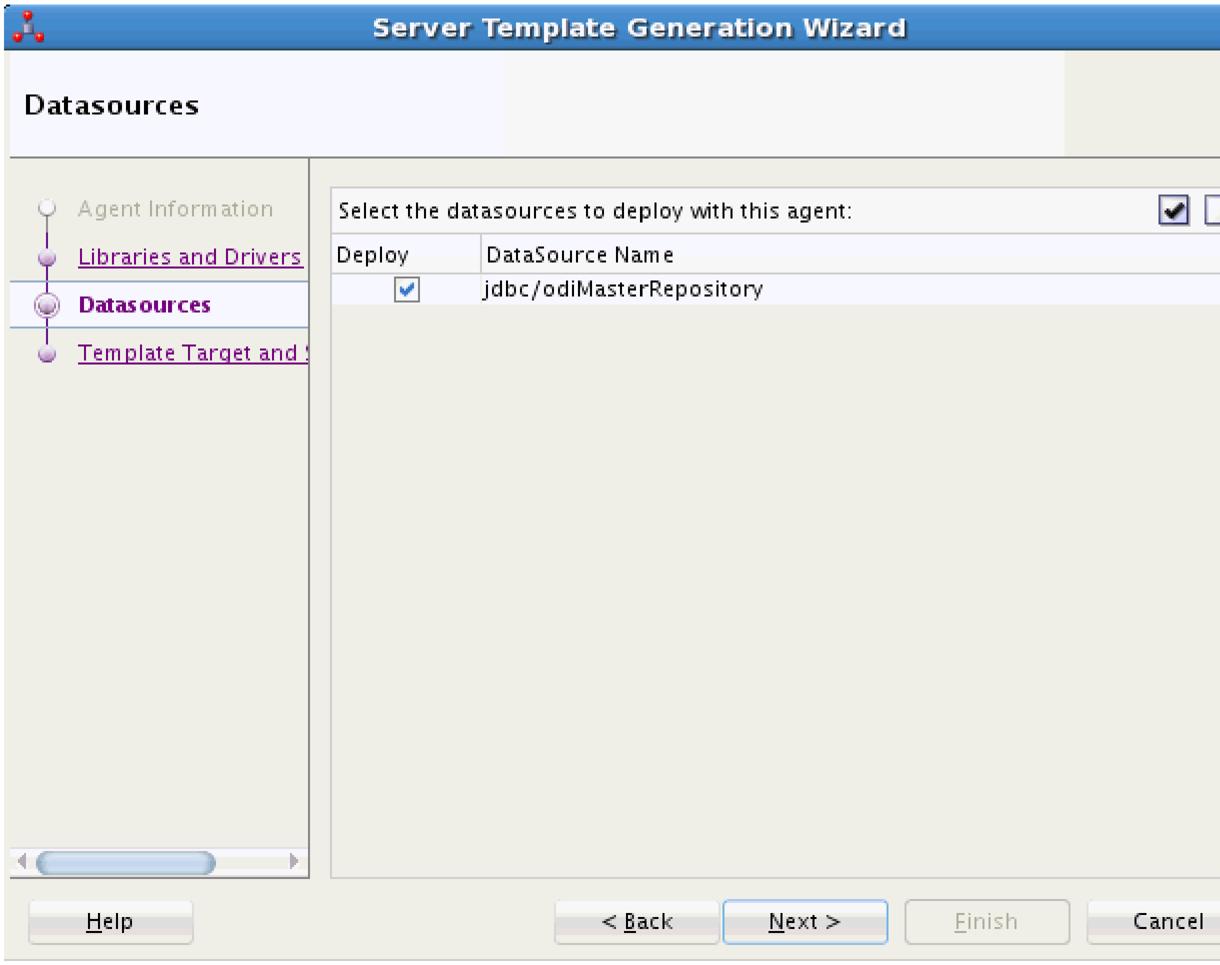
Note:

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 11-3.

Figure 11-3 Server Template Generation Wizard - Datasources Tab



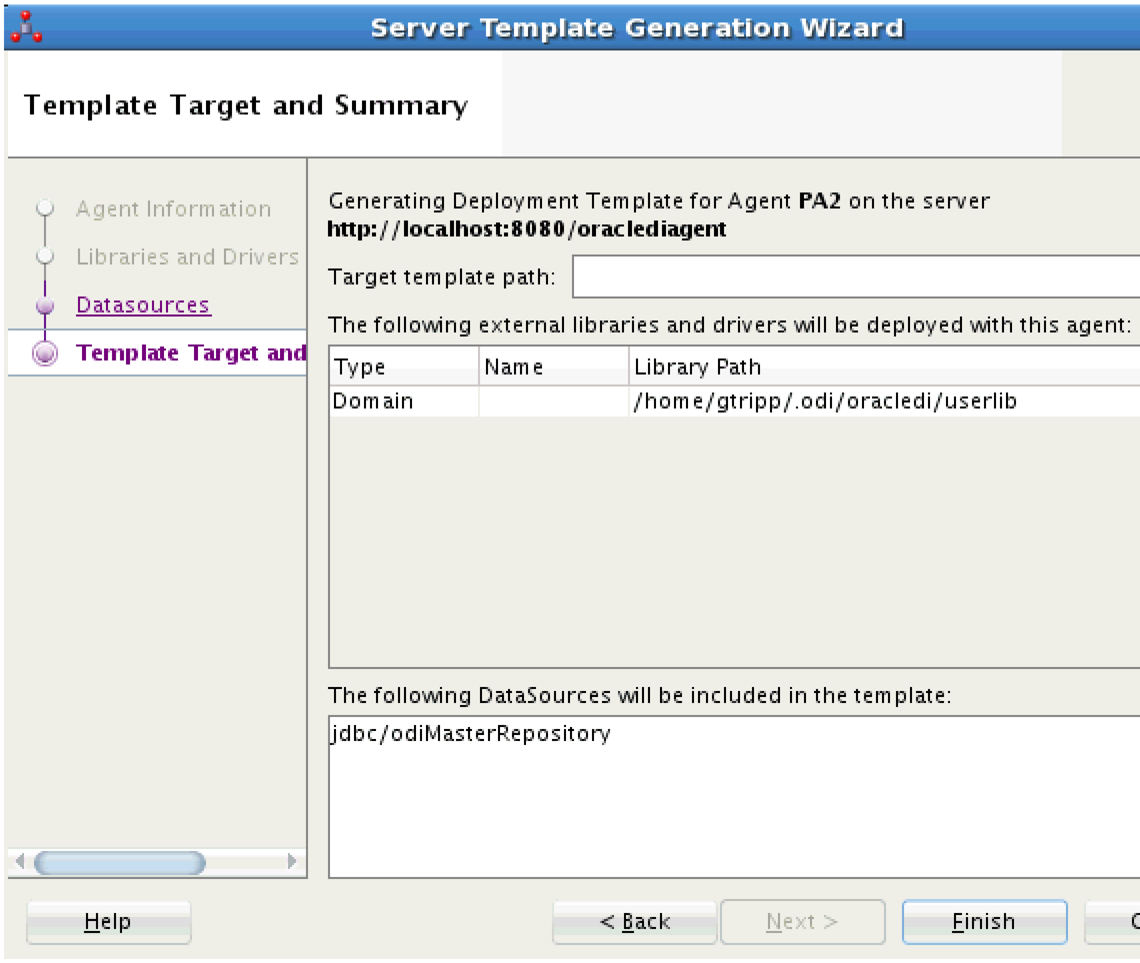
Select the datasources to deploy with this agent:

Deploy	DataSource Name
✓	jdbc/odiMasterRepository

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 11-4.

Figure 11-4 Server Template Generation Wizard - Template Target and Summary Tab



Generating Deployment Template for Agent **PA2** on the server **http://localhost:8080/oraclediagent**

Target template path:

The following external libraries and drivers will be deployed with this agent:

Type	Name	Library Path
Domain		/home/gtripp/.odi/oracledi/userlib

The following DataSources will be included in the template:

jdbc/odiMasterRepository

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 11-1 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 11-2 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 Agent for 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.

12

Creating and Using Data Models and Datastores

This chapter describes how to create a model, how to reverse-engineer this model to populate it with datastores and how to create manually datastores of a model. This chapter also explains how to use partitioning and check the quality of the data in a model.
This chapter includes the following sections:

Introduction to 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, etc.). Models in Data Integrator only contain *Metadata*, that is the description of the data structures. They do not contain a copy of the actual data.

Note:

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

Models can be organized into model folders and the datastores of a model can be organized into sub-models. The section Organizing Models with Folders in *Developing Integration Projects with Oracle Data Integrator* describes how to create and organize model folders and sub-models.

Datastores

A datastore represents a data structure. It can be a table, a flat file, a message queue or any other data structure accessible by Oracle Data Integrator.

A datastore describes data in a tabular structure. Datastores are composed of attributes.

As datastores are based on the relational paradigm, it is also possible to associate the following elements to a datastore:

- **Keys**

  A Key is a set of attributes with a specific role in the relational paradigm. Primary and Alternate Keys identify each record uniquely. Non-Unique Indexes enable optimized record access.
- **References**

  A Reference is a functional link between two datastores. It corresponds to a Foreign Key in a relational model. For example: The INVOICE datastore references the CUSTOMER datastore through the customer number.
- **Conditions and Filters**

  Conditions and Filters are a WHERE-type SQL expressions attached to a datastore. They are used to validate or filter the data in this datastore.

Data Integrity

A model contains constraints such as Keys, References or Conditions, but also non-null flags on attributes. Oracle Data Integrator includes a data integrity framework for ensuring the quality of a data model.

This framework allows to perform:

- **Static Checks** to verify the integrity of the data contained in a data model. This operation is performed to assess the quality of the data in a model when constraints do not physically exist in the data server but are defined in Data Integrator only.
- **Flow Check** to verify the integrity of a data flow before it is integrated into a given datastore. The data flow is checked against the constraints defined in Oracle Data Integrator for the datastore that is the target of the data flow.

Reverse-engineering

A new model is created with no datastores. Reverse-engineering is the process that populates the model in Oracle Data Integrator by retrieving metadata from the data server containing the data structures. There are two different types of reverse-engineering:

- **Standard reverse-engineering** uses standard JDBC driver features to retrieve the metadata. Note that unique keys are not reverse-engineered when using a standard reverse-engineering.
- **Customized reverse-engineering** uses a technology-specific Reverse Knowledge Module (RKM) to retrieve the metadata, using a method specific to the given technology. This method is recommended if a technology specific RKM exists because it usually retrieves more information than the Standard reverse-engineering method. See the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* for a list of available RKMs.

Other methods for reverse-engineering exist for flat file datastores. They are detailed in Reverse-Engineering File Datastores.

Oracle Data Integrator is able to reverse-engineer models containing datastore shortcuts. For more information, see the Using Shortcuts chapter in *Developing Integration Projects with Oracle Data Integrator*.

Changed Data Capture

Change Data Capture (CDC), also referred to as *Journalizing*, allows to trap changes occurring on the data. CDC is used in Oracle Data Integrator to eliminate the transfer of unchanged data. This feature can be used for example for data synchronization and replication.

Journalizing can be applied to models, sub-models or datastores based on certain type of technologies.

For information about setting up Changed Data Capture, see the Using Journalizing chapter in *Developing Integration Projects with Oracle Data Integrator*.

Creating and Reverse-Engineering a Model

This section explains the following topics:

- Creating a Model
- Creating a Model and Topology Objects
- Reverse-engineering a Model

Creating a Model

A Model is a set of datastores corresponding to data structures contained in a Physical Schema.

Tip:

To create a new model and new topology objects at the same time, use the procedure described in Creating a Model and Topology Objects.

To create a Model:

1. In Designer Navigator expand the **Models** panel.
2. Right-click then select **New Model**.
3. Fill in the following fields in the Definition tab:
   - **Name**: Name of the model used in the user interface.
   - **Technology**: Select the model's technology.
   - **Logical Schema**: Select the Logical Schema on which your model will be based.
4. On the Reverse Engineer tab, select a **Context** which will be used for the model's reverse-engineering.

Note that if there is only one context that maps the logical schema, this context will be set automatically.

5. Select **Save** from the File main menu.

The model is created, but contains no datastore yet.

Creating a Model and Topology Objects

You can create a Model along with other topology objects, including new data servers, contexts and schemas, at the same time:

1. In ODI Studio, select **File** and click **New...**.
2. In the **New Gallery** dialog, select **Create a New Model and Topology Objects** and click **OK**.

   The **Create New Model and Topology Objects** wizard appears.
3. In the **Model** panel of the wizard, fill in the following fields.
   - **Name**: Name of the model used in the user interface.
   - **Technology**: Select the model's technology.
   - **Logical Schema**: Select the Logical Schema on which your model will be based.
   - **Context**: Select the context that will be used for the model's reverse-engineering.
4. Click **Next**.
5. In the **Data Server** panel of the wizard, fill in the following data server fields.
   - **Name**: Name of the data server, as it appears in the user interface.

     Note: or naming data servers, it is recommended to use the following naming standard: <TECHNOLOGY_NAME>_<SERVER_NAME>.
   - **Technology**: Technology linked to the data server.

     Note: Appears only if the **Technology** selected for the Model is of the type Generic SQL.
   - **User**: User name used for connecting to the data server.
   - **Password**: Password linked with the user name.

     Note: This password is stored encrypted in the repository.
   - **Driver List:** Provides a list of available drivers available to be used with the data server.
   - **Driver:** Name of the driver used for connecting to the data server.
   - **URL:** Provides the connection details.
   - **Properties:** Lists the properties that you can set for the selected driver.
6. Click **Next**.
7. In the **Physical Schema** panel of the wizard, fill in the physical schema fields.
   - **Name**: Name of the physical schema. It is calculated automatically and is read-only.
   - **Datasource (Catalog)**: Name of the catalog in the data server.

  Note: Appears only if the **Technology** selected supports Catalogs.

- **Schema (Schema)**: Name of the schema in the data server. Schema, owner, or library where the required data is stored.

  Note: Oracle Data Integrator lists all the schemas present in the data server. Sometimes, Oracle Data Integrator cannot draw up this list. In this case, you should enter the schema name, respecting the case.

- **Datasource (Work Catalog)**: Indicate the catalog in which you want to create these objects. For some data validation or transformation operations, Oracle Data Integrator may require work objects to be created.

  Note: Appears only if the **Technology** selected supports Catalogs.

- **Schema (Work Schema)**: Indicates the schema in which you want to create these objects. For some data validation or transformation operations, Oracle Data Integrator may require work objects to be created. If you do not set a **Work Schema**, it defaults to the **Schema** during execution

  It is recommended that you create a specific schema dedicated to any work tables. By creating a schema named SAS or ODI in all your data servers, you ensure that all Oracle Data Integrator activity remains totally independent from your applications.

  Note: Oracle Data Integrator lists all the schemas present in the data server. Sometimes, Oracle Data Integrator cannot draw up this list. In this case, you should enter the schema name, respecting the case.

- **Driver:** Name of the driver used for connecting to the data server.
- **URL:** Provides the connection details.
- **Properties:** Specifies the properties for the selected driver.

8. Click **Next**.
9. Click **Finish**. The model and topology objects are created, but the model contains no datastore yet.

Reverse-engineering a Model

To automatically populate datastores into the model you need to perform a reverse-engineering for this model.

Standard Reverse-Engineering

A Standard Reverse-Engineering uses the capacities of the JDBC driver used to connect the data server to retrieve the model metadata.

To perform a Standard Reverse- Engineering:

1. In the **Reverse Engineer** tab of your Model:
   - Select **Standard**.
   - Select the **Context** used for the reverse-engineering
   - Select the **Types of objects to reverse-engineer**. Only object of these types will be taken into account by the reverse-engineering process.
   - Enter in the **Mask** field the mask of tables to reverse engineer. The mask selects the objects to reverse. This mask uses the SQL LIKE syntax. The

percent (`%`) symbol means zero or more characters, and the underscore (_) symbol means one character.

- Optionally, you can specify the **characters to remove for the table alias**. These are the characters to delete in order to derive the alias. Note that if the datastores already exist, the characters specified here will not be removed from the table alias. Updating a datastore is not applied to the table alias.

2. In the **Selective Reverse-Engineering** tab select **Selective Reverse-Engineering**, **New Datastores**, **Existing Datastores** and **Objects to Reverse Engineer**.
3. A list of datastores to be reverse-engineered appears. Leave those you wish to reverse-engineer checked.
4. Select **Save** from the File main menu.
5. Click **Reverse Engineer** in the Model toolbar menu.
6. Oracle Data Integrator launches a reverse-engineering process for the selected datastores. A progress bar indicates the progress of the reverse-engineering process.

The reverse-engineered datastores appear under the model node in the **Models** panel.

Customized Reverse-Engineering

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.

For example, for the Oracle technology, the RKM Oracle accesses the database dictionary tables to retrieve the definition of tables, attributes, keys, etc., that are created in the model.

Note:

The RKM must be available as a global RKM or imported into the project. For more information on KM import, see the Creating an Integration Project chapter in *Developing Integration Projects with Oracle Data Integrator*.

To perform a Customized Reverse-Engineering using a RKM:

1. In the **Reverse Engineer** tab of your Model:
   - Select **Customized**.
   - Select the **Context** used for the reverse-engineering
   - Select the **Types of objects to reverse-engineer**. Only object of these types will be taken into account by the reverse-engineering process.
   - Enter in the **Mask** the mask of tables to reverse engineer.
   - Select the KM that you want to use for performing the reverse-engineering process. This KM is typically called `RKM <technology>.<name of the project>`.
   - Optionally, you can specify the **characters to remove for the table alias**. These are the characters to delete in order to derive the alias. Note that if the

datastores already exist, the characters specified here will not be removed from the table alias. Updating a datastore is not applied to the table alias.

2. Click **Reverse Engineer** in the Model toolbar menu, then **Yes** to validate the changes.
3. Click **OK.**
4. The **Session Started Window** appears.
5. Click **OK**.

You can review the reverse-engineering tasks in the Operator Navigator. If the reverse-engineering process completes correctly, reverse-engineered datastores appear under the model node in the **Models** panel.

Creating and Reverse-Engineering a Datastore

Although the recommended method for creating datastores in a model is reverse-engineering, it is possible to manually define datastores in a blank model. It is the recommended path for creating flat file datastores.
This section explains how to create a datastore.

Creating a Datastore

To create a datastore:

1. From the Models tree in Designer Navigator, select a Model or a Sub-Model.
2. Right-click and select **New Datastore**.
3. In the **Definition** tab, fill in the following fields:
   - **Name of the Datastore**: This is the name that appears in the trees and that is used to reference the datastore from a project.

     Note:

     Do not use ODI reserved names like, for example, `JRN_FLAG`, `JRN_SUBSCRIBER`, and `JRN_DATE` for the datastore name. These names would cause Duplicate Attribute name SQL errors in ODI intermediate tables such as error tables.

   - **Resource Name**: Name of the object in the form recognized by the data server which stores it. This may be a table name, a file name, the name of a JMS Queue, etc.

> **Note:**
>
> If the Resource is a database table, it must respect the Database rules for object and attributes identifiers. There is a limit in the object identifier for most of the Technologies (in Oracle, typically 30 characters). To avoid these errors, ensure in the topology for a specific technology that maximum lengths for the object names (tables and columns) correspond to your database configuration.

- **Alias**: This is a default alias used to prefix this datastore's attributes names in expressions.

4. If the datastore represents a flat file (delimited or fixed), in the **File** tab, fill in the following fields:
   - **File Format**: Select the type of your flat file, fixed or delimited.
   - **Header**: Number of header lines for the flat file.
   - **Record Separator** and **Field Separator** define the characters used to separate records (lines) in the file, and fields within one record.

     Record Separator: One or several characters separating lines (or records) in the file:
     – MS-DOS: DOS carriage return
     – Unix: UNIX carriage return
     – Other: Free text you can input as characters or hexadecimal codes

     Field Separator: One ore several characters separating the fields in a record.
     – Tabulation
     – Space
     – Other: Free text you can input as characters or hexadecimal code
5. Select **Save** from the File main menu.

The datastore is created. If this is a File datastore, refer to Reverse-Engineering File Datastores for creating attributes for this datastore. It is also possible to manually edit attributes for all datastores. See Adding and Deleting Datastore Attributes for more information.

Reverse-Engineering File Datastores

Oracle Data Integrator provides specific methods for reverse-engineering flat files. The methods for reversing flat files are described below.

Adding and Deleting Datastore Attributes

To add attributes to a datastore:

1. In the **Attributes** tab of the datastore, click **Add Attribute** in the tool bar menu.
2. An empty line appears. Fill in the information about the new attribute. You should at least fill in the **Name**, **Datatype** and **Length** fields.
3. Repeat steps 1 and 2 for each attribute you want to add to the datastore.

4. Select **Save** from the File main menu.

To delete attributes from a datastore:

1. In the **Attributes** tab of the datastore, select the attribute to delete.
2. Click the **Delete Attribute** button. The attribute disappears from the list.

Adding and Deleting Constraints and Filters

Oracle Data Integrator manages constraints on data model including Keys, References, Conditions and Mandatory Attributes. It includes a data integrity framework for ensuring the quality of a data model based on these constraints.

Filters are not constraints but are defined similarly to Conditions. A Filter is not used to enforce a data quality rule on a datastore, but is used to automatically filter this datastore when using it as a source.

Editing and Viewing a Datastore's Data

To view a datastore's data:

1. Select the datastore from the model in the Designer Navigator.
2. Right-click and select **View Data**.

The data appear in a non editable grid.

To edit a datastore's data:

1. Select the datastore from the model in the Designer Navigator.
2. Right-click and select **Data...**

The data appear in an editable grid in the Data Editor. The **Refresh** button enables you to edit and run again the query returning the datastore data. You can filter the data and perform free queries on the datastore using this method.

It is possible to edit a datastore's data if the connectivity used and the data server user's privileges allow it, and if the datastore structure enables to identify each row of the datastore (PK, etc.).

Note:

The data displayed is the data stored in the physical schema corresponding to the model's logical schema, in the current working context.

Using Partitioning

Oracle Data Integrator is able to use database-defined partitions when processing data in partitioned tables used as source or targets of mappings. These partitions are created in the datastore corresponding to the table, either through the reverse-engineering process or manually. For example with the Oracle technology, partitions are reverse-engineered using the RKM Oracle.

The partitioning methods supported depend on the technology of the datastore. For example, for the Oracle technology the following partitioning methods are supported: Range, Hash, List.

Once defined on a datastore, partitions can be selected when this datastore is used as a source or a target of a mapping.

Refer to Creating and Using Mappings, for information.

If using the common format designer, you can also create partitions when performing the Generate DDL operation.

Manually Defining Partitions and Sub-Partitions of Model Datastores

Partition information can be reverse-engineered along with the datastore structures or defined manually.

Note:

Standard reverse-engineering does not support the revers-engineering of partitions. To reverse-engineer partitions and sub-partitions, you have to use customized reverse-engineering.

To manually define partitions and sub-partitions for a datastore:

1. In the Models accordion, double-click the datastore for which you want to define the partition or sub-partition. The Datastore Editor opens.
2. In the **Partitions** tab, enter the following details to define the partition and sub-partition:
   - **Partition by**

     Select the partitioning method. This list displays the partitioning methods supported by the technology on which the model relies.
   - **Sub-Partition by**

     If you want to define sub-partitions in addition to partitions, select the sub-partitioning method. This list displays the partitioning methods supported by the technology on which the model relies.
3. Click **Add Partition**.
4. In the **Name** field, enter a name for the partition, for example: `FY08`.
5. In the **Description** field, enter a description for the partition, for example: `Operations for Fiscal Year 08`.
6. If you want to add:
   - additional partitions, repeat steps 3 through 5.
   - a sub-partition of a selected partition, click **Add Sub-Partition** and repeat steps 4 and 5.
7. From the File menu, select **Save**.

Checking Data Quality in a Model

Data Quality control is essential in ensuring the overall consistency of the data in your information system's applications.

Application data is not always valid for the constraints and declarative rules imposed by the information system. You may, for instance, find orders with no customer, or order lines with no product, etc. In addition, such incorrect data may propagate via integration flows.

Introduction to Data Integrity

Oracle Data Integrator provides a working environment to detect these constraint violation and store them for recycling or reporting purposes.

There are two different main types of controls: **Static Control** and **Flow Control**. We will examine the differences between the two.

Static Control

Static Control implies the existence of rules that are used to verify the integrity of your application data. Some of these rules (referred to as constraints) may already be implemented in your data servers (using primary keys, reference constraints, etc.)

With Oracle Data Integrator, you can refine the validation of your data by defining additional constraints, without implementing them directly in your servers. This procedure is called **Static Control** since it allows you to perform checks directly on existing - or static - data. Note that the active database constraints (these are those that have **Defined in the Database** and **Active** selected on the Controls tab) need no additional control from Oracle Data Integrator since they are already controlled by the database.

Flow Control

The information systems targeted by transformation and integration processes often implement their own declarative rules. The **Flow Control** function is used to verify an application's incoming data according to these constraints before loading the data into these targets.

Setting up flow control is detailed in Creating and Using Mappings.

Checking a Constraint

While creating a constraint in Oracle Data Integrator, it is possible to retrieve the number of lines violating this constraint. This action, referred as **Synchronous Control** is performed from the **Control** tab of the given constraint Editor by clicking the **Check** button.

The result of a synchronous control is not persistent. This type of control is used to quickly evaluate the validity of a constraint definition.

Perform a Static Check on a Model, Sub-Model or Datastore

To perform a Static Check on a Model, Sub-Model or Datastore:

1. In the **Models** tree in the Designer Navigator, select the model that you want to check.
2. Double-click this model to edit it.
3. In the **Control** tab of the model Editor, select the Check Knowledge Module (CKM) used in the static check.
4. From the File menu, select **Save All**.
5. Right-click the model, sub-model or datastore that you want to check in the **Model** tree in the Designer Navigator and select **Control** > **Check**. Or, in the model editor menu bar, click the **Check Model** button.
6. In the **Run** dialog, select the execution parameters:
   - Select the **Context** into which the step must be executed.
   - Select the **Logical Agent** that will run the step.
   - Select a **Log Level**.
   - Check the **Delete Errors from the Checked Tables** option if you want rows detected as erroneous to be removed from the checked tables.
   - Select **Recurse Sub-Models** to check sub-models of this models
   - Optionally, select **Simulation**. This option performs a simulation of the run operation and generates a run report, without actually affecting data.

   See the Execution Parameters table in *Administering Oracle Data Integrator* for more information about the execution parameters.
7. Click **OK**.
8. The **Session Started Window** (or, if running a simulation, the **Simulation** window) appears.
9. Click **OK**.

You can review the check tasks in the Operator Navigator. If the control process completes correctly, you can review the erroneous records for each datastore that has been checked.

Reviewing Erroneous Records

To view a datastore's errors:

1. Select the datastore from the model in the Designer Navigator.
2. Right-click and select **Control > Errors...**.

The erroneous rows detected for this datastore appear in a grid.

13 Creating and Using Mappings

Learn how to create mappings to organize data sources, targets, and the transformations through which the data flows from source to target.

Topics

Introduction to Mappings

Mappings are the logical and physical organization of your data sources, targets, and the transformations through which the data flows from source to target. You create and manage mappings using the mapping editor, a new feature of ODI 12*c*.

The mapping editor opens whenever you open a mapping. Mappings are organized in folders under individual projects, found under Projects in the Designer Navigator.

Parts of a Mapping

A mapping is made up of and defined by the following parts:

- **Datastores**

  Data from source datastores is extracted by a mapping, and can be filtered during the loading process. Target datastores are the elements that are loaded by the mapping. Datastores act as Projector Components.

  Datastores that will be used as sources and targets of the loading process must exist in data models before you can use them in a mapping. See Creating and Using Data Models and Datastores for more information.

- **Datasets**

  Optionally, you can use datasets within mappings as 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. Datasets operate similarly to ODI 11*g* interfaces, and if you import 11*g* interfaces into ODI

12*c*, ODI will automatically create datasets based on your interface logic. Datasets act as Selector Components.

- **Reusable 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 (that is, reusable mappings may be nested). A reusable mapping can also include datastores as sources and targets itself, like other mapping components. Reusable mappings act as Projector Components.

- **Other Components**

  ODI provides additional components that are used in between sources and targets to manipulate the data. These components are available on the component palette in the mapping diagram.

  The following are the components available by default in the component palette:

  – Expression
  – Aggregate
  – Distinct
  – Set
  – Filter
  – Join
  – Lookup
  – Pivot
  – Sort
  – Split
  – Subquery Filter
  – Table Function
  – Unpivot

- **Connectors**

  Connectors create a flow of data between mapping components. Most components can have both input and output connectors. Datastores with only output connectors are considered sources; datastores with only input connectors are considered targets. Some components can support multiple input or output connectors; for example, the split component supports two or more output connectors, allowing you to split data into multiple downstream flows.

  – **Connector points** define the connections between components inside a mapping. A connector point is a single pathway for input or output for a component.
  – **Connector ports** are the small circles on the left and/or right sides of components displayed in the mapping diagram.

  In the mapping diagram, two components connected by a single visible line between their connector ports could have one or more connector points. The diagram only shows a single line to represent all of the connections between two components. You can select the line to show details about the connection in the property inspector.

- **Staging Schemas**

  Optionally, you can specify a staging area for a mapping or for a specific physical mapping design of a mapping. If you want to define a different staging area than any of the source or target datastores, you must define the correct physical and logical schemas in the mapping's execution context before creating a mapping. See the chapter Overview of Oracle Data Integrator Topology in *Developing Integration Projects with Oracle Data Integrator* for more information.

- **Knowledge Modules**

  Knowledge modules define how data will be transferred between data servers and loaded into data targets. Knowledge Modules (IKMs, LKMs, EKMs, and CKMs) that will be selected in the flow must be imported into the project or must be available as global Knowledge Modules.

  IKMs allow you to define (or specify) how the actual transformation and loading is performed.

  LKMs allow you to specify how the transfer of the data between one data server to another data server is performed.

  When used as flow control, CKMs allow you to check for errors in the data flow during the loading of records into a target datastore. When used as static control, CKMs can be used to check for any errors in a table. You can launch static control at any time on a model to see if the data satisfies constraints.

  You can select a strategy to perform these tasks by selecting an appropriate KM. For example, you can decide whether to use a JDBC to transfer data between two databases, or use an Oracle database link if the transfer is between two Oracle databases.

  See the chapter Creating an Integration Project in *Developing Integration Projects with Oracle Data Integrator* for more information.

- **Variables, Sequences, and User Functions**

  Variables, Sequences, and User Functions that will be used in expressions within your mappings must be created in the project. See the chapter Creating and Using Procedures, Variables, Sequences, and User Functions in *Developing Integration Projects with Oracle Data Integrator* for more information.

Navigating the Mapping Editor

The mapping editor provides a single environment for designing and editing mappings.

Mappings are organized within folders in a project in the Designer Navigator. Each folder has a mappings node, within which all mappings are listed.

To open the mapping editor, right-click an existing mapping and select **Open**, or double-click the mapping. To create a new mapping, right-click the **Mappings** node and select **New Mapping**. The mapping is opened as a tab on the main pane of ODI Studio. Select the tab corresponding to a mapping to view the mapping editor.

The mapping editor consists of the sections described in the table below:

Table 13-1 Mapping Editor Sections

Section	Location in Mapping Editor	Description
Mapping Diagram	Middle	The mapping diagram displays an editable logical or physical view of a mapping. These views are sometimes called the logical diagram or the physical diagram. You can drag datastores into the diagram from the Models tree, and Reusable Mappings from the Global Objects or Projects tree, into the mapping diagram. You can also drag components from the component palette to define various data operations.
Mapping Editor tabs	Middle, at the bottom of the mapping diagram	The Mapping Editor tabs are ordered according to the mapping creation process. These tabs are: • **Overview**: displays the general properties of the mapping • **Logical:** displays the logical organization of the mapping in the mapping diagram • **Physical**: displays the physical organization of the mapping in the mapping diagram
Property Inspector	Bottom	Displays properties for the selected object. If the Property Inspector does not display, select Properties from the Window menu.
Component Palette	Right	Displays the mapping components you can use for creating mappings. You can drag and drop components into the logical mapping diagram from the components palette. If the Component Palette does not display, select Components from the Window menu.
Structure Panel	Not shown	Displays a text-based hierarchical tree view of a mapping, which is navigable using the tab and arrow keys. The Structure Panel does not display by default. To open it, select Structure from the Window menu.
Thumbnail Panel	Not shown	Displays a miniature graphic of a mapping, with a rectangle indicating the portion currently showing in the mapping diagram. This panel is useful for navigating very large or complex mappings. The Thumbnail Panel does not display by default. To open it, select Thumbnail from the Window menu.

Creating a Mapping

Creating a mapping follows a standard process which can vary depending on the use case.

Using the logical diagram of the mapping editor, you can construct your mapping by dragging components onto the diagram, dragging connections between the components, dragging attributes across those connections, and modifying the properties of the components using the property inspector. When the logical diagram is complete, you can use the physical diagram to define where and how the integration process will run on your physical infrastructure. When the logical and physical design of your mapping is complete, you can run it.

The following step sequence is usually performed when creating a mapping, and can be used as a guideline to design your first mappings:

1. Creating a New Mapping
2. Adding and Removing Components
3. Connecting and Configuring Components
4. Defining a Physical Configuration
5. Running Mappings

Note:

You can also use the Property Inspector and the Structure Panel to perform the steps 2 to 5. See Editing Mappings Using the Property Inspector and the Structure Panel for more information.

Creating a New Mapping

To create a new mapping:

1. In Designer Navigator select the **Mappings** node in the folder under the project where you want to create the mapping.
2. Right-click and select **New Mapping**. The New Mapping dialog is displayed.
3. In the New Mapping dialog, fill in the mapping **Name**. Optionally, enter a **Description**. If you want the new mapping to contain a new empty dataset, select **Create Empty Dataset**. Click **OK**.

   Note:

   You can add and remove datasets (including this empty dataset) after you create a mapping. Datasets are entirely optional and all behavior of a dataset can be created using other components in the mapping editor.

   In ODI 12*c*, Datasets offer you the option to create data flows using the entity relationship method familiar to users of previous versions of ODI. In some cases creating an entity relationship diagram may be faster than creating a flow diagram, or make it easier and faster to introduce changes.

   When a physical diagram is calculated based on a logical diagram containing a Dataset, the entity relationships in the Dataset are automatically converted by ODI into a flow diagram and merged with the surrounding flow. You do not need to be concerned with how the flow is connected.

Your new mapping opens in a new tab in the main pane of ODI Studio.

Tip:

To display the editor of a datastore, a reusable mapping, or a dataset that is used in the Mapping tab, you can right-click the object and select **Open**.

Adding and Removing Components

Add components to the logical diagram by dragging them from the Component Palette. Drag datastores and reusable mappings from the Designer Navigator.

Delete components from a mapping by selecting them, and then either pressing the Delete key, or using the right-click context menu to select Delete. A confirmation dialog is shown.

Source and target datastores are the elements that will be extracted by, and loaded by, the mapping.

Between the source and target datastores are arranged all the other components of a mapping. When the mapping is run, data will flow from the source datastores, through the components you define, and into the target datastores.

Preserving and Removing Downstream Expressions

Where applicable, when you delete a component, a check box in the confirmation dialog allows you to preserve, or remove, downstream expressions; such expressions may have been created when you connected or modified a component. By default ODI preserves these expressions.

This feature allows you to make changes to a mapping without destroying work you have already done. For example, when a source datastore is mapped to a target datastore, the attributes are all mapped. You then realize that you need to filter the source data. To add the filter, one option is to delete the connection between the two datastores, but preserve the expressions set on the target datastore, and then connect a filter in the middle. None of the mapping expressions are lost.

Connecting and Configuring Components

Create connectors between components by dragging from the originating connector port to the destination connector port. Connectors can also be implicitly created by dragging attributes between components. When creating a connector between two ports, an attribute matching dialog may be shown which allows you to automatically map attributes based on name or position.

Attribute Matching

The Attribute Matching Dialog is displayed when a connector is drawn to a projector component (see the **Projector Components** section in Using Mapping Components) in the Mapping Editor. The Attribute Matching Dialog gives you an option to automatically create expressions to map attributes from the source to the target component based on a matching mechanism. It also gives the option to create new attributes on the target based on the source, or new attributes on the source based on the target.

This feature allows you to easily define a set of attributes in a component that are derived from another component. For example, you could drag a connection from a new, empty Set component to a downstream target datastore. If you leave checked the **Create Attributes On Source** option in the Attribute Matching dialog, the Set component will be populated with all of the attributes of the target datastore. When you connect the Set component to upstream components, you will already have the target attributes ready for you to map the upstream attributes to.

Connector Points and Connector Ports

Review the section on **Connectors** in Parts of a Mapping for an introduction to ODI connector terminology.

You can click a connector port on one component and drag a line to another component's connector port to define a connection. If the connection is allowed, ODI will either use an unused existing connector point on each component, or create an additional connector point as needed. The connection is displayed in the mapping diagram with a line drawn between the connector ports of the two connected components. Only a single line is shown even if two components have multiple connections between them.

Most components can use both input and output connectors to other components, which are visible in the mapping diagram as small circles on the sides of the component. The component type may place limitations on how many connectors of each type are allowed, and some components can have only input or only output connections.

Some components allow the addition or deletion of connector points using the property inspector.

For example, a Join component by default has two input connector points and one output connector point. It is allowed to have more than two inputs, though. If you drag a third connection to the input connector port of a join component, ODI creates a third input connector point. You can also select a Join component and, in the property inspector, in the Connector Points section, click the green plus icon to add additional Input Connector Points.

Note:

You cannot drag a connection to or from an input port that already has the maximum number of connections. For example, a target datastore can only have one input connector point; if you try to drag another connection to the input connector port, no connection is created.

You can delete a connector by right-clicking the line between two connector points and selecting **Delete**, or by selecting the line and pressing the Delete key.

Defining New Attributes

When you add components to a mapping, you may need to create attributes in them in order to move data across the flow from sources, through intermediate components, to targets. Typically you define new attributes to perform transformations of the data.

Use any of the following methods to define new attributes:

- **Attribute Matching Dialog**: This dialog is displayed in certain cases when dragging a connection from a connector port on one component to a connector port on another, when at least one component is a projector component.

  The attribute matching dialog includes an option to create attributes on the target. If target already has attributes with matching names, ODI will automatically map to these attributes. If you choose **By Position**, ODI will map the first attributes to existing attributes in the target, and then add the rest (if there are more) below it. For example, if there are three attributes in the target component, and the source has 12, the first three attributes map to the existing attributes, and then the remaining nine are copied over with their existing labels.
- **Drag and drop attributes**: Drag and drop a single (or multi-selected) attribute from a one component into another component (into a blank area of the component graphic, not on top of an existing attribute). ODI creates a connection (if one did not already exist), and also creates the attribute.

  > **Tip:**
  >
  > If the graphic for a component is "full", you can hover over the attributes and a scroll bar appears on the right. Scroll to the bottom to expose a blank line. You can then drag attributes to the blank area.
  >
  > If you drag an attribute onto another attribute, ODI maps it into that attribute, even if the names do not match. This does not create a new attribute on the target component.
- **Add new attributes in the property inspector**: In the property inspector, on the **Attributes** tab, use the green plus icon to create a new attribute. You can select or enter the new attribute's name, data type, and other properties in the **Attributes** table. You can then map to the new attribute by dragging attributes from other components onto the new attribute.

  > **Caution:**
  >
  > ODI will allow you to create an illegal data type connection. Therefore, you should always set the appropriate data type when you create a new attribute. For example, if you intend to map an attribute with a DATE data type to a new attribute, you should set the new attribute to have the DATE type as well.
  >
  > Type-mismatch errors will be caught during execution as a SQL error.

> **Note:**
>
> From ODI 12.2.1 onwards, when the DB2 TIME column is mapped to the target column, the target column displays only the time and omits the date.

Defining Expressions and Conditions

Expressions and conditions are used to map individual attributes from component to component. Component types determine the default expressions and conditions that will be converted into the underlying code of your mapping.

For example, any target component has an expression for each attribute. A filter, join, or lookup component will use code (such as SQL) to create the expression appropriate to the component type.

> **Tip:**
>
> When an expression is set on the target, any source attributes referenced by that expression are highlighted in magenta in the upstream sources. For example, an expression `emp.empno` on the target column `tgt_empno`, when `tgt_empno` is selected (by clicking on it), the attribute `empno` on the source datastore `emp` is highlighted.
>
> This highlighting function is useful for rapidly verifying that each desired target attribute has an expression with valid cross references. If an expression is manually edited incorrectly, such as if a source attribute is misspelled, the cross reference will be invalid, and no source attribute will be highlighted when clicking that target attribute.

You can modify the expressions and conditions of any component by modifying the code displayed in various property fields.

> **Note:**
>
> Oracle recommends using the expression editor instead of manually editing expressions in most cases. Selection of a source attribute from the expression editor will always give the expression a valid cross reference, minimizing editing errors. For more information, see The Expression Editor.

Expressions have a result type, such as VARCHAR or NUMERIC. The result type of conditions are boolean, meaning, the result of a condition should always evaluate to TRUE or FALSE. A condition is needed for filter, join, and lookup (selector) components, while an expression is used in datastore, aggregate, and distinct (projector) components, to perform some transformation or create the attribute-level mappings.

Every projector component can have expressions on its attributes. (For most projector components, an attribute has one expression, but the attribute of the Set component can have multiple expressions.) If you modify the expression for an attribute, a small "f" icon appears on the attribute in the logical diagram. This icon provides a visual cue that a function has been placed there.

To define the mapping of a target attribute:

1. In the mapping editor, select an attribute to display the attribute's properties in the Property Inspector.
2. In the **Target** tab (for expressions) or **Condition** tab (for conditions), modify the **Expression** or **Condition** field(s) to create the required logic.

   Tip:

   The attributes from any component in the diagram can be drag-and-dropped into an expression field to automatically add the fully-qualified attribute name to the code.

3. Optionally, select or hover over any field in the property inspector containing an expression, and then click the gear icon that appears to the right of the field, to open the advanced **Expression Editor**.

   The attributes on the left are only the ones that are in scope (have already been connected). So if you create a component with no upstream or downstream connection to a component with attributes, no attributes are listed.
4. Optionally, after modifying an expression or condition, consider validating your mapping to check for errors in your SQL code. Click the green check mark icon at the top of the logical diagram. Validation errors, if any, will be displayed in a panel.

Defining a Physical Configuration

In the **Physical** tab of the mapping editor, you define the loading and integration strategies for mapped data. Oracle Data Integrator automatically computes the flow depending on the configuration in the mapping's logical diagram. It proposes default knowledge modules (KMs) for the data flow. The **Physical** tab enables you to view the data flow and select the KMs used to load and integrate data.

For more information about physical design, see Physical Design.

Running Mappings

Once a mapping is created, you can run it. This section briefly summarizes the process of running a mapping. For detailed information about running your integration processes, see the Running Integration Processes chapter in *Administering Oracle Data Integrator*.

To run a mapping:

1. From the Projects menu of the Designer Navigator, right-click a mapping and select **Run**.

   Or, with the mapping open in the mapping editor, click the run icon in the toolbar. Or, select **Run** from the **Run** menu.
2. In the **Run** dialog, select the execution parameters:

- Select the **Context** into which the mapping must be executed. For more information about contexts, see the Contexts section in *Developing Integration Projects with Oracle Data Integrator*.
- Select the **Physical Mapping Design** you want to run. See Creating and Managing Physical Mapping Designs.
- Select the **Logical Agent** that will run the mapping. The object can also be executed using the agent that is built into Oracle Data Integrator Studio, by selecting Local (No Agent). For more information about logical agents, see the Agents section in *Developing Integration Projects with Oracle Data Integrator*.
- Select a **Log Level** to control the detail of messages that will appear in the validator when the mapping is run. For more information about logging, see the Managing the Log section in *Administering Oracle Data Integrator*.
- Check the **Simulation** box if you want to preview the code without actually running it. In this case no data will be changed on the source or target datastores. For more information, see the Simulating an Execution section in *Administering Oracle Data Integrator*.

3. Click **OK**.
4. The **Information** dialog appears. If your session started successfully, you will see "Session started."
5. Click **OK**.

Note:

- When you run a mapping, the Validation Results pane opens. You can review any validation warnings or errors there.
- You can see your session in the Operator navigator Session List. Expand the Sessions node and then expand the mapping you ran to see your session. The session icon indicates whether the session is still running, completed, or stopped due to errors. For more information about monitoring your sessions, see: the Monitoring Integration Processes chapter in *Administering Oracle Data Integrator*.

Using Mapping Components

In the logical view of the mapping editor, you design a mapping by combining datastores with other components. You can use the mapping diagram to arrange and connect components such as datasets, filters, sorts, and so on. You can form connections between datastores and components by dragging lines between the connector ports displayed on these objects.

Mapping components can be divided into two categories which describe how they are used in a mapping: projector components and selector components.

Projector Components

Projectors are components that influence the attributes present in the data that flows through a mapping. Projector components define their own attributes: attributes from preceding components are mapped through expressions to the projector's attributes. A projector hides attributes originating from preceding components; all succeeding components can only use the attributes from the projector.

Review the following topics to learn how to use the various projector components:

- Source and Target Datastores
- Creating Multiple Targets
- Adding a Reusable Mapping
- Creating Aggregates
- Creating Distincts
- Creating Pivots
- Creating Sets
- Creating Subquery Filters
- Creating Table Functions
- Creating Unpivots
- Creating Flatten Components
- Creating Jagged Components

Selector Components

Selector components reuse attributes from preceding components. Join and Lookup selectors combine attributes from the preceding components. For example, a Filter component following a datastore component reuses all attributes from the datastore component. As a consequence, selector components don't display their own attributes in the diagram and as part of the properties; they are displayed as a round shape. (The Expression component is an exception to this rule.)

When mapping attributes from a selector component to another component in the mapping, you can select and then drag an attribute from the source, across a chain of connected selector components, to a target datastore or next projector component. ODI will automatically create the necessary queries to bring that attribute across the intermediary selector components.

Review the following topics to learn how to use the various selector components:

- Creating Expressions
- Creating Filters
- Creating Joins and Lookups
- Creating Sorts
- Creating Splits
- Creating a Dataset in a Mapping

The Expression Editor

Most of the components you use in a mapping are actually representations of an expression in the code that acts on the data as it flows from your source to your target datastores. When you create or modify these components, you can edit the expression's code directly in the Property Inspector.

To assist you with more complex expressions, you can also open an advanced editor called the Expression Editor. (In some cases, the editor is labeled according to the type of component; for example, from a Filter component, the editor is called the Filter Condition Advanced Editor. However, the functionality provided is the same.)

To access the Expression Editor, select a component, and in the Property Inspector, select or hover over with the mouse pointer any field containing code. A gear icon appears to the right of the field. Click the gear icon to open the Expression Editor.

For example, to see the gear icon in a Filter component, select or hover over the Filter Condition field on the Condition tab; to see the gear icon in a Datastore component, select or hover over the Journalized Data Filter field of the Journalizing tab.

The Expression Editor is made up of the following panels:

- **Attributes**: This panel appears on the left of the Expression Editor. When editing an expression for a mapping, this panel contains the names of attributes which are "in scope," meaning, attributes that are currently visible and can be referenced by the expression of the component. For example, if a component is connected to a source datastore, all of the attributes of that datastore are listed.
- **Expression**: This panel appears in the middle of the Expression Editor. It displays the current code of the expression. You can directly type code here, or drag and drop elements from the other panels.
- **Technology functions**: This panel appears below the expression. It lists the language elements and functions appropriate for the given technology.
- **Variables, Sequences, User Functions and odiRef API**: This panel appears to the right of the technology functions and contains:
  - Project and global Variables.
  - Project and global Sequences.
  - Project and global User-Defined Functions.
  - OdiRef Substitution Methods.

Standard editing functions (cut/copy/paste/undo/redo) are available using the tool bar buttons.

Source and Target Datastores

To insert a source or target datastore in a mapping:

1. In the Designer Navigator, expand the **Models** tree and expand the model or sub-model containing the datastore to be inserted as a source or target.
2. Select this datastore, then drag it into the mapping panel. The datastore appears.
3. To make the datastore a source, drag a link from the output (right) connector of the datastore to one or more components. A datastore is not a source until it has at least one outgoing connection.

   To make the datastore a target, drag a link from a component to the input (left) connector of the datastore. A datastore is not a target until it has an incoming connection.

Once you have defined a datastore you may wish to view its data.

To display the data of a datastore in a mapping:

1. Right-click the title of the datastore in the mapping diagram.
2. Select **Data...**

The Data Editor opens.

Creating Multiple Targets

In Oracle Data Integrator 12*c*, creating multiple targets in a mapping is straightforward. Every datastore component which has inputs but no outputs in the logical diagram is considered a target.

ODI allows splitting a component output into multiple flows at any point of a mapping. You can also create a single mapping with multiple independent flows, avoiding the need for a package to coordinate multiple mappings.

The output port of many components can be connected to multiple downstream components, which will cause all rows of the component result to be processed in each of the downstream flows. If rows should be routed or conditionally processed in the downstream flows, consider using a split component to define the split conditions.

See Also:

Creating Splits

Specifying Target Order

Mappings with multiple targets do not, by default, follow a defined order of loading data to targets. You can define a partial or complete order by using the **Target Load Order** property. Targets which you do not explicitly assign an order will be loaded in an arbitrary order by ODI.

Note:

Target load order also applies to reusable mappings. If a reusable mapping contains a source or a target datastore, you can include the reusable mapping component in the target load order property of the parent mapping.

The order of processing multiple targets can be set in the **Target Load Order** property of the mapping:

1. Click the background in the logical diagram to deselect objects in the mapping. The property inspector displays the properties for the mapping.
2. In the property inspector, accept the default target load order, or enter a new target load order, in the **Target Load Order** field.

Note:

A default load order is automatically computed based on primary key/foreign key relationships of the target datastores in the mapping. You can modify this default if needed, even if the resultant load order conflicts with the primary key/foreign key relationship. A warning will be displayed when you validate the mapping in this case.

Select or hover over the **Target Load Order** field and click the gear icon to open the **Target Load Order Dialog.** This dialog displays all available datastores (and reusable mappings containing datastores) that can be targets, allowing you to move one or more to the **Ordered Targets** field. In the **Ordered Targets** field, use the icons on the right to rearrange the order of processing.

Tip:

Target Order is useful when a mapping has multiple targets and there are foreign key (FK) relationships between the targets. For example, suppose a mapping has two targets called `EMP` and `DEPT`, and `EMP.DEPTNO` is a FK to `DEPT.DEPTNO`. If the source data contains information about the employee and the department, the information about the department (`DEPT`) must be loaded first before any rows about the employee can be loaded (`EMP`). To ensure this happens, the target load order should be set to `DEPT, EMP`.

Adding a Reusable Mapping

Reusable mappings may be stored within folders in a project, or as global objects within the Global Objects tree, of the Designer Navigator.

To add a reusable mapping to a mapping:

1. To add a reusable mapping stored within the current project:

   In the Designer Navigator, expand the **Projects** tree and expand the tree for the project you are working on. Expand the Reusable Mappings node to list all reusable mappings stored within this project.

   To add a global reusable mapping:

   In the Designer Navigator, expand the Global Objects tree, and expand the Reusable Mappings node to list all global reusable mappings.

2. Select a reusable mapping, and drag it into the mapping diagram. A reusable mapping component is added to the diagram as an interface to the underlying reusable mapping.

Creating Aggregates

The aggregate component is a projector component (see the **Projector Components** section in Using Mapping Components) which groups and combines attributes using aggregate functions, such as average, count, maximum, sum, and so on. ODI will automatically select attributes without aggregation functions to be used as group-by attributes. You can override this by using the **Is Group By** and **Manual Group By Clause** properties.

To create an aggregate component:

1. Drag and drop the aggregate component from the component palette into the logical diagram.
2. Define the attributes of the aggregate if the attributes will be different from the source components. To do this, select the **Attributes** tab in the property inspector, and click the green plus icon to add attributes. Enter new attribute names in the **Target** column and assign them appropriate values.

   If attributes in the aggregate component will be the same as those in a source component, use attribute matching (see Step 4).
3. Create a connection from a source component by dragging a line from the connector port of the source to the connector port of the aggregate component.
4. The **Attribute Matching** dialog will be shown. If attributes in the aggregate component will be the same as those in a source component, check the **Create Attributes on Target** box (see: Attribute Matching).
5. If necessary, map all attributes from source to target that were not mapped though attribute matching, and create transformation expressions as necessary (see: Defining Expressions and Conditions).
6. In the property inspector, the attributes are listed in a table on the **Attributes** tab. Specify aggregation functions for each attribute as needed. By default all attributes not mapped using aggregation functions (such as sum, count, avg, max, min, and so on) will be used as Group By.

   You can modify an aggregation expression by clicking the attribute. For example, if you want to calculate average salary per department, you might have two attributes: the first attribute called `AVG_SAL`, which you give the expression `AVG(EMP.SAL)`, while the second attribute called `DEPTNO` has no expression. If **Is Group By** is set to `Auto`, `DEPTNO` will be automatically included in the `GROUP BY` clause of the generated code.

   You can override this default by changing the property **Is Group By** on a given attribute from `Auto` to `Yes` or `No`, by double-clicking on the table cell and selecting the desired option from the drop down list.

   You can set a different `GROUP BY` clause other than the default for the entire aggregate component. Select the **General** tab in the property inspector, and then set a **Manual Group by Clause**. For example, set the **Manual Group by Clause** to `YEAR(customer.birthdate)` to group by birthday year.
7. Optionally, add a `HAVING` clause by setting the **HAVING** property of the aggregate component: for example, `SUM(order.amount) > 1000`.

Creating Distincts

A distinct is a projector component (see the **Projector Components** section in Using Mapping Components) that projects a subset of attributes in the flow. The values of each row have to be unique; the behavior follows the rules of the SQL DISTINCT clause.

To select distinct rows from a source datastore:

1. Drag and drop a Distinct component from the component palette into the logical diagram.
2. Connect the preceding component to the Distinct component by dragging a line from the preceding component to the Distinct component.

   The **Attribute Mapping Dialog** will appear: select **Create Attributes On Target** to create all of the attributes in the Distinct component. Alternatively, you can manually map attributes as desired using the **Attributes** tab in the property inspector.
3. The distinct component will now filter all rows that have all projected attributes matching.

Creating Expressions

An expression is a selector component (see the **Selector Components** section in Using Mapping Components) that inherits attributes from a preceding component in the flow and adds additional reusable attributes. An expression can be used to define a number of reusable expressions within a single mapping. Attributes can be renamed and transformed from source attributes using SQL expressions. The behavior follows the rules of the `SQL SELECT` clause.

The best use of an expression component is in cases where intermediate transformations are used multiple times, such as when pre-calculating fields that are used in multiple targets.

If a transformation is used only once, consider performing the transformation in the target datastore or other component.

Tip:

If you want to reuse expressions across multiple mappings, consider using reusable mappings or user functions, depending on the complexity. See: Reusable Mappings, and the section Working with User Functions in *Developing Integration Projects with Oracle Data Integrator*.

To create an expression component:

1. Drag and drop an Expression component from the component palette into the logical diagram.
2. Connect a preceding component to the Expression component by dragging a line from the preceding component to the Expression component.

   The **Attribute Mapping Dialog** will appear; select **Create Attributes On Target** to create all of the attributes in the Expression component.

   In some cases you might want the expression component to match the attributes of a downstream component. In this case, connect the expression component with the downstream component first and select **Create Attributes on Source** to populate the Expression component with attributes from the target.
3. Add attributes to the expression component as desired using the **Attributes** tab in the property inspector. It might be useful to add attributes for pre-calculated fields that are used in multiple expressions in downstream components.

4. Edit the expressions of individual attributes as necessary (see: Defining Expressions and Conditions).

Creating Filters

A filter is a selector component (see the **Selector Components** section in Using Mapping Components) that can select a subset of data based on a filter condition. The behavior follows the rules of the `SQL WHERE` clause.

Filters can be located in a dataset or directly in a mapping as a flow component.

When used in a dataset, a filter is connected to one datastore or reusable mapping to filter all projections of this component out of the dataset. For more information, see Creating a Mapping Using a Dataset.

To define a filter in a mapping:

1. Drag and drop a Filter component from the component palette into the logical diagram.
2. Drag an attribute from the preceding component onto the filter component. A connector will be drawn from the preceding component to the filter, and the attribute will be referenced in the filter condition.

   In the **Condition** tab of the Property Inspector, edit the **Filter Condition** and complete the expression. For example, if you want to select from the CUSTOMER table (that is the source datastore with the CUSTOMER alias) only those records with a NAME that is not null, an expression could be `CUSTOMER.NAME IS NOT NULL`.

   Tip:

   Click the gear icon to the right of the **Filter Condition** field to open the **Filter Condition Advanced Editor**. The gear icon is only shown when you have selected or are hovering over the Filter Condition field with your mouse pointer. For more information about the Filter Condition Advanced Editor, see: The Expression Editor.

3. Optionally, on the **General** tab of the Property Inspector, enter a new name in the **Name** field. Using a unique name is useful if you have multiple filters in your mapping.
4. Optionally, set an **Execute on Hint**, to indicate your preferred execution location: `No hint`, `Source`, `Staging`, or `Target`. The physical diagram will locate the execution of the filter according to your hint, if possible. For more information, see Configuring Execution Locations.

Creating Joins and Lookups

This section contains the following topics:

About Joins

A Join is a selector component (see the **Selector Components** section in Using Mapping Components) that creates a join between multiple flows. The attributes from upstream components are combined as attributes of the Join component.

A Join can be located in a dataset or directly in a mapping as a flow component. A join combines data from two or more data flows, which may be datastores, datasets, reusable mappings, or combinations of various components.

When used in a dataset, a join combines the data of the datastores using the selected join type. For more information, see Creating a Mapping Using a Dataset.

A join used as a flow component can join two or more sources of attributes, such as datastores or other upstream components. A join condition can be formed by dragging attributes from two or more components successively onto a join component in the mapping diagram; by default the join condition will be an equi-join between the two attributes.

About Lookups

A Lookup is a selector component (see the **Selector Components** section in Using Mapping Components) that returns data from a lookup flow being given a value from a driving flow. The attributes of both flows are combined, similarly to a join component.

Lookups can be located in a dataset or directly in a mapping as a flow component.

When used in a dataset, a Lookup is connected to two datastores or reusable mappings combining the data of the datastores using the selected join type. For more information, see Creating a Mapping Using a Dataset.

Lookups used as flow components (that is, not in a dataset) can join two flows. A lookup condition can be created by dragging an attribute from the driving flow and then the lookup flow onto the lookup component; the lookup condition will be an equi-join between the two attributes.

The **Multiple Match Rows** property defines which row from the lookup result must be selected as the lookup result if the lookup returns multiple results. Multiple rows are returned when the lookup condition specified matches multiple records.

You can select one of the following options to specify the action to perform when multiple rows are returned by the lookup operation:

- **Error: multiple rows will cause a mapping failure**

  This option indicates that when the lookup operation returns multiple rows, the mapping execution fails.

  > **Note:**
  >
  > In ODI 12.1.3, the **Deprecated - Error: multiple rows will cause a mapping failure** option with the `EXPRESSION_IN_SELECT` option value is deprecated. It is included for backward compatibility with certain patched versions of ODI 12.1.2.
  >
  > This option is replaced with the `ERROR_WHEN_MULTIPLE_ROW` option of **Error: multiple rows will cause a mapping failure**.

- **All Rows (number of result rows may differ from the number of input rows)**

  This option indicates that when the lookup operation returns multiple rows, all the rows should be returned as the lookup result.

> **Note:**
>
> In ODI 12.1.3, the **Deprecated - All rows (number of result rows may differ from the number of input rows** option with the `LEFT_OUTER` option value is deprecated. It is included for backward compatibility with certain patched versions of ODI 12.1.2.
>
> This option is replaced with the `ALL_ROWS` option of **All rows (number of result rows may differ from the number of input rows**.

- **Select any single row**

  This option indicates that when the lookup operation returns multiple rows, any one row from the returned rows must be selected as the lookup result.

- **Select first single row**

  This option indicates that when the lookup operation returns multiple rows, the first row from the returned rows must be selected as the lookup result.

- **Select nth single row**

  This option indicates that when the lookup operation returns multiple rows, the nth row from the result rows must be selected as the lookup result. When you select this option, the **Nth Row Number** field appears, where you can specify the value of n.

- **Select last single row**

  This option indicates that when the lookup operation returns multiple rows, the last row from the returned rows must be selected as the lookup result.

Use the **Lookup Attributes Default Value & Order By** table to specify how the result set that contains multiple rows should be ordered, and what the default value should be if no matches are found for the input attribute in the lookup flow through the lookup condition. Ensure that the attributes are listed in the same order (from top to bottom) in which you want the result set to be ordered. For example, to implement an ordering such as ORDER BY attr2, attr3, and then attr1, the attributes should be listed in the same order. You can use the arrow buttons to change the position of the attributes to specify the order.

The **No-Match Rows** property indicates the action to be performed when there are no rows that satisfy the lookup condition. You can select one of the following options to perform when no rows are returned by the lookup operation:

- **Return no row**

  This option does not return any row when no row in the lookup results satisfies the lookup condition.

- **Return a row with the following default values**

  This option returns a row that contains default values when no row in the lookup results satisfies the lookup condition. Use the **Lookup Attributes Default Value & Order By:** table below this option to specify the default values for each lookup attribute.

Creating a Join or Lookup

To create a join or a lookup between two upstream components:

1. Drag a join or lookup from the component palette into the logical diagram.
2. Drag the attributes participating in the join or lookup condition from the preceding components onto the join or lookup component. For example, if attribute `ID` from source datastore `CUSTOMER` and then `CUSTID` from source datastore `ORDER` are dragged onto a join, then the join condition `CUSTOMER.ID = ORDER.CUSTID` is created.

   Note:

   When more than two attributes are dragged into a join or lookup, ODI compares and combines attributes with an AND operator. For example, if you dragged attributes from sources A and B into a Join component in the following order:

   ```
   A.FIRSTNAME
   B.FIRSTNAME
   A.LASTNAME
   B.LASTNAME
   ```

   The following join condition would be created:

   ```
   A.FIRSTNAME=B.FIRSTNAME AND A.LASTNAME=B.LASTNAME
   ```

   You can continue with additional pairs of attributes in the same way.

   You can edit the condition after it is created, as necessary.

3. In the **Condition** tab of the Property Inspector, edit the **Join Condition** or **Lookup Condition** and complete the expression.

   Tip:

   Click the gear icon to the right of the **Join Condition** or **Lookup Condition** field to open the Expression Editor. The gear icon is only shown when you have selected or are hovering over the condition field with your mouse pointer. For more information about the Expression Editor, see: The Expression Editor.

4. Optionally, set an **Execute on Hint**, to indicate your preferred execution location: `No hint`, `Source`, `Staging`, or `Target`. The physical diagram will locate the execution of the filter according to your hint, if possible.
5. For a join:

   Select the **Join Type** by checking the various boxes (**Cross**, **Natural**, **Left Outer**, **Right Outer**, **Full Outer** (by checking both left and right boxes), or (by leaving all boxes empty) **Inner Join**). The text describing which rows are retrieved by the join is updated.

   For a lookup:

   Select the **Multiple Match Rows** by selecting an option from the drop down list. The Technical Description field is updated with the SQL code representing the lookup, using fully-qualified attribute names.

   If applicable, use the **Lookup Attributes Default Value & Order By** table to specify how a result set that contains multiple rows should be ordered.

Select a value for the **No-Match Rows** property to indicate the action to be performed when there are no rows that satisfy the lookup condition.

6. Optionally, for joins, if you want to use an ordered join syntax for this join, check the **Generate ANSI Syntax** box.

   The Join Order box will be checked if you enable **Generate ANSI Syntax**, and the join will be automatically assigned an order number.

   > **WARNING:**
   >
   > The ordered join option works only in Datasets.

7. For joins inside of datasets, define the join order. Check the **Join Order** check box, and then in the **User Defined** field, enter an integer. A join component with a smaller join order number means that particular join will be processed first among other joins. The join order number determines how the joins are ordered in the `FROM` clause. A smaller join order number means that the join will be performed earlier than other joins. This is important when there are outer joins in the dataset.

   For example: A mapping has two joins, *JOIN1* and *JOIN2*. *JOIN1* connects *A* and *B*, and its join type is `LEFT OUTER JOIN`. *JOIN2* connects *B* and *C*, and its join type is `RIGHT OUTER JOIN`.

   To generate `(A LEFT OUTER JOIN B) RIGHT OUTER JOIN C`, assign a join order `10` for *JOIN1* and `20` for *JOIN2*.

   To generate `A LEFT OUTER JOIN (B RIGHT OUTER JOIN C)`, assign a join order `20` for `JOIN1` and `10` for *JOIN2*.

Creating Pivots

A pivot component is a projector component (see the **Projector Components** section in Using Mapping Components) that lets you transform data that is contained in multiple input rows into a single output row. The pivot component lets you extract data from a source once, and produce one row from a set of source rows that are grouped by attributes in the source data. The pivot component can be placed anywhere in the data flow of a mapping.

Example: Pivoting Sales Data

The table below shows a sample of data from the SALES relational table. The QUARTER attribute has four possible character values, one for each quarter of the year. All the sales figures are contained in one attribute, SALES.

Table 13-2 SALES

YEAR	QUARTER	SALES
2010	Q1	10.5
2010	Q2	11.4
2010	Q3	9.5
2010	Q4	8.7
2011	Q1	9.5

Table 13-2 (Cont.) SALES

YEAR	QUARTER	SALES
2011	Q2	10.5
2011	Q3	10.3
2011	Q4	7.6

The following table depicts data from the relational table SALES after pivoting the table. The data that was formerly contained in the QUARTER attribute (Q1, Q2, Q3, and Q4) corresponds to 4 separate attributes (Q1_Sales, Q2_Sales, Q3_Sales, and Q4_Sales). The sales figures formerly contained in the SALES attribute are distributed across the 4 attributes for each quarter.

Table 13-3 PIVOTED DATA

Year	Q1_Sales	Q2_Sales	Q3_Sales	Q4_Sales
2010	10.5	11.4	9.5	8.7
2011	9.5	10.5	10.3	7.6

The Row Locator

When you use the pivot component, multiple input rows are transformed into a single row based on the row locator. The row locator is an attribute that you must select from the source to correspond with the set of output attributes that you define. It is necessary to specify a row locator to perform the pivot operation.

In this example, the row locator is the attribute QUARTER from the SALES table and it corresponds to the attributes Q1_Sales, Q2_Sales, Q3_Sales, and Q4_Sales attributes in the pivoted output data.

Using the Pivot Component

To use a pivot component in a mapping:

1. Drag and drop the source datastore into the logical diagram.
2. Drag and drop a Pivot component from the component palette into the logical diagram.
3. From the source datastore drag and drop the appropriate attributes on the pivot component. In this example, the YEAR attribute.

Note:

Do not drag the row locator attribute or the attributes that contain the data values that correspond to the output attributes. In this example, QUARTER is the row locator attribute and SALES is the attribute that contain the data values (sales figures) that correspond to the Q1_Sales, Q2_Sales, Q3_Sales, and Q4_Sales output attributes.

4. Select the pivot component. The properties of the pivot component are displayed in the Property Inspector.
5. Enter a name and description for the pivot component.
6. If required, change the Aggregate Function for the pivot component. The default is `MIN`.
7. Type in the expression or use the Expression Editor to specify the row locator. In this example, since the QUARTER attribute in the SALES table is the row locator, the expression will be SALES.QUARTER.
8. Under Row Locator Values, click the + sign to add the row locator values. In this example, the possible values for the row locator attribute QUARTER are Q1, Q2, Q3, and Q4.
9. Under Attributes, add output attributes to correspond to each input row. If required, you can add new attributes or rename the listed attributes.

   In this example, add 4 new attributes, Q1_Sales, Q2_Sales, Q3_Sales, and Q4_Sales that will correspond to 4 input rows Q1, Q2, Q3, and Q4 respectively.
10. If required, change the expression for each attribute to pick up the sales figures from the source and select a matching row for each attribute.

    In this example, set the expressions for each attribute to SALES.SALES and set the matching rows to Q1, Q2, Q3, and Q4 respectively.
11. Drag and drop the target datastore into the logical diagram.
12. Connect the pivot component to the target datastore by dragging a link from the output (right) connector of the pivot component to the input (left) connector of the target datastore.
13. Drag and drop the appropriate attributes of the pivot component on to the target datastore. In this example, YEAR, Q1_Sales, Q2_Sales, Q3_Sales, and Q4_Sales.
14. Go to the physical diagram and assign new KMs if you want to.

    Save and execute the mapping to perform the pivot operation.

Creating Sets

A set component is a projector component (see the **Projector Components** section in Using Mapping Components) that combines multiple input flows into one using set operation such as `UNION`, `INTERSECT`, `EXCEPT`, `MINUS` and others. The behavior reflects the SQL operators.

Note:

`PigSetCmd` does not support the `EXCEPT` set operation.

Additional input flows can be added to the set component by connecting new flows to it. The number of input flows is shown in the list of Input Connector Points in the Operators tab. If an input flow is removed, the input connector point needs to be removed as well.

To create a set from two or more sources:

1. Drag and drop a Set component from the component palette into the logical diagram.
2. Define the attributes of the set if the attributes will be different from the source components. To do this, select the **Attributes** tab in the property inspector, and click the green plus icon to add attributes. Select the new attribute names in the **Target** column and assign them appropriate values.

   If Attributes will be the same as those in a source component, use attribute matching (see step 4).
3. Create a connection from the first source by dragging a line from the connector port of the source to the connector port of the Set component.
4. The **Attribute Matching** dialog will be shown. If attributes of the set should be the same as the source component, check the **Create Attributes on Target** box (see: Attribute Matching).
5. If necessary, map all attributes from source to target that were not mapped through attribute matching, and create transformation expressions as necessary (see: Defining Expressions and Conditions).
6. All mapped attributes will be marked by a yellow arrow in the logical diagram. This shows that not all sources have been mapped for this attribute; a set has at least two sources.
7. Repeat the connection and attribute mapping steps for all sources to be connected to this set component. After completion, no yellow arrows should remain.
8. In the property inspector, select the **Operators** tab and select cells in the **Operator** column to choose the appropriate set operators (`UNION`, `EXCEPT`, `INTERSECT`, and so on). `UNION` is chosen by default. You can also change the order of the connected sources to change the set behavior.

Note:

You can set **Execute On Hint** on the attributes of the set component, but there is also an **Execute On Hint** property for the set component itself. The hint on the component indicates the preferred location where the actual set operation (`UNION`, `EXCEPT`, and so on) is performed, while the hint on an attribute indicates where the preferred location of the expression is performed.

A common use case is that the set operation is performed on a staging execution unit, but some of its expressions can be done on the source execution unit. For more information about execution units, see Configuring Execution Locations.

Creating Sorts

A Sort is a projector component (see the **Projector Components** section in Using Mapping Components) that will apply a sort order to the rows of the processed dataset, using the `SQL ORDER BY` statement.

To create a sort on a source datastore:

1. Drag and drop a Sort component from the component palette into the logical diagram.
2. Drag the attribute to be sorted on from a preceding component onto the sort component. If the rows should be sorted based on multiple attributes, they can be dragged in desired order onto the sort component.
3. Select the sort component and select the **Condition** tab in the property inspector. The **Sorter Condition** field follows the syntax of the `SQL ORDER BY` statement of the underlying database; multiple fields can be listed separated by commas, and `ASC` or `DESC` can be appended after each field to define if the sort will be ascending or descending.

Creating Splits

A Split is a selector component (see the **Selector Components** section in Using Mapping Components) that divides a flow into two or more flows based on specified conditions. Split conditions are not necessarily mutually exclusive: a source row is evaluated against all split conditions and may be valid for multiple output flows.

If a flow is divided unconditionally into multiple flows, no split component is necessary: you can connect multiple downstream components to a single outgoing connector port of any preceding component, and the data output by that preceding component will be routed to all downstream components.

A split component is used to conditionally route rows to multiple proceeding flows and targets.

To create a split to multiple targets in a mapping:

1. Drag and drop a Split component from the component palette into the logical diagram.
2. Connect the split component to the preceding component by dragging a line from the preceding component to the split component.
3. Connect the split component to each following component. If either of the upstream or downstream components contain attributes, the Attribute Mapping Dialog will appear. In the **Connection Path** section of the dialog, it will default to the first unmapped connector point and will add connector points as needed. Change this selection if a specific connector point should be used.
4. In the property inspector, open the **Split Conditions** tab. In the **Output Connector Points** table, enter expressions to select rows for each target. If an expression is left empty, all rows will be mapped to the selected target. Check the **Remainder** box to map all rows that have not been selected by any of the other targets.

Creating Subquery Filters

A subquery filter component is a projector component (see the **Projector Components** section in Using Mapping Components) that lets you to filter rows based on the results of a subquery. The conditions that you can use to filter rows are EXISTS, NOT EXISTS, IN, and NOT IN.

For example, the EMP datastore contains employee data and the DEPT datastore contains department data. You can use a subquery to fetch a set of records from the

DEPT datastore and then filter rows from the EMP datastore by using one of the subquery conditions.

A subquery filter component has two input connector points and one output connector point. The two input connector points are Driver Input connector point and Subquery Filter Input connector point. The Driver Input connector point is where the main datastore is set, which drives the whole query. The Subquery Filter Input connector point is where the datastore that is used in the sub-query is set. In the example, EMP is the Driver Input connector point and DEPT is the Subquery Filter Input connector point.

To filter rows using a subquery filter component:

1. Drag and drop a subquery filter component from the component palette into the logical diagram.
2. Connect the subquery filter component with the source datastores and the target datastore.
3. Drag and drop the input attributes from the source datastores on the subquery filter component.
4. Drag and drop the output attributes of the subquery filter component on the target datastore.
5. Go to the Connector Points tab and select the input datastores for the driver input connector point and the subquery filter input connector point.
6. Click the subquery filter component. The properties of the subquery filter component are displayed in the Property Inspector.
7. Go to the Attributes tab. The output connector point attributes are listed. Set the expressions for the driver input connector point and the subquery filter connector point.

   > **Note:**
   >
   > You are required to set an expression for the subquery filter input connector point only if the subquery filter input role is set to one of the following:
   >
   > IN, NOT IN, =, >, <, >=, <=, !=, <>, ^=

8. Go to the Condition tab.
9. Type an expression in the Subquery Filter Condition field. It is necessary to specify a subquery filter condition if the subquery filter input role is set to EXISTS or NOT EXISTS.
10. Select a subquery filter input role from the **Subquery Filter Input Role** drop-down list.
11. Select a group comparison condition from the **Group Comparison Condition** drop-down list. A group comparison condition can be used only with the following subquery input roles:

    =, >, <, >=, <=, !=, <>, ^=
12. Save and then execute the mapping.

Creating Table Functions

A table function component is a projector component (see the **Projector Components** section in Using Mapping Components) that represents a table function in a mapping. Table function components enable you to manipulate a set of input rows and return another set of output rows of the same or different cardinality. The set of output rows can be queried like a physical table. A table function component can be placed anywhere in a mapping, as a source, a target, or a data flow component.

A table function component can have multiple input connector points and one output connector point. The input connector point attributes act as the input parameters for the table function, while the output connector point attributes are used to store the return values.

For each input connector, you can define the parameter type, REF_CURSOR or SCALAR, depending on the type of attributes the input connector point will hold.

To use a table function component in a mapping:

1. Create a table function in the database if it does not exist.
2. Right-click the **Mappings** node and select **New Mapping**.
3. Drag and drop the source datastore into the logical diagram.
4. Drag and drop a table function component from the component palette into the logical diagram. A table function component is created with no input connector points and one default output connector point.
5. Click the table function component. The properties of the table function component are displayed in the Property Inspector.
6. In the property inspector, go to the Attributes tab.
7. Type the name of the table function in the **Name** field. If the table function is in a different schema, type the function name as SCHEMA_NAME.FUNCTION_NAME.
8. Go to the Connector Points tab and click the + sign to add new input connector points. Do not forget to set the appropriate parameter type for each input connector.

   Note:

   Each REF_CURSOR attribute must be held by a separate input connector point with its parameter type set to REF_CURSOR. Multiple SCALAR attributes can be held by a single input connector point with its parameter type set to SCALAR.

9. Go to the Attributes tab and add attributes for the input connector points (created in previous step) and the output connector point. The input connector point attributes act as the input parameters for the table function, while the output connector point attributes are used to store the return values.
10. Drag and drop the required attributes from the source datastore on the appropriate attributes for the input connector points of the table function component. A connection between the source datastore and the table function component is created.

11. Drag and drop the target datastore into the logical diagram.
12. Drag and drop the output attributes of the table function component on the attributes of the target datastore.
13. Go to the physical diagram of the mapping and ensure that the table function component is in the correct execution unit. If it is not, move the table function to the correct execution unit.
14. Assign new KMs if you want to.
15. Save and then execute the mapping.

Creating Unpivots

An unpivot component is a projector component (see the **Projector Components** section in Using Mapping Components) that lets you transform data that is contained across attributes into multiple rows.

The unpivot component does the reverse of what the pivot component does. Similar to the pivot component, an unpivot component can be placed anywhere in the flow of a mapping.

The unpivot component is specifically useful in situations when you extract data from non-relational data sources such as a flat file, which contains data across attributes rather than rows.

Example: Unpivoting Sales Data

The external table, QUARTERLY_SALES_DATA, shown in the table below, contains data from a flat file. There is a row for each year and separate attributes for sales in each quarter.

Table 13-4 QUARTERLY_SALES_DATA

Year	Q1_Sales	Q2_Sales	Q3_Sales	Q4_Sales
2010	10.5	11.4	9.5	8.7
2011	9.5	10.5	10.3	7.6

The table below shows a sample of the data after an unpivot operation is performed. The data that was formerly contained across multiple attributes (Q1_Sales, Q2_Sales, Q3_Sales, and Q4_Sales) is now contained in a single attribute (SALES). The unpivot component breaks the data in a single attribute (Q1_Sales) into two attributes (QUARTER and SALES). A single row in QUARTERLY_SALES_DATA corresponds to 4 rows (one for sales in each quarter) in the unpivoted data.

Table 13-5 UNPIVOTED DATA

YEAR	QUARTER	SALES
2010	Q1	10.5
2010	Q2	11.4
2010	Q3	9.5
2010	Q4	8.7

Table 13-5 (Cont.) UNPIVOTED DATA

YEAR	QUARTER	SALES
2011	Q1	9.5
2011	Q2	10.5
2011	Q3	10.3
2011	Q4	7.6

The Row Locator

The row locator is an output attribute that corresponds to the repeated set of data from the source. The unpivot component transforms a single input attribute into multiple rows and generates values for a row locator. The other attributes that correspond to the data from the source are referred as value locators. In this example, the attribute QUARTER is the row locator and the attribute SALES is the value locator.

Note:

To use the unpivot component, you are required to create the row locator and the value locator attributes for the unpivot component.

The **Value Locator** field in the **Unpivot Transforms** table can be populated with an arbitrary expression. For example:

```
UNPIVOT_EMP_SALES.Q1_SALES + 100
```

Using the Unpivot Component

To use an unpivot component in a mapping:

1. Drag and drop the source data store into the logical diagram.
2. Drag and drop an unpivot component from the component palette into the logical diagram.
3. From the source datastore drag and drop the appropriate attributes on the unpivot component. In this example, the YEAR attribute.

   Note:

   Do not drag the attributes that contain the data that corresponds to the value locator. In this example, Q1_Sales, Q2_Sales, Q3_Sales, and Q4_Sales.

4. Select the unpivot component. The properties of the unpivot component are displayed in the Property Inspector.
5. Enter a name and description for the unpivot component.

6. Create the row locator and value locator attributes using the Attribute Editor. In this example, you need to create two attributes named QUARTER and SALES.

   Note:

   Do not forget to define the appropriate data types and constraints (if required) for the attributes.

7. In the Property Inspector, under UNPIVOT, select the row locator attribute from the **Row Locator** drop-down list. In this example, QUARTER.

   Now that the row locator is selected, the other attributes can act as value locators. In this example, SALES.

8. Under UNPIVOT TRANSFORMS, click + to add transform rules for each output attribute. Edit the default values of the transform rules and specify the appropriate expressions to create the required logic.

   In this example, you need to add 4 transform rules, one for each quarter. The transform rules define the values that will be populated in the row locator attribute QUARTER and the value locator attribute SALES. The QUARTER attribute must be populated with constant values (Q1, Q2, Q3, and Q4), while the SALES attribute must be populated with the values from source datastore attributes (Q1_Sales, Q2_Sales, Q3_Sales, and Q4_Sales).

9. Leave the INCLUDE NULLS check box selected to generate rows with no data for the attributes that are defined as NULL.
10. Drag and drop the target datastore into the logical diagram.
11. Connect the unpivot component to the target datastore by dragging a link from the output (right) connector of the unpivot component to the input (left) connector of the target datastore.
12. Drag and drop the appropriate attributes of the unpivot component on to the target datastore. In this example, YEAR, QUARTER, and SALES.
13. Go to the physical diagram and assign new KMs if you want to.
14. Click **Save** and then execute the mapping to perform the unpivot operation.

Creating Flatten Components

The flatten component is a Projector component (see the **Projector Components** section in Using Mapping Components) that processes input data with complex structure and produces a flattened representation of the same data using standard datatypes.

Flatten produces a cross product of the nested structure with the enclosing structure, so that every row of the nested structure produces a new row in the result.

The flatten component has one input connector point and one output connector point.

Example: Flatten Complex Data

The table below shows an example of a datastore `movie_ratings`, which has a complex type attribute `ratings`. The complex type attribute `ratings` is repeating and has child attribute `rating`.

Table 13-6 movie_ratings

movie_id	year	title	ratings		
-	-	-	**rating**	**rating**	**rating**
1	1970	Nexus	2	5	3

The table below shows the resulting records after flatten.

Table 13-7 movie_ratings

movie_id	year	title	rating
1	1970	Nexus	2
1	1970	Nexus	5
1	1970	Nexus	3

Using a Flatten Component in a Mapping

To use a flatten component in a mapping:

1. Drag and drop the source data store into the logical diagram.
2. Drag and drop a flatten component from the component palette into the logical diagram.
3. Choose one attribute from the source component to be flattened and enter it into the property **Complex Type Attribute** of the Flatten component. This attribute should be a complex type in the source data.
4. Manually enter all attributes of the complex type in the **Attributes** properties of the Flatten component. These attributes should have no expression.
5. Map any other source attributes into the Flatten component.
6. Check the property **Include Nulls** if the complex type attribute can be null or an empty array.
7. Connect the Flatten component to a target datastore or any other downstream components.
8. Go to the physical diagram and assign new KMs if you want to.
9. Click **Save** and then execute the mapping to perform the flatten operation.

Considerations for using Flatten component with JSON Source

When you use Flatten component and your source is JSON data, you must consider the following points:

- Flatten does not support multiple child objects and nested objects within a JSON file. The source JSON data must be plain, as shown in the following example:

  Example: `{"POSTAL_AREA":1, "POSTAL_AREA_DETAIL": [{"STATE":"CA","POSTAL_CODE":"200001"}]}`

- When a JSON file and a flatten component is used in a mapping with Pig as the staging area, you must set the **Storage Function** option and the **Schema for Complex Fields** option for LKM File to Pig.

  These options must be set as shown in the following example:

  Storage Function: `JsonLoader`

  Schema for Complex Fields: `POSTAL_AREA_DETAIL:{(STATE:chararray,POSTAL_CODE:chararray)}`

Creating Jagged Components

The jagged component is a Projector component that processes unstructured data using meta pivoting. With the jagged component, you can transform data into structured entities that can be loaded into database tables.

The jagged data component has one input group and multiple output groups, based on the configuration of the component.

The input group has two mandatory attributes: one each for name and value part of the incoming data set. A third, optional, attribute is used for row identifier sequences to delineate row sets.

To use a jagged component in a mapping:

1. Drag and drop the source data store into the logical diagram.
2. Drag and drop an jagged component from the component palette into the logical diagram.
3. The jagged input group requires two attribute level mappings; one for **name** and one for **value**.
4. Map one or more of the output groups to the downstream components.

   Note:

   In some cases, you may not need any additional attributes other than the default group: **others**.

5. Go to the physical diagram and assign new KMs if you want to.
6. Click **Save** and then execute the mapping to perform the flatten operation.

Creating a Mapping Using a Dataset

A dataset component is a container component that allows you to group multiple data sources and join them through relationship joins. A dataset can contain the following components:

- Datastores
- Joins
- Lookups
- Filters

- Reusable Mappings: Only reusable mappings with no input signature and one output signature are allowed.

Create Joins and lookups by dragging an attribute from one datastore to another inside the dataset. A dialog is shown to select if the relationship will be a join or lookup.

Note:

A driving table will have the key to look up, while the lookup table has additional information to add to the result.

In a dataset, drag an attribute from the driving table to the lookup table. An arrow will point from the driving table to the lookup table in the diagram.

By comparison, in a flow-based lookup (a lookup in a mapping that is not inside a dataset), the driving and lookup sources are determined by the order in which connections are created. The first connection is called `DRIVER_INPUT1`, the second connection `LOOKUP_INPUT1`.

Create a filter by dragging a datastore or reusable mapping attribute onto the dataset background. Joins, lookups, and filters cannot be dragged from the component palette into the dataset.

This section contains the following topics:

- Differences Between Flow and Dataset Modeling
- Creating a Dataset in a Mapping
- Converting a Dataset to Flow-Based Mapping

Differences Between Flow and Dataset Modeling

Datasets are container components which contain one or more source datastores, which are related using filters and joins. To other components in a mapping, a dataset is indistinguishable from any other projector component (like a datastore); the results of filters and joins inside the dataset are represented on its output port.

Within a dataset, data sources are related using relationships instead of a flow. This is displayed using an entity relationship diagram. When you switch to the physical tab of the mapping editor, datasets disappear: ODI models the physical flow of data exactly the same as if a flow diagram had been defined in the logical tab of the mapping editor.

Datasets mimic the ODI 11*g* way of organizing data sources, as opposed to the flow metaphor used in an ODI 12*c* mapping. If you import projects from ODI 11*g*, interfaces converted into mappings will contain datasets containing your source datastores.

When you create a new, empty mapping, you are prompted whether you would like to include an empty dataset. You can delete this empty dataset without harm, and you can always add an empty dataset to any mapping. The option to include an empty dataset is purely for your convenience.

A dataset exists only within a mapping or reusable mapping, and cannot be independently designed as a separate object.

Creating a Dataset in a Mapping

To create a dataset in a mapping, drag a dataset from the component palette into the logical diagram. You can then drag datastores into the dataset from the Models section of the Designer Navigator. Drag attributes from one datastore to another within a dataset to define join and lookup relationships.

Drag a connection from the dataset's output connector point to the input connector point on other components in your mapping, to integrate it into your data flow.

See Also:

To create a Join or Lookup inside a Dataset, see the Creating a Join or Lookup section in Creating Joins and Lookups

Converting a Dataset to Flow-Based Mapping

You can individually convert datasets into a flow-based mapping diagram, which is merged with the parent mapping flow diagram.

The effect of conversion of a dataset into a flow is the permanent removal of the dataset, together with the entity relationship design. It is replaced by an equivalent flow-based design. The effect of the conversion is irreversible.

To convert a dataset into a flow-based mapping:

1. Select the dataset in the mapping diagram.
2. Right click on the title and select **Convert to Flow** from the context menu.
3. A warning and confirmation dialog is displayed. Click **Yes** to perform the conversion, or click **No** to cancel the conversion.

   The dataset is converted into flow-based mapping components.

Physical Design

The physical tab shows the distribution of execution among different execution units that represent physical servers. ODI computes a default physical mapping design containing execution units and groups based on the logical design, the topology of those items and any rules you have defined.

You can also customize this design by using the physical diagram. You can use the diagram to move components between execution units, or onto the diagram background, which creates a separate execution unit. Multiple execution units can be grouped into execution groups, which enable parallel execution of the contained execution units.

A mapping can have multiple physical mapping designs; they are listed in tabs under the diagram. By having multiple physical mapping designs you can create different execution strategies for the same mapping.

To create new physical mapping tabs, click the **Create New** tab.

To delete physical mapping designs, right-click on the physical mapping design tab you want to delete, and select **Delete** from the context menu.

Physical components define how a mapping is executed at runtime; they are the physical representation of logical components. Depending on the logical component a physical component might have a different set of properties.

This section contains the following topics:

- About the Physical Mapping Diagram
- Selecting LKMs, IKMs and CKMs
- Configuring Execution Locations
- Adding Commands to be Executed Before and After a Mapping
- Configuring In-Session Parallelism
- Configuring Parallel Target Table Load
- Configuring Temporary Indexes
- Configuring Journalizing
- Configuring Extraction Options
- Creating and Managing Physical Mapping Designs

About the Physical Mapping Diagram

In the physical diagram, the following items appear:

- **Physical Mapping Design**: The entire physical diagram represents one physical mapping design. Click the background or select the white tab with the physical mapping design label to display the physical mapping properties. By default, the staging location is colocated on the target, but you can explicitly select a different staging location to cause ODI to automatically move staging to a different host.

  You can define additional physical mapping designs by clicking the small tab at the bottom of the physical diagram, next to the current physical mapping design tab. A new physical mapping design is created automatically from the logical design of the mapping.

- **Execution Groups**: Yellow boxes display groups of objects called execution units, which are executed in parallel within the same execution group. These are usually Source Groups and Target Groups:
  - **Source Execution Group(s)**: Source Datastores that are within the same dataset or are located on the same physical data server are grouped in a single source execution group in the physical diagram. A source execution group represents a group of datastores that can be extracted at the same time.
  - **Target Execution Group(s)**: Target Datastores that are located on the same physical data server are grouped in a single target execution group in the physical diagram. A target execution group represents a group of datastores that can be written to at the same time.
- **Execution Units**: Within the yellow execution groups are blue boxes called execution units. Execution units within a single execution group are on the same physical data server, but may be different structures.

- **Access Points**: In the target execution group, whenever the flow of data goes from one execution unit to another there is an access point (shown with a round icon). Loading Knowledge Modules (LKMs) control how data is transferred from one execution unit to another.

  An access point is created on the target side of a pair of execution units, when data moves from the source side to the target side (unless you use Execute On Hint in the logical diagram to suggest a different execution location). You cannot move an access point node to the source side. However, you can drag an access point node to the empty diagram area and a new execution unit will be created, between the original source and target execution units in the diagram.
- **Components**: mapping components such as joins, filters, and so on are also shown on the physical diagram.

You use the following knowledge modules (KMs) in the physical tab:

- **Loading Knowledge Modules (LKMs)**: LKMs define how data is moved. One LKM is selected for each access point for moving data from the sources to a staging area. An LKM can be also selected to move data from a staging area not located within a target execution unit, to a target, when a single technology IKM is selected for the staging area. Select an access point to define or change its LKM in the property inspector.
- **Integration Knowledge Modules (IKMs) and Check Knowledge Modules (CKMs)**: IKMs and CKMs define how data is integrated into the target. One IKM and one CKM is typically selected on a target datastore. When the staging area is different from the target, the selected IKM can be a multi-technology IKM that moves and integrates data from the staging area into the target. Select a target datastore to define or change its IKM and CKM in the property inspector.

Note:

- Only built-in KMs, or KMs that have already been imported into the project or the global KM list, can be selected in the mapping. Make sure that you have imported the appropriate KMs in the project before proceeding.
- For more information on the KMs and their options, refer to the KM description and to the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator*.

Selecting LKMs, IKMs and CKMs

ODI automatically selects knowledge modules in the physical diagram as you create your logical diagram.

Note:

The **Integration Type** property of a target datastore (which can have the values `Control Append`, `Incremental Update`, or `Slowly Changing Dimension`) is referenced by ODI when it selects a KM. This property is also used to restrict the IKM selection shown, so you will only see IKMs listed that are applicable.

You can use the physical diagram to change the KMs in use.

To change the LKM in use:

1. In the physical diagram, select an access point. The Property Inspector opens for this object.
2. Select the **Loading Knowledge Module** tab, and then select a different LKM from the **Loading Knowledge Module** list.
3. KMs are set with default options that work in most use cases. You can optionally modify the KM Options.

   Note:

   If an identically-named option exists, when switching from one KM to another KM options of the previous KM are retained. However, options that are not duplicated in the new KM are lost.

To change the IKM in use:

Note:

In order to use a multi-connect IKM on the target node, you must select LKM SQL Multi-Connect, or no LKM, for the access point of that execution unit. If another LKM is selected, only mono-connect IKMs are selectable.

1. In the physical diagram, select a target datastore by clicking its title. The Property Inspector opens for this object.
2. In the Property Inspector, select the **Integration Knowledge Module** tab, and then select an IKM from the **Integration Knowledge Module** list.
3. KMs are set with default options that work in most use cases. You can optionally modify the KM Options.

   Note:

   If an identically-named option exists, when switching from one KM to another KM options of the previous KM are retained. However, options that are not duplicated in the new KM are lost.

To change the CKM in use:

1. In the physical diagram, select a target datastore by clicking its title. The Property Inspector opens for this object.
2. In the Property Inspector, select the **Check Knowledge Module** tab, and then select a CKM from the **Check Knowledge Module** list.
3. KMs are set with default options that work in most use cases. You can optionally modify the KM Options.

> **Note:**
>
> If an identically-named option exists, when switching from one KM to another KM options of the previous KM are retained. However, options that are not duplicated in the new KM are lost.

Configuring Execution Locations

In the physical tab of the mapping editor, you can change the staging area and determine where components will be executed. When you designed the mapping using components in the logical diagram, you optionally set preferred execution locations using the Execute On Hint property. In the physical diagram, ODI attempts to follow these hints where possible.

You can further manipulate execution locations in the physical tab. See the following topics for details:

- Moving Physical Nodes
- Moving Expressions
- Defining New Execution Units

Moving Physical Nodes

You can move the execution location of a physical node. Select the node and drag it from one Execution Group into another Execution Group. Or, drag it to a blank area of the physical diagram, and ODI will automatically create a new Execution Group for the component.

You can change the order of execution of certain components only. The following components can be reordered on the physical diagram:

- Expressions
- Filters
- Joins
- Lookups

Note:

An inner input Connector Point of an outer join is an input whose data source contributes only matched rows to the output data set. For example - In ANSI SQL, 'A LEFT OUTER JOIN B' signifies that B corresponds to the inner input. In Oracle outer join syntax, 'WHERE A.col1 = B.col2 (+)' signifies that B corresponds to the inner input. A physical node can be reordered around an outer join only if it does not cause a change in the nodes that are connected to the inner input Connector Points of the outer join.

Moving Expressions

You can move expressions in the physical diagram. Select the Execution Unit and in the property inspector, select the **Expressions** tab. The execution location of the expression is shown in the **Execute on** property. Double-click the property to alter the execution location.

Defining New Execution Units

You can define a new execution unit by dragging a component from its current execution unit onto a blank area of the physical diagram. A new execution unit and group is created. Select the execution unit to modify its properties using the property inspector.

Adding Commands to be Executed Before and After a Mapping

ODI allows the addition of commands to be executed before and after a mapping. These commands can be in ODI-supported languages such as SQL, Jython, Groovy, and others. In the SQL language the Begin Mapping and End Mapping commands are executed in the same transaction as the mapping. The physical design of a mapping has the following properties to control this behavior:

Property	Description
Begin Mapping Command	Command to be executed at the beginning of the mapping.
Technology for Begin Mapping Command	Technology that this command will be executed with.
Location for Begin Mapping Command	Logical Schema that this command will be executed in.
End Mapping Command	Command to be executed at the end of the mapping.
Technology for End Mapping Command	Technology that this command will be executed with.
Location for End Mapping Command	Logical Schema that this command will be executed in.

You can view and set these properties from the Property Inspector by selecting a Physical Mapping Design.

Configuring In-Session Parallelism

ODI agent is the scheduler that runs an entire ODI mapping job on a given host. If your have two or more loads, it will either run them one after another (serialized), or simultaneously (parallelized, using separate processor threads).

Execution units in the same execution group are parallelized. If you move an execution unit into its own group, it is no longer parallelized with other execution units: it is now serialized. The system will select the order in which separate execution groups are run.

You might choose to run loads serially to reduce instantaneous system resource usage, while you might choose to run loads in parallel to reduce the longevity of system resource usage.

Configuring Parallel Target Table Load

You can enable parallel target table loading in a physical mapping design. Select the physical mapping design (by clicking on the tab at the bottom of the physical diagram, or clicking an empty area of the diagram) and in the property inspector, check the box for the property **Use Unique Temporary Object Names**.

This option allows multiple instances of the same mapping to be executed concurrently. To load data from source to staging area, `C$` tables are created in the staging database.

Note:

In ODI 11*g*, `C$` table names were derived from the target table of the interface. As a result, when multiple instances of the same mapping were executed at the same time, data from different sessions could load into the same `C$` table and cause conflicts.

In ODI 12*c*, if the option **Use Unique Temporary Object Names** is set to true, the system generates a globally-unique name for `C$` tables for each mapping execution. This prevents any conflict from occurring.

Configuring Temporary Indexes

If you want ODI to automatically generate a temporary index to optimize the execution of a filter, join, or datastore, select the node in the physical diagram. In the property inspector, select the **Temporary Indexes** tab. You can double-click the **Index Type** field to select a temporary index type.

Note:

The creation of temporary indexes may be a time consuming operation in the overall flow. Oracle recommends reviewing execution statistics and comparing the execution time saved by the indexes to the time spent creating them.

Configuring Journalizing

A source datastore can be configured in the physical diagram to use journalized data only. This is done by enabling **Journalized Data Only** in the **General** properties of a source datastore. The check box is only available if the referenced datastore is added to CDC in the model navigator.

Only one datastore per mapping can have journalizing enabled.

For more information about journalizing, see the Using Journalizing chapter in *Developing Integration Projects with Oracle Data Integrator*.

Configuring Extraction Options

Each component in the physical diagram, excluding access points and target datastores, has an **Extraction Options** tab in the property inspector. Extraction options influence the way that SQL is generated for the given component. Most components have an empty list of extraction options, meaning that no further configuration of the SQL generation is supported.

Extraction options are driven by the Extract Knowledge Module (XKM) selected in the **Advanced** sub-tab of the **Extract Options** tab. XKMs are part of ODI and cannot be created or modified by the user.

Creating and Managing Physical Mapping Designs

The entire physical diagram represents one physical mapping design. Click the background or select the white tab with the physical mapping design label to display the physical mapping properties for the displayed physical mapping design.

You can define additional physical mapping designs by clicking the small tab at the bottom of the physical diagram, next to the current physical mapping design tab(s). A new physical mapping design is created automatically, generated from the logical design of the mapping. You can modify this physical mapping design, and save it as part of the mapping.

For example, you could use one physical mapping design for your initial load, and another physical mapping design for incremental load using changed data capture (CDC). The two physical mapping designs would have different journalizing and knowledge module settings.

As another example, you could use different optimization contexts for each physical mapping design. Each optimization context represents a slightly different users' topology. One optimization context can represent a development environment, and another context represents a testing environment. You could select different KMs appropriate for these two different topologies.

Reusable Mappings

Reusable mappings allow you to encapsulate a multi-step integration (or portion of an integration) into a single component, which you can save and use just as any other components in your mappings. Reusable mappings are a convenient way to avoid the labor of creating a similar or identical subroutine of data manipulation that you will use many times in your mappings.

For example, you could load data from two tables in a join component, pass it through a filter component, and then a distinct component, and then output to a target datastore. You could then save this procedure as a reusable mapping, and place it into future mappings that you create or modify.

After you place a reusable mapping component in a mapping, you can select it and make modifications to it that only affect the current mapping.

Reusable mappings consist of the following:

- **Input Signature and Output Signature components**: These components describe the attributes that will be used to map into and out of the reusable mapping. When the reusable mapping is used in a mapping, these are the attributes that can be matched by other mapping components.
- **Regular mapping components**: Reusable mappings can include all of the regular mapping components, including datastores, projector components, and selector components. You can use these exactly as in regular mappings, creating a logical flow.

By combining regular mapping components with signature components, you can create a reusable mapping intended to serve as a data source, as a data target, or as an intermediate step in a mapping flow. When you work on a regular mapping, you can use a reusable mapping as if it were a single component.

Creating a Reusable Mapping

You can create a reusable mapping within a project, or as a global object. To create a reusable mapping, perform the following steps:

1. From the designer navigator:

   Open a project, right-click Reusable Mappings, and select New Reusable Mapping.

   Or, expand the Global Objects tree, right click Global Reusable Mappings, and select New Reusable Mapping.
2. Enter a name and, optionally, a description for the new reusable mapping. Optionally, select Create Default Input Signature and/or Create Default Output Signature. These options add empty input and output signatures to your reusable mapping; you can add or remove input and output signatures later while editing your reusable mapping.

Note:

In order to make use of these signatures, you will need to connect them to your reusable mapping flow.

3. Drag components from the component palette into the reusable mapping diagram, and drag datastores and other reusable mappings from the designer navigator, to assemble your reusable mapping logic. Follow all of the same processes as for creating a normal mapping.

Note:

When you have a reusable mapping open for editing, the component palette contains the Input Signature and Output Signature components in addition to the regular mapping components.

4. Validate your reusable mapping by clicking the **Validate the Mapping** button (a green check mark icon). Any errors will be displayed in a new error pane.

   When you are finished creating your reusable mapping, click **File** and select **Save**, or click the **Save** button, to save your reusable mapping. You can now use your reusable mapping in your mapping projects.

Editing Mappings Using the Property Inspector and the Structure Panel

You can use the Property Inspector with the Structure Panel to perform the same actions as on the logical and physical diagrams of the mapping editor, in a non-graphical form.

Using the Structure Panel

When creating and editing mappings without using the logical and physical diagrams, you will need to open the Structure Panel. The Structure Panel provides an expandable tree view of a mapping, which you can traverse using the tab keys, allowing you to select the components of your mapping. When you select a component or attribute in the Structure Panel, its properties are shown in the Property Inspector exactly the same as if you had selected the component in the logical or physical diagram.

The Structure Panel is useful for accessibility requirements, such as when using a screen reader.

To open the structure panel, select **Window** from the main menu and then click **Structure**. You can also open the Structure Panel using the hotkey **Ctrl+Shift-S**.

This section contains the following topics:

- Adding and Removing Components
- Editing a Component
- Customizing Tables

- Using Keyboard Navigation for Common Tasks

Adding and Removing Components

With the Property Inspector, the Component Palette, and the Structure Panel, you can add or remove components of a mapping.

Adding Components

To add a component to a mapping with the Component Palette and the Structure Panel:

1. With the mapping open in the Mapping Editor, open the Component Palette.
2. Select the desired component using the Tab key, and hit Enter to add the selected component to the mapping diagram and the Structure Panel.

Removing Components

To remove a component with the Structure Panel:

1. In the Structure Panel, select the component you want to remove.
2. While holding down Ctrl+Shift, hit Tab to open a pop-up dialog. Keep holding down Ctrl+Shift, and use the arrow keys to navigate to the left column and select the mapping. You can then use the right arrow key to select the logical or physical diagram. Release the Ctrl+Shift keys after you select the logical diagram.

   Alternatively, select **Windows** > **Documents...** from the main menu bar. Select the mapping from the list of document windows, and click **Switch to Document**.
3. The component you selected in the Structure Panel in step 1 is now highlighted in the mapping diagram. Hit Delete to delete the component. A dialog box confirms the deletion.

Editing a Component

To edit a component of a mapping using the Structure Panel and the Property Inspector:

1. In the Structure Panel, select a component. The component's properties are shown in the Property Inspector.
2. In the Property Inspector, modify properties as needed. Use the Attributes tab to add or remove attributes. Use the Connector Points tab to add connections to other components in your mapping.
3. Expand any component in the Structure Panel to list individual attributes. You can then select individual attributes to show their properties in the Property Inspector.

Customizing Tables

There are two ways to customize the tables in the Property Inspector to affect which columns are shown. In each case, open the Structure Panel and select a component to display its properties in the Property Inspector. Then, select a tab containing a table and use one of the following methods:

- From the table toolbar, click the **Select Columns...** icon (on the top right corner of the table) and then, from the drop down menu, select the columns to display in the table. Currently displayed columns are marked with a check mark.
- Use the Customize Table Dialog:
  1. From the table toolbar, click **Select Columns...**.
  2. From the drop down menu, select **Select Columns...**
  3. In the **Customize Table Dialog**, select the columns to display in the table.
  4. Click **OK**.

Using Keyboard Navigation for Common Tasks

This section describes the keyboard navigation in the Property Inspector.

The table below shows the common tasks and the keyboard navigation used in the Property Inspector.

Table 13-8 Keyboard Navigation for Common Tasks

Navigation	Task
Arrow keys	Navigate: move one cell up, down, left, or right
TAB	Move to next cell
SHIFT+TAB	Move to previous cell
SPACEBAR	Start editing a text, display items of a list, or change value of a checkbox
CTRL+C	Copy the selection
CTRL+V	Paste the selection
ESC	Cancel an entry in the cell
ENTER	Complete a cell entry and move to the next cell or activate a button
DELETE	Clear the content of the selection (for text fields only)
BACKSPACE	Delete the content of the selection or delete the preceding character in the active cell (for text fields only)
HOME	Move to the first cell of the row
END	Move to the last cell of the row
PAGE UP	Move up to the first cell of the column
PAGE DOWN	Move down to the last cell of the column

Flow Control and Static Control

In a mapping, it is possible to set two points of control. Flow Control checks the data in the incoming flow before it gets integrated into a target, and Static Control checks constraints on the target datastore after integration.

IKMs can have options to run `FLOW_CONTROL` and to run `STATIC_CONTROL`. If you want to enable either of these you must set the option in the IKM, which is a property set on the target datastore. In the physical diagram, select the datastore, and select the **Integration Knowledge Module** tab in the property inspector. If flow control options are available, they are listed in the Options table. Double-click an option to change it.

Note:

- Flow control is not supported for component KMs like IKM Oracle Insert. For more information, see the Knowledge Modules section in *Connectivity and Knowledge Modules Guide for Oracle Data Integrator Developer's Guide*. The description of each IKM indicates if it supports flow control.
- In ODI 11*g* the CKM to be used when flow or static control is invoked was defined on the interface. ODI 12*c* supports multiple targets on different technologies within the same mapping, so the CKM is now defined on each target datastore

This section contains the following topics:

- Setting up Flow Control
- Setting up Static Control
- Defining the Update Key

Setting up Flow Control

The flow control strategy defines how data is checked against the constraints defined on a target datastore before being integrated into this datastore. It is defined by a Check Knowledge Module (CKM). The CKM can be selected on the target datastore physical node. The constraints that checked by a CKM are specified in the properties of the datastore component on the logical tab.

To define the CKM used in a mapping, see: Selecting LKMs, IKMs and CKMs.

Setting up Static Control

The post-integration control strategy defines how data is checked against the constraints defined on the target datastore. This check takes place once the data is integrated into the target datastore. It is defined by a CKM. In order to have the post-integration control running, you must set the `STATIC_CONTROL` option in the IKM to `true`. Post-integration control requires that a primary key is defined in the data model for the target datastore of your mapping.

The settings **Maximum Number of Errors Allowed** and **Integration Errors as Percentage** can be set on the target datastore component. Select the datastore in the logical diagram, and in the property inspector, select the **Target** tab.

Post-integration control uses the same CKM as flow control.

Defining the Update Key

If you want to use update or flow control features in your mapping, it is necessary to define an update key on the target datastore.

The update key of a target datastore component contains one or more attributes. It can be the unique key of the datastore that it is bound to, or a group of attributes that are marked as the key attribute. The update key identifies each record to update or check before insertion into the target.

To define the update key from a unique key:

1. In the mapping diagram, select the header of a target datastore component. The component's properties will be displayed in the Property Inspector.
2. In the **Target** properties, select an **Update Key** from the drop down list.

> **Note:**
>
> - The Target properties are only shown for datastores which are the target of incoming data. If you do not see the Target properties, your datastore does not have an incoming connection defined.
> - Only unique keys defined in the model for this datastore appear in this list.

You can also define an update key from the attributes if:

- You don't have a unique key on your datastore.
- You want to specify the key regardless of already defined keys.

When you define an update key from the attributes, you select manually individual attributes to be part of the update key.

To define the update key from the attributes:

1. Unselect the update key, if it is selected.
2. In the Target Datastore panel, select one of the attributes that is part of the update key to display the Property Inspector.
3. In the Property Inspector, under **Target** properties, check the **Key** box. A key symbol appears in front of the key attribute(s) in the datastore component displayed in the mapping editor logical diagram.
4. Repeat the operation for each attribute that is part of the update key.

Designing E-LT and ETL-Style Mappings

> **See Also:**
>
> E-LT and ETL are defined and described in the What is E-LT section in *Understanding Oracle Data Integrator*.

In an E-LT-style integration mapping, ODI processes the data in a staging area, which is located on the target. Staging area and target are located on the same RDBMS. The data is loaded from the source(s) to the target. To create an E-LT-style integration mapping, follow the standard procedure described in Creating a Mapping.

In an ETL-style mapping, ODI processes the data in a staging area, which is different from the target. The data is first extracted from the source(s) and then loaded to the staging area. The data transformations take place in the staging area and the intermediate results are stored in temporary tables in the staging area. The data loading and transformation tasks are performed with the standard ELT KMs.

Oracle Data Integrator provides two ways for loading the data from the staging area to the target:

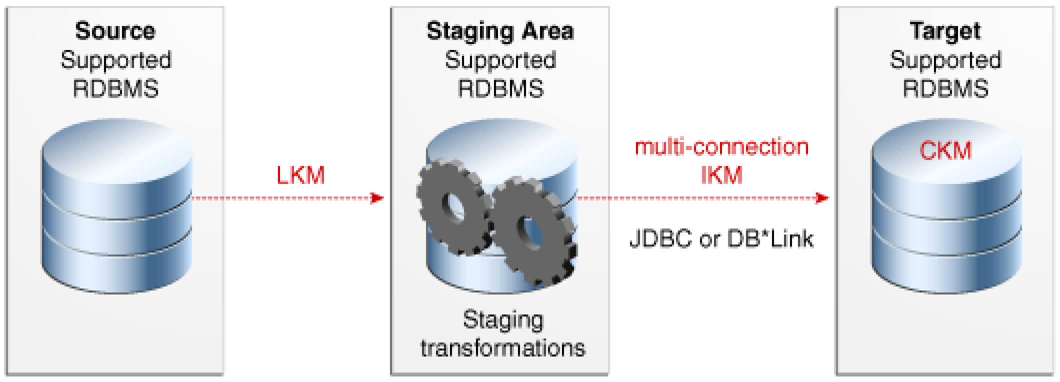
- Using a Multi-connection IKM
- Using an LKM and a mono-connection IKM

Depending on the KM strategy that is used, flow and static control are supported. See Designing an ETL-Style Mapping in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator Developer's Guide* for more information.

Using a Multi-connection IKM

A multi-connection IKM allows updating a target where the staging area and sources are on different data servers. The figure below shows the configuration of an integration mapping using a multi-connection IKM to update the target data.

Figure 13-1 ETL-Mapping with Multi-connection IKM



See the chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* that corresponds to the technology of your staging area for more information on when to use a multi-connection IKM.

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

1. Create a mapping using the standard procedure as described in Creating a Mapping. This section describes only the ETL-style specific steps.
2. In the Physical tab of the Mapping Editor, select a physical mapping design by clicking the desired physical mapping design tab and clicking on the diagram background. In the property inspector, the field **Preset Staging Location** defines the staging location. The empty entry specifies the target schema as staging location. Select a different schema as a staging location other than the target.
3. Select an Access Point component in the physical schema and go to the property inspector. For more information about Access Points, see: About the Physical Mapping Diagram.

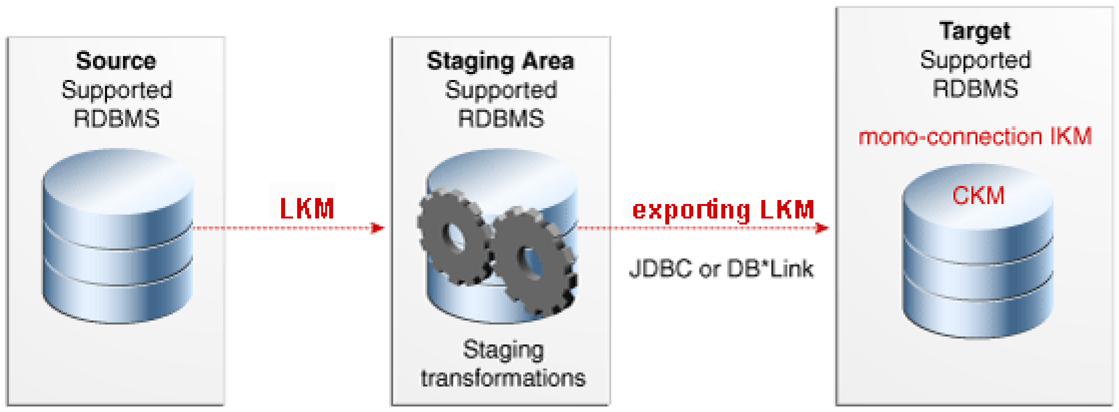
4. Select an LKM from the LKM Selector list to load from the source(s) to the staging area. See the chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* that corresponds to the technology of your staging area to determine the LKM you can use.
5. Optionally, modify the KM options.
6. In the Physical diagram, select a target datastore. The property inspector opens for this target object.

   In the Property Inspector, select an ETL multi-connection IKM from the IKM Selector list to load the data from the staging area to the target. See the chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* that corresponds to the technology of your staging area to determine the IKM you can use.
7. Optionally, modify the KM options.

Using an LKM and a mono-connection IKM

If there is no dedicated multi-connection IKM, use a standard exporting LKM in combination with a standard mono-connection IKM. The figure below shows the configuration of an integration mapping using an exporting LKM and a mono-connection IKM to update the target data. The exporting LKM is used to load the flow table from the staging area to the target. The mono-connection IKM is used to integrate the data flow into the target table.

Figure 13-2 ETL-Mapping with an LKM and a Mono-connection IKM



Note that this configuration (LKM + exporting LKM + mono-connection IKM) has the following limitations:

- Neither simple CDC nor consistent CDC are supported when the source is on the same data server as the staging area (explicitly chosen in the Mapping Editor)
- Temporary Indexes are not supported

See the chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* that corresponds to the technology of your staging area for more information on when to use the combination of a standard LKM and a mono-connection IKM.

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

1. Create a mapping using the standard procedure as described in Creating a Mapping. This section describes only the ETL-style specific steps.
2. In the Physical tab of the Mapping Editor, select a physical mapping design by clicking the desired physical mapping design tab and clicking on the diagram background. In the property inspector, the field **Preset Staging Location** defines the staging location. The empty entry specifies the target schema as staging location. Select a different schema as a staging location other than the target.
3. Select an Access Point component in the physical schema and go to the property inspector. For more information about Access Points, see: About the Physical Mapping Diagram.
4. In the Property Inspector, in the **Loading Knowledge Module** tab, select an LKM from the **Loading Knowledge Module** drop-down list to load from the source(s) to the staging area. See the chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* that corresponds to the technology of your staging area to determine the LKM you can use.
5. Optionally, modify the KM options. Double-click a cell in the **Value** column of the options table to change the value.
6. Select the access point node of a target execution unit. In the Property Inspector, in the **Loading Knowledge Module** tab, select an LKM from the **Loading Knowledge Module** drop-down list to load from the staging area to the target. See the chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* that corresponds to the technology of your staging area to determine the LKM you can use.
7. Optionally, modify the options.
8. Select the Target by clicking its title. The Property Inspector opens for this object.

   In the Property Inspector, in the **Integration Knowledge Module** tab, select a standard mono-connection IKM from the **Integration Knowledge Module** drop-down list to update the target. See the chapter in the *Connectivity and Knowledge Modules Guide for Oracle Data Integrator* that corresponds to the technology of your staging area to determine the IKM you can use.
9. Optionally, modify the KM options.

14 Creating and Using Packages

This chapter includes the following sections:

Introduction to Packages

The Package is a large unit of execution in Oracle Data Integrator. 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.

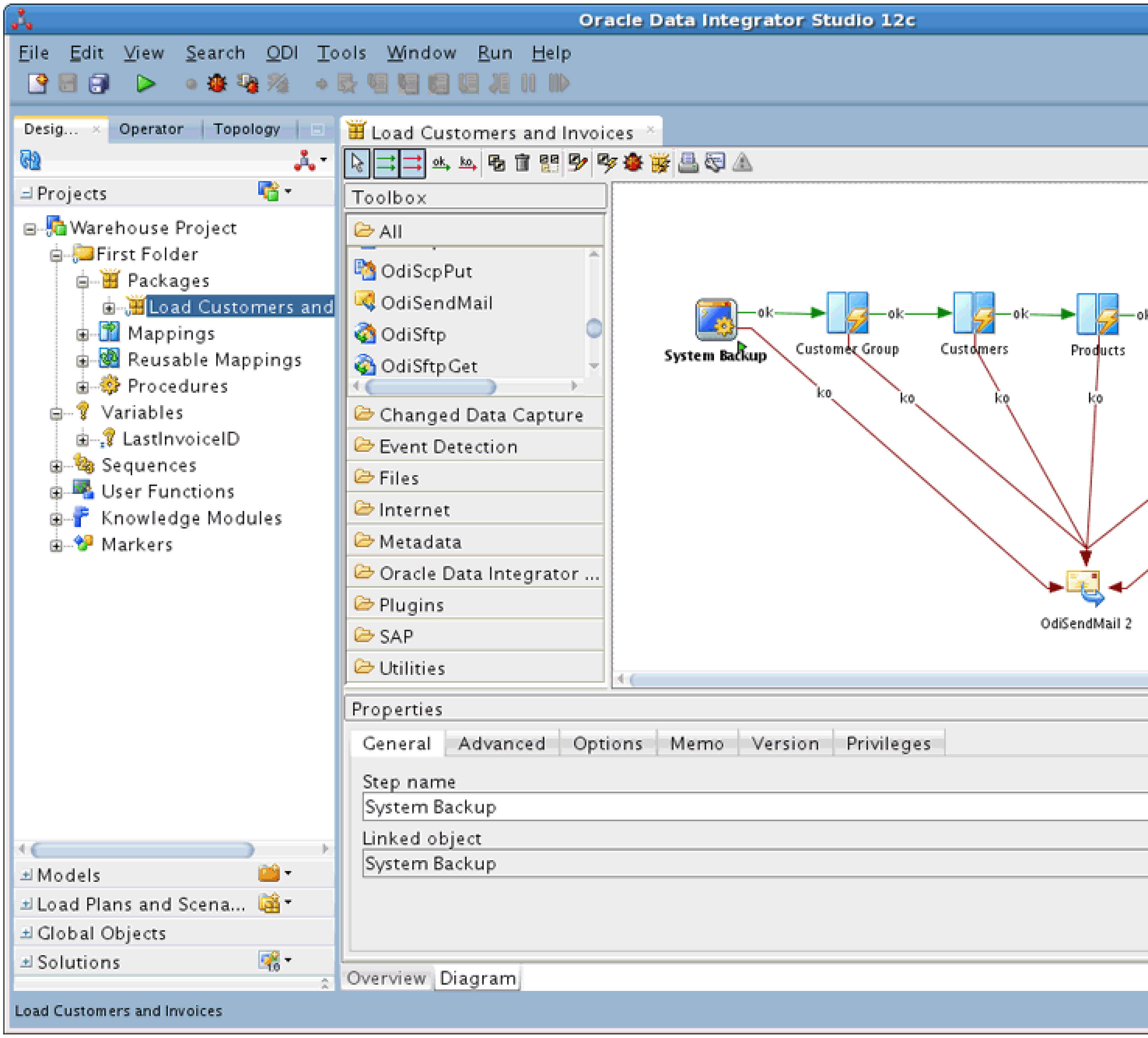
Introduction to Steps

The following table lists the different types of steps. References are made to sections that provide additional details.

Table 14-1 Step Types

Type	Description	See Section
Flow (Mapping)		Adding a Mapping step
Procedure	Executes a Procedure.	Adding a Procedure step
Variable	Declares, sets, refreshes or evaluates the value of a variable.	Variable Steps
Oracle Data Integrator Tools	These tools, available in the Toolbox, provide access to all Oracle Data Integrator API commands, or perform operating system calls.	Adding Oracle Data Integrator Tool Steps
Models, Sub-models, and Datastores	Performs journalizing, static check or reverse-engineering operations on these objects	Adding a Model, Sub-Model or Datastore

Figure 14-1 Sample Package



Step name: System Backup
Linked object: System Backup

For example, the "Load Customers and Invoice" Package example shown in the above figure performs the following actions:

1. Execute procedure "System Backup" that runs some backup operations.
2. Execute mapping "Customer Group" that loads the customer group datastore.
3. Execute mapping "Customer" that loads the customer datastore.
4. Execute mapping "Product" that loads the product datastore.

5. Refresh variable "Last Invoice ID" step to set the value of this variable for use later in the Package.
6. Execute mapping "Invoice Headers" that load the invoice header datastore.
7. Execute mapping "Invoices" that load the invoices datastore.
8. If any of the steps above fails, then the Package runs the "OdiSendMail 2" step that sends an email to the administrator using an Oracle Data Integrator tool.

Introduction to Creating Packages

Packages are created in the Package Diagram Editor. See Introduction to the Package editor for more information.

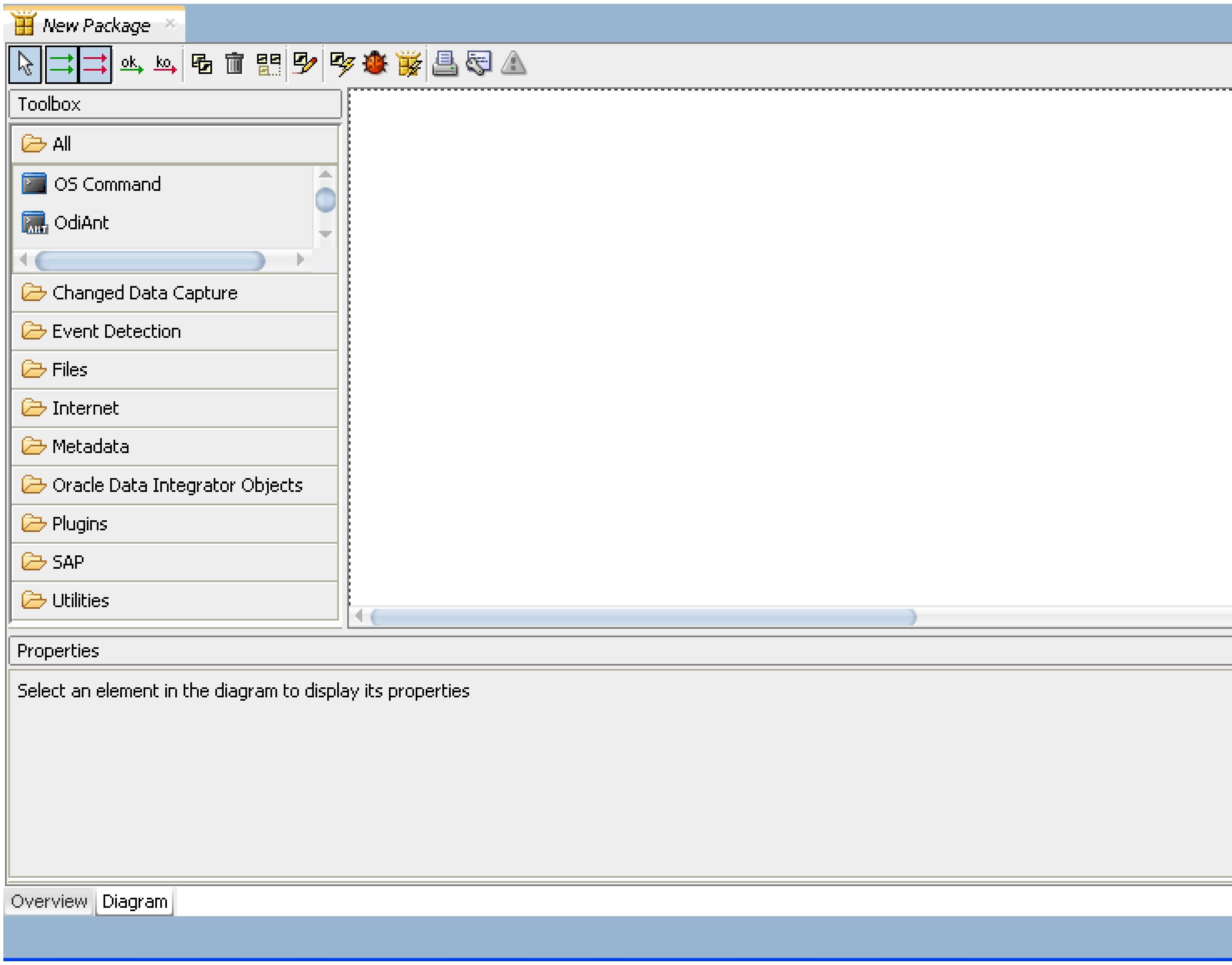
Creating a Package consists of the following main steps:

1. Creating a New Package. See Creating a new Package for more information.
2. Working with Steps in the Package (add, duplicate, delete, and so on). See Working with Steps for more information.
3. Defining Step Sequences. See Defining the Sequence of Steps for more information.
4. Running the Package. See Running a Package for more information.

Introduction to the Package editor

The Package editor provides a single environment for designing Packages. The figure below gives an overview of the Package editor.

Figure 14-2   Package editor



Select an element in the diagram to display its properties

Table 14-2   Package editor Sections

Section	Location in Figure	Description
Package Diagram	Middle	You drag components such as mappings, procedures, datastores, models, sub-models or variables from the Designer Navigator into the Package Diagram for creating steps for these components. You can also define sequence of steps and organize steps in this diagram.
Package Toolbox	Left side of the Package diagram	The Toolbox shows the list of Oracle Data Integrator tools available and that can be added to a Package. These tools are grouped by type.

Table 14-2 (Cont.) Package editor Sections

Section	Location in Figure	Description
Package Toolbar	Top of the Package diagram	The Package Toolbar provides tools for organizing and sequencing the steps in the Package.
Properties Panel	Under the Package diagram	This panel displays the properties for the object that is selected in the Package Diagram.

Creating a new Package

To create a new Package:

1. In the **Project** tree in Designer Navigator, click the **Packages** node in the folder where you want to create the Package.
2. Right-click and select **New Package**.
3. In the New Package dialog, type in the **Name**, and optionally a **Description**, of the Package. Click **OK**.
4. Use the **Overview** tab to set properties for the package.
5. Use the **Diagram** tab to design your package, adding steps as described in Working with Steps.
6. From the **File** menu, click **Save**.

Working with Steps

Packages are an organized sequence of steps. Designing a Package consists mainly in working with the steps of this Package.

Adding a Step

Adding a step depends on the nature of the steps being inserted. See Introduction to Steps for more information on the different types of steps. The procedures for adding the different type of steps are given below.

Adding a Mapping step

To insert a Mapping step:

1. Open the Package editor and go to the **Diagram** tab.
2. In the Designer Navigator, expand the project node and then expand the Mappings node, to show your mappings for this project.
3. Drag and drop a mapping into the diagram. A Flow (Mapping) step icon appears in the diagram.
4. Click the step icon in the diagram. The properties panel shows the mapping's properties.
5. In the properties panel, modify properties of the mapping as needed.

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

Adding a Procedure step

To insert a Procedure step:

1. Open the Package editor and go to the **Diagram** tab.
2. In the Designer Navigator, expand the project node and then expand the Procedures node, to show your procedures for this project.
3. Drag and drop a procedure into the diagram. A Procedure step icon appears in the diagram.
4. Click the step icon in the diagram. The properties panel shows the procedure's properties.
5. In the properties panel, modify properties of the procedure as needed.
6. From the **File** menu, click **Save**.

Variable Steps

There are different variable step types within Oracle Data Integrator:

- **Declare Variable**: When a variable is used in a Package (or in elements of the topology which are used in the Package), Oracle strongly recommends that you insert a Declare Variable step in the Package. This step explicitly declares the variable in the Package.
- **Refresh Variable**: This variable step refreshes the variable by running the query specified in the variable definition.
- **Set Variable**: There are two functions for this step:
  - **Assign** sets the current value of a variable.
  - **Increment** increases or decreases a numeric value by the specified amount.
- **Evaluate Variable**: This variable step type compares the value of the variable with a given value according to an operator. If the condition is met, then the evaluation step is true, otherwise it is false. This step allows for branching in Packages.

Adding a Variable step

To add a Variable step (of any type):

1. Open the Package editor and go to the **Diagram** tab.
2. In the Designer Navigator, expand the project node and then expand the Variables node, to show your variables for this project. Alternatively, expand the Global Objects node and expand the Variables node, to show global variables.
3. Drag and drop a variable into the diagram. A Variable step icon appears in the diagram.
4. Click the step icon in the diagram. The properties panel shows the variable's properties.
5. In the properties panel, modify properties of the variable as needed. On the **General** tab, select the variable type from the **Type** list.

- For Set Variables, select `Assign`, or `Increment` if the variable is of Numeric type. For Assign, type into the **Value** field the value to be assigned to the variable (this value may be another variable). For Increment, type into the **Increment** field a numeric constant by which to increment the variable.
- For Evaluate Variables, select the **Operator** used to compare the variable value. Type in the **Value** field the value to compare with your variable. This value may be another variable.

Note:

– You can specify a list of values in the Value field. When using the `IN` operator, use the semicolon character (`;`) to separate the values of a list.

– An evaluate variable step can be branched based on the evaluation result. See Defining the Sequence of Steps for more information on branching steps.

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

Adding Oracle Data Integrator Tool Steps

Oracle Data Integrator provides tools that can be used within Packages for performing simple operations. The tools are either built-in tools or Open Tools that enable you to enrich the data integrator toolbox.

To insert an Oracle Data Integrator Tool step:

1. Open the Package editor and go to the **Diagram** tab.
2. From the Package **Toolbox**, select the tool that you want to use. Note that Open tools appear in the **Plugins** group.
3. Click in the Package diagram. A step corresponding to your tool appears.

Tip:

As long as a tool is selected, left-clicking in the diagram will continue to place steps. To stop placing steps, click the **Free Choice** button in the Package Toolbar. The mouse pointer changes to an arrow, indicating you are no longer placing tools.

4. Click the step icon in the diagram. The properties panel shows the tool's properties.
5. Set the values for the parameters of the tool. The parameters descriptions appear when you select one, and are detailed in *Oracle Data Integrator Tool Reference*.
6. You can edit the code of this tool call in the **Command** tab.
7. From the **File** menu, click **Save**.

The following tools are frequently used in Oracle Data Integrator Package:

- **OdiStartScen**: starts an Oracle Data Integrator scenario synchronously or asynchronously. To create an OdiStartScen step, you can directly drag and drop the scenario from the Designer Navigator into the diagram.
- **OdiInvokeWebService**: invokes a web service and saves the response in an XML file. OdiInvokeWebService uses the HTTP Analyzer tool to set up and test the tool parameters. For more information, see the Using HTTP Analyzer section in *Developing Integration Projects with Oracle Data Integrator*.
- **OS Command**: calls an Operating System command. Using an operating system command may make your Package platform-dependent.

The Oracle Data Integrator tools are listed in *Oracle Data Integrator Tool Reference*.

Note:

When setting the parameters of a tool using the steps properties panel, graphical helpers allow value selection in a user-friendly manner. For example, if a parameter requires a project *identifier*, the graphical mapping will redesign it and display a list of project *names* for selection. By switching to the **Command** tab, you can review the raw command and see the identifier.

Adding a Model, Sub-Model or Datastore

You can perform journalizing, static check or reverse-engineering operations on models, sub-models, and datastores.

To insert a check, reverse engineer, or journalizing step in a Package:

Note:

- To perform a static check, you must define the CKM in the model.
- To perform journalizing operations, you must define the JKM in the model.
- Reverse engineering options set in the model definition are used for performing reverse-engineering processes in a package.

1. Open the Package editor and go to the **Diagram** tab.
2. In Designer Navigator, select the model, sub-model or datastore to check from the **Models** tree.
3. Drag and drop this model, sub-model or datastore into the diagram.
4. In the **General** tab of the properties panel, select the Type: `Check`, `Reverse Engineer`, or `Journalizing`.
   - For Check steps, select **Delete Errors from the Checked Tables** if you want this static check to remove erroneous rows from the tables checked in this process.

- For Journalizing steps, set the journalizing options. For more information on these options, see Using Journalizing in *Developing Integration Projects with Oracle Data Integrator*.

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

Deleting a Step

Caution:

It is not possible to undo a delete operation in the Package diagram.

To delete a step:

1. In the Package toolbar tab, select the **Free Choice** tool.
2. Select the step to delete in the diagram.
3. Right-click and then select **Delete Step**. Or, hit the Delete key on your keyboard.
4. Click **Yes** to continue.

The step disappears from the diagram.

Duplicating a Step

To duplicate a step:

1. In the Package toolbar tab, select the **Free Choice** tool.
2. Select the step to duplicate in the diagram.
3. Right-click and then select **Duplicate Step**.

A copy of the step appears in the diagram.

Running a Step

To run a step:

1. In the Package toolbar tab, select the **Free Choice** tool.
2. Select the step to run in the diagram.
3. Right-click and then select **Execute Step**.
4. In the **Run** dialog, select the execution parameters:
   - Select the **Context** into which the step must be executed.
   - Select the **Logical Agent** that will run the step.
   - Select a **Log Level**.
   - Optionally, select **Simulation**. This option performs a simulation of the run operation and generates a run report, without actually affecting data.
5. Click **OK**.
6. The **Session Started Window** appears.

7. Click **OK**.

You can review the step execution in the Operator Navigator.

Editing a Step's Linked Object

The step's linked object is the mapping, procedure, variable, or other object from which the step is created. You can edit this object from the Package diagram.

To edit a step's linked object:

1. In the Package toolbar tab, select the **Free Choice** tool.
2. Select the step to edit in the diagram.
3. Right-click and then select **Edit Linked Object**.

The Editor for the linked object opens.

Arranging the Steps Layout

The steps can be rearranged automatically or manually in the diagram in order to make it more readable.

You can use the **Reorganize** button from the toolbar to automatically reorganize all of the steps in your package.

To manually arrange the steps in the diagram:

1. From the Package toolbar menu, select the **Free Choice** tool.
2. Select the steps you wish to arrange using either of the following methods:
   - Keep the CTRL key pressed and select each step.
   - Drag a box around multiple items in the diagram with the left mouse button pressed.
3. To arrange the selected steps, you may either:
   - Drag them to arrange their position into the diagram
   - Right-click, then select a **Vertical Alignment** or **Horizontal Alignment** option from the context menu.

Defining the Sequence of Steps

Once the steps are created, you must order them into a data processing chain. This chain has the following rules:

- It starts with a unique step defined as the First Step.
- Each step has two termination states: Success or Failure.
- A step in failure or success can be followed by another step, or by the end of the Package.
- In case of failure, it is possible to define a number of retries.

A Package has one entry point, the First Step, but several possible termination steps.

Failure Conditions

The table below details the conditions that lead a step to a Failure state. In other situations, the steps ends in a Success state.

Note:

By default, open transactions are not rolled back in a failure state. You can change this behavior using the Physical Agent property "Rollback all open transactions on step failure". Refer to the ODI Studio Online Help for details.

Step Type	Failure conditions
Flow	• Error in a mapping command. • Maximum number or percentage of errors allowed reached.
Procedure	Error in a procedure command.
Refresh Variable	Error while running the refresh query.
Set Variable	Error when setting the variable (invalid value).
Evaluate Variable	The condition defined in the step is not matched.
Declare Variable	This step has no failure condition and always succeeds.
Oracle Data Integrator Tool	Oracle Data Integrator Tool return code is different from zero. If this tool is an OS Command, a failure case is a command return code different from zero.
Journalize Datastore, Model or Sub-Model	Error in a journalizing command.
Check Datastore, Model or Sub-Model	Error in the check process.
Reverse Model	Error in the reverse-engineering process.

Defining the Sequence

To define the first step of the Package:

1. In the Package toolbar tab, select the **Free Choice** tool.
2. Select the step to set as the first one in the diagram.
3. Right-click and then select **First Step**.

The first step symbol appears on the step's icon.

To define the next step upon success:

1. In the Package toolbar tab, select the **Next Step on Success** tool.
2. Drag a line from one step to another, using the mouse.
3. Repeat this operation to link all your steps in a success path sequence. This sequence should start from the step defined as the First Step.

Green arrows representing the success path are shown between the steps, with an *ok* labels on these arrows. In the case of an evaluate variable step, the label is *true*.

To define the next step upon failure:

1. In the Package toolbar tab, select the **Next Step on Failure** tool.
2. Drag a line from one step to another, using the mouse.
3. Repeat this operation to link steps according to your workflow logic.

Red arrows representing the failure path are shown between the steps, with a *ko* labels on these arrows. In the case of an evaluate variable step, the arrow is green and the label is *false*.

To define the end of the Package upon failure:

By default, a step that is linked to no other step after a success or failure condition will terminate the Package when this success or failure condition is met. You can set this behavior by editing the step's behavior.

1. In the Package toolbar tab, select the **Free Choice** tool.
2. Select the step to edit.
3. In the properties panel, select the **Advanced** tab.
4. Select **End** in **Processing after failure** or **Processing after success**. The links after the step disappear from the diagram.
5. You can optionally set a **Number of attempts** and a **Time between attempts** for the step to retry a number of times with an interval between the retries.

Running a Package

To run a Package:

1. Use any of the following methods:
   - In the **Projects** node of the Designer Navigator, expand a project and select the Package you want to execute. Right-click and select **Run**, or click the **Run** button in the ODI Studio toolbar, or select **Run** from the **Run** menu of the ODI menu bar.
   - In the package editor, select the package by clicking the tab with the package name at the top of the editor. Click the **Run** button in the ODI Studio toolbar, or select **Run** from the **Run** menu of the ODI menu bar.
2. In the **Run** dialog, select the execution parameters:
   - Select the **Context** into which the package must be executed.
   - Select the **Logical Agent** that will run the package.
   - Select a **Log Level**.
   - Optionally, select **Simulation.** This option performs a simulation of the run operation and generates a run report, without actually affecting data.
3. Click **OK**.
4. The **Session Started Window** appears.
5. Click **OK**.

You can review the Package execution in the Operator Navigator.

15 Using Scenarios

This chapter describes how to work with 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. This chapter includes the following sections:

Introduction to Scenarios

When a component is finished and tested, you can generate the **scenario** corresponding to its actual state. This operation takes place in the Designer Navigator.

The scenario code (the language generated) is frozen, and all subsequent modifications of the components which contributed to creating it will not change it in any way.

It is possible to generate scenarios for packages, procedures, mappings, or variables. Scenarios generated for procedures, mappings, or variables are single step scenarios that execute the procedure, mapping, or refresh the variable.

Scenario variables are variables used in the scenario that should be set when starting the scenario to parameterize its behavior.

Once generated, the scenario is stored inside the work repository. The scenario can be exported, and then imported to another repository (remote or not) and used in different contexts. A scenario can only be created from a development work repository, but can be imported into both development and execution work repositories.

Scenarios appear in both the Operator and Designer Navigators, in the Load Plans and Scenarios section. Scenarios can also appear within a project in the Projects section of the Designer navigator.

Scenarios can also be versioned. See the Using Version Control (Legacy Mode) chapter in *Developing Integration Projects with Oracle Data Integrator*, for more information.

Scenarios can be launched from a command line, from the Oracle Data Integrator Studio and can be scheduled using the built-in scheduler of the run-time agent or an external scheduler. Scenario execution and scheduling scenarios is covered in the Running Integration Processes chapter in *Administering Oracle Data Integrator*.

Generating a Scenario

Generating a scenario for an object compiles the code for this object for deployment and execution in a production environment.

To generate a scenario:

1. In Designer Navigator double-click the Package, Mapping, Procedure or Variable under the project for which you want to generate the scenario. The corresponding Object Editor opens. Then, on the **ODI** menu, select **Generate** and then **Scenario**. The New Scenario dialog appears.

   Alternatively, from the Designer Navigator, right-click a Package, Mapping, Procedure or Variable, and select **Generate Scenario...**. The New Scenario dialog appears.

2. Enter the **Name** and the **Version** of the scenario. As this name can be used in an operating system command, the name is automatically uppercased and special characters are replaced by underscores.

   Note that the **Name** and **Version** fields of the Scenario are preset with the following values:

   - **Name**: The same name as the latest scenario generated for the component
   - **Version**: The version number is automatically incremented, or set to `001` if no prior numeric version exists

   If no scenario has been created yet for the component, a first version of the scenario is automatically created.

   > **Note:**
   >
   > New scenarios are named after the component according to the **Scenario Naming Convention** user parameter. You can set this parameter by clicking **Preferences** from the **Tools** option on the menu bar; expand the **ODI** node, and then the **System** node, and select the **Scenarios** node.

3. Click **OK**.
4. If you use variables in the scenario, you can define in the Scenario Variables dialog the variables that will be considered as parameters for the scenario.
   - Select **Use All** if you want all variables to be parameters
   - Select **Use Selected** to use the selected variables to be parameters
   - Select **None** to deselect all variables
5. Click **OK**.

The scenario appears on the Scenarios tab and under the Scenarios node of the source object under the project.

Regenerating a Scenario

An existing scenario can be regenerated with the same name and version number. This lets you replace the existing scenario by a scenario generated from the source object contents. Schedules attached to this scenario are preserved.

To regenerate a scenario:

1. Select a scenario in the **Projects** or **Load Plans and Scenarios** section of the Designer Navigator.
2. Right-click and select **Regenerate...**
3. Click **OK**.

Caution:

Regenerating a scenario cannot be undone. For important scenarios, it is better to generate a scenario with a new version number.

Generating a Group of Scenarios

When a set of packages, mappings, procedures, and variables grouped under a project or folder is finished and tested, you can generate the scenarios. This operation takes place in Designer Navigator.

To generate a group of scenarios:

1. Select the Project or Folder containing the group of objects.
2. Right-click and select **Generate All Scenarios...**
3. In the **Scenario Source Objects** section, select the types of objects for which you want to generate scenarios.
4. In the **Marker Filter** section, you can filter the components to generate according to a marker from a marker group.
5. Select the scenario **Generation Mode**:
   - **Replace**: Overwrites for each object the last scenario version with a new one with the same internal ID, name and version. Sessions, scenario reports and schedules are deleted. If no scenario exists for an object, a scenario with version number 001 is created.
   - **Re-generate**: Overwrites for each object the last scenario version with a new one with the same internal ID, name and version. It preserves the schedule, sessions, scenario reports, variable selections, and concurrent execution control settings. If no scenario exists for an object, no scenario is created using this mode.
   - **Creation**: Creates for each object a new scenario with the same name as the last scenario version and with an automatically incremented version number. If no scenario exists for an object, a scenario named after the object with version number 001 is created.

> **Note:**
>
> If no scenario has been created yet for the component, a first version of the scenario is automatically created.
>
> New scenarios are named after the component according to the **Scenario Naming Convention** user parameter. You can set this parameter by clicking **Preferences** from the **Tools** option on the menu bar; expand the **ODI** node, and then the **System** node, and select the **Scenarios** node.
>
> When selecting the **Creation** generation mode, the version number is automatically incremented, or set to `001` if no prior numeric version exists.

- Select **Generate scenario as if all underlying objects are materialized** to generate the scenario as if the shortcuts were real objects.

6. Click **OK**.
7. If you use variables in the scenario, you can define in the Scenario Variables dialog the variables that will be considered as parameters for the scenario. Select **Use All** if you want all variables to be parameters, or **Use Selected** and check the parameter variables.

Controlling Concurrent Execution of Scenarios and Load Plans

By default, nothing prevents two instances of the same scenario or load plan from running simultaneously.

This situation could occur in several ways. For example:

- A load plan containing a Run Scenario Step is running in two or more instances, so the Run Scenario Step may be executed at the same time in more than one load plan instance.
- A scenario is run from the command line, from ODI Studio, or as scheduled on an agent, while another instance of the same scenario is already running (on the same or a different agent or ODI Studio session.

Concurrent executions of the same scenario or load plan apply across all remote and internal agents.

Concurrent execution of multiple instances of a scenario or load plan may be undesirable, particularly if the job involves writing data. You can control concurrent execution using the Concurrent Execution Control options.

ODI identifies a specific scenario or load plan by its internal ID, and not by the name and version. Thus, a regenerated or modified scenario or load plan having the same internal ID is still treated as the same scenario or load plan. Conversely, deleting a scenario and generating a new one with the same name and version number would be creating a different scenario (because it will have a different internal ID).

While Concurrent Execution Control can be enabled or disabled for a scenario or load plan at any time, there are implications to existing running sessions and newly invoked sessions:

- When switching Concurrent Execution Control from disabled to enabled, existing running and queued jobs are counted as executing jobs and new job submissions are processed with the Concurrent Execution Control settings at time of job submission.
- When switching Concurrent Execution Control from enabled to disabled for a scenario or load plan, jobs that are already submitted and in waiting state (or those that are restarted later) will carry the original Concurrent Execution Control setting values to consider and wait for running and queued jobs as executing jobs.

  However, if new jobs are submitted at that point with Concurrent Execution Control disabled, they could be run ahead of already waiting jobs. As a result, a waiting job may be delayed if, at the time of polling, the system finds executing jobs that were started without Concurrent Execution Control enabled. And, after a waiting job eventually starts executing, it may still be affected by uncontrolled jobs submitted later and executing concurrently.

To limit concurrent execution of a scenario or load plan, perform the following steps:

1. Open the scenario or load plan by right-clicking it in the Designer or Operator Navigators and selecting **Open**.
2. Select the Definition tab and modify the Concurrent Execution Controller options:
   - Enable the **Limit Concurrent Executions** check box if you do not want to allow multiple instances of this scenario or load plan to be run at the same time. If **Limit Concurrent Executions** is disabled (unchecked), no restriction is imposed and more than one instance of this scenario or load plan can be run simultaneously.
   - If **Limit Concurrent Executions** is enabled, set your desired **Violation Behavior**:
     – **Raise Execution Error**: if an instance of the scenario or load plan is already running, attempting to run another instance will result in a session being created but immediately ending with an execution error message identifying the session that is currently running which caused the Concurrent Execution Control error.

     – **Wait to Execute**: if an instance of the scenario or load plan is already running, additional executions will be placed in a wait status and the system will poll for its turn to run. The session's status is updated periodically to show the currently running session, as well as all concurrent sessions (if any) that are waiting in line to run after the running instance is complete.

       If you select this option, the **Wait Polling Interval** sets how often the system will check to see if the running instance has completed. You can only enter a **Wait Polling Interval** if **Wait to Execute** is selected.

       If you do not specify a wait polling interval, the default for the executing agent will be used: in ODI 12.1.3, the default agent value is 30 seconds.
3. Click **Save** to save your changes.

Exporting Scenarios

The export (and import) procedure allows you to transfer Oracle Data Integrator objects from one repository to another.

It is possible to export a single scenario or groups of scenarios.

Exporting one single scenario is covered in the Exporting one ODI Object section of *Developing Integration Projects with Oracle Data Integrator*.

To export a group of scenarios:

1. Select the Project or Folder containing the group of scenarios.
2. Right-click and select **Export All Scenarios...** The Export all scenarios dialog opens.
3. In the Export all scenarios dialog, specify the export parameters as follows:

Parameter	Description
Export Directory	Directory in which the export file will be created. Note that if the **Export Directory** is not specified, the export file is created in the Default Export Directory.
Child components export	If this option is checked, the objects linked to the object to be exported will be also exported. These objects are those visible under the exported object in the tree. It is recommended to leave this option checked. See Exporting an Object with its Child Components section of *Developing Integration Projects with Oracle Data Integrator* for more details.
Replace existing files without warning	If this option is checked, the existing file will be replaced by the ones of the export.

4. Select the type of objects whose scenarios you want to export.
5. Set the encryption options. Set an Export Key if you want the exported scenario to preserve and encrypt sensitive data such as passwords. You will need to supply this Export Key when you later import this scenario if you want to import and decrypt the sensitive data.
6. Set the advanced options. This set of options allow to parameterize the XML output file format. It is recommended that you leave the default values.
7.

Parameter	Description
XML Version	XML Version specified in the export file. Parameter xml version in the XML file header. `<?`**`xml version="1.0"`**` encoding="ISO-8859-1"?>`
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.

8. Click **OK**.

The XML-formatted export files are created at the specified location.

Importing Scenarios in Production

A scenario generated from Designer can be exported and then imported into a development or execution repository. This operation is used to deploy scenarios in a different repository, possibly in a different environment or site.

Importing a scenario in a development repository is performed with the Designer or Operator Navigator. With an execution repository, only the Operator Navigator is available for this purpose.

There are two ways to import a scenario:

- **Import** uses the standard object import method. During this import process, it is possible to choose to import the schedules attached to the exported scenario.
- **Import Replace** replaces an existing scenario with the content of an export file, preserving references from other objects to this scenario. Sessions, scenario reports and schedules from the original scenario are deleted and replaced with the schedules from the export file.

Scenarios can also be deployed and promoted to production using versions and solutions. See the Using Version Control (Legacy Mode) chapter in *Developing Integration Projects with Oracle Data Integrator*, for more information.

Import Scenarios

To import one or more scenarios into Oracle Data Integrator:

1. In Operator Navigator, select the **Scenarios** panel.
2. Right-click and select **Import** > **Import Scenario**.
3. Select the **Import Type**. Refer to the Exporting and Importing chapter in *Developing Integration Projects with Oracle Data Integrator* for more information on the import types.
4. Specify the **File Import Directory**.
5. Check the **Import schedules** option, if you want to import the schedules exported with the scenarios as well.
6. Select one or more scenarios to import from the **Select the file(s) to import** list.
7. Click **OK**.

The scenarios are imported into the work repository. They appear in the Scenarios tree of the Operator Navigator. If this work repository is a development repository, these scenario are also attached to their source Package, Mapping, Procedure, or Variable.

Replace a Scenario

Use the import replace mode if you want to replace a scenario with an exported one.

To import a scenario in replace mode:

1. In Designer or Operator Navigator, select the scenario you wish to replace.
2. Right-click the scenario, and select **Import Replace...**.

3. In the Replace Object dialog, specify the scenario export file.
4. Click **OK**.

Working with a Scenario from a Different Repository

A scenario may have to be operated from a different work repository than the one where it was generated.

Examples

Here are two examples of organizations that give rise to this type of process:

- A company has a large number of agencies equipped with the same software applications. In its IT headquarters, it develops packages and scenarios to centralize data to a central data center. These scenarios are designed to be executed identically in each agency.
- A company has three distinct IT environments for developing, qualifying, and operating its software applications. The company's processes demand total separation of the environments, which cannot share the Repository.

Prerequisites

The prerequisite for this organization is to have a work repository installed on each environment (site, agency, or environment). The topology of the master repository attached to this work repository must be compatible in terms of its logical architecture (the same logical schema names). The connection characteristics described in the physical architecture can differ.

Note that in cases where some procedures or mappings explicitly specify a context code, the target topology must have the same context codes. The topology, that is, the physical and logical architectures, can also be exported from a development master repository, then imported into the target repositories. Use the Topology module to carry out this operation. In this case, the physical topology (the servers' addresses) should be personalized before operating the scenarios. Note also that a topology import simply references the new data servers without modifying those already present in the target repository.

To operate a scenario from a different work repository:

1. Export the scenario from its original repository (right-click, export)
2. Forward the scenario export file to the target environment
3. Open Designer Navigator in the target environment (connection to the target repository)
4. Import the scenario from the export file

Encrypting and Decrypting a Scenario

Encrypting a scenario allows you to protect valuable code. An encrypted scenario can be executed but cannot be read or modified if it is not decrypted. The commands generated in the log by an encrypted scenario are also unreadable.

Oracle Data Integrator uses a DES Encryption algorithm based on a personal encryption key. This key can be saved in a file and can be reused to perform encryption or decryption operations.

WARNING:

There is no way to decrypt an encrypted scenario or procedure without the encryption key. It is therefore strongly advised to keep this key in a safe location.

To encrypt a scenario:

1. In Designer or Operator Navigator, select the scenario you want to encrypt.
2. Right-click and select **Encrypt**.
3. In the Encryption Options dialog, you can either:
   - **Encrypt with a personal key** that already exists by giving the location of the personal key file or by typing in the value of the personal key.
   - **Get a new encryption key** to have a new key generated.
4. Click **OK** to encrypt the scenario. If you have chosen to generate a new key, a dialog will appear with the new key. Click **Save** to save the key in a file.

Note:

If you type in a personal key with too few characters, an invalid key size error appears.

To decrypt a scenario:

1. Right-click the scenario you want to decrypt.
2. Select **Decrypt**.
3. In the **Scenario Decryption** dialog, either
   - Select an existing encryption key file
   - or type in (or paste) the string corresponding to your personal key.

A message appears when decryption is finished.

16

Using Load Plans

This chapter gives an introduction to Load Plans. It describes how to create a Load Plan and provides information about how to work with Load Plans.
This chapter includes the following sections:

Introduction to Load Plans

Oracle Data Integrator is often used for populating very large data warehouses. In these use cases, it is common to have thousands of tables being populated using hundreds of scenarios. The execution of these scenarios has to be organized in such a way that the data throughput from the sources to the target is the most efficient within the batch window. Load Plans help the user organizing the execution of scenarios in a hierarchy of sequential and parallel steps for these type of use cases.

A *Load Plan* is an executable object in Oracle Data Integrator that can contain a hierarchy of *steps* that can be executed conditionally, in parallel or in series. The leaf nodes of this hierarchy are *Scenarios*. Packages, mappings, variables, and procedures can be added to Load Plans for executions in the form of scenarios. For more information, see Creating a Load Plan.

Load Plans allow setting and using variables at multiple levels. See Working with Variables in Load Plans for more information. Load Plans also support exception handling strategies in the event of a scenario ending in error. See Handling Load Plan Exceptions and Restartability for more information.

Load Plans can be started, stopped, and restarted from a command line, from Oracle Data Integrator Studio, Oracle Data Integrator Console or a Web Service interface. They can also be scheduled using the run-time agent's built-in scheduler or 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*. See Running Load Plans for more information.

A Load Plan can be modified in production environments and steps can be enabled or disabled according to the 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 will not be available in an execution work repository. See Editing Load Plan Steps for more information.

Once created, a Load Plan is stored in the work repository. The Load Plan can be exported then imported to another repository and executed in different contexts. Load

Plans can also be versioned. See Exporting, Importing, and Versioning Load Plans for more information.

Load Plans appear in Designer Navigator and in Operator Navigator in the Load Plans and Scenarios accordion. The Load Plan Runs are displayed in the Load Plan Executions accordion in Operator Navigator.

Load Plan Execution Lifecycle

When running or scheduling a Load Plan you provide the variable values, the contexts and logical agents used for this Load Plan execution.

Executing a Load Plan creates a *Load Plan instance* and a first *Load Plan run*. This Load Plan instance is separated from the original Load Plan, and the Load Plan Run corresponds to the first attempt to execute this instance. If a run is restarted a new *Load Plan run* is created under this Load Plan instance. As a consequence, each execution attempt of the Load Plan Instance is preserved as a different Load Plan run in the Log.

See Running Load Plans for more information.

For a load plan instance, only one run can be running, and it must be the last load plan instance run. However, as with Scenarios, it is possible to run multiple instances of the same load plan (determined by the load plan's internal ID) concurrently, depending on the Concurrent Execution Control settings for the load plan.

For more information about how ODI handles concurrent execution, and about using the Concurrent Execution Control, see the section Controlling Concurrent Execution of Scenarios and Load Plans in *Developing Integration Projects with Oracle Data Integrator*.

Differences between Packages, Scenarios, and Load Plans

A *Load Plan* is the largest executable object in Oracle Data Integrator. It uses *Scenario*s in its steps. When an executable object is used in a Load Plan, it is automatically converted into a scenario. For example, a package is used in the form of a scenario in Load Plans. Note that Load Plans cannot be added to a Load Plan. However, it is possible to add a scenario in form of a Run Scenario step that starts another Load Plan using the OdiStartLoadPlan tool.

Load plans are not substitutes for packages or scenarios, but are used to organize at a higher level the execution of packages and scenarios.

Unlike packages, Load Plans provide native support for parallelism, restartability and exception handling. Load plans are moved to production as is, whereas packages are moved in the form of scenarios. Load Plans can be created in Production environments.

The *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 is added after the previous Load Plan Runs for this Load Plan Instance. Note that the Load Plan Instance cannot be modified at run-time.

Load Plan Structure

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 table below lists the different types of Load Plan steps and the possible child steps.

Table 16-1 Load Plan Steps

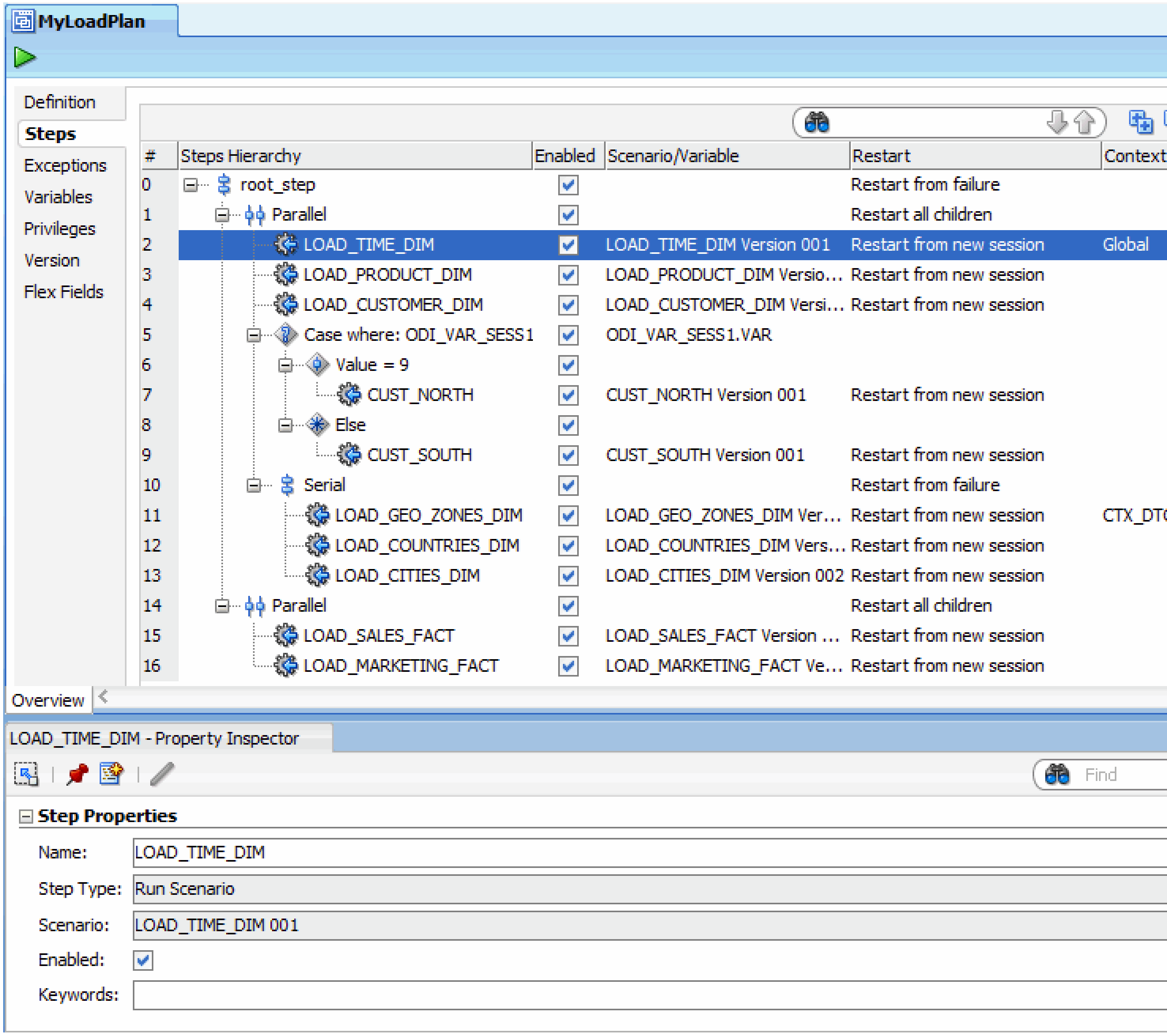
Type	Description	Possible Child Steps
Serial Step	Defines a serial execution of its child steps. Child steps are ordered and a child step is executed only when the previous one is terminated. The root step is a Serial step.	• Serial step • Parallel step • Run Scenario step • Case step
Parallel Step	Defines a parallel execution of its child steps. Child steps are started immediately in their order of Priority.	• Serial step • Parallel step • Run Scenario step • Case step
Run Scenario Step	Launches the execution of a scenario.	This type of step cannot have a child steps.
Case Step When Step Else Steps	The combination of these steps allows conditional branching based on the value of a variable. **Note**: If you have several When steps under a Case step, only the first enabled When step that satisfies the condition is executed. If no When step satisfies the condition or the Case step does not contain any When steps, the Else step is executed.	Of a Case Step: • When step • Else step Of a When step: • Serial step • Parallel step • Run Scenario step • Case step Of an Else step: • Serial step • Parallel step • Run Scenario step • Case step
Exception Step	Defines a group of steps that is executed when an exception is encountered in the associated step from the Step Hierarchy. The same exception step can be attached to several steps in the Steps Hierarchy.	• Serial step • Parallel step • Run Scenario step • Case step

The figure below shows a sample Load Plan created in Oracle Data Integrator. This sample Load Plan loads a data warehouse:

- Dimensions are loaded in parallel. This includes the LOAD_TIME_DIM, LOAD_PRODUCT_DIM, LOAD_CUSTOMER_DIM scenarios, the geographical dimension and depending on the value of the ODI_VAR_SESS1 variable, the CUST_NORTH or CUST_SOUTH scenario.

- The geographical dimension consists of a sequence of three scenarios (LOAD_GEO_ZONE_DIM, LOAD_COUNTRIES_DIM, LOAD_CITIES_DIM).
- After the dimensions are loaded, the two fact tables are loaded in parallel (LOAD_SALES_FACT and LOAD_MARKETING_FACT scenarios).

Figure 16-1 Sample Load Plan



MyLoadPlan

Definition
Steps
Exceptions
Variables
Privileges
Version
Flex Fields

#	Steps Hierarchy	Enabled	Scenario/Variable	Restart	Context
0	root_step	✓		Restart from failure	
1	Parallel	✓		Restart all children	
2	LOAD_TIME_DIM	✓	LOAD_TIME_DIM Version 001	Restart from new session	Global
3	LOAD_PRODUCT_DIM	✓	LOAD_PRODUCT_DIM Versio...	Restart from new session	
4	LOAD_CUSTOMER_DIM	✓	LOAD_CUSTOMER_DIM Versi...	Restart from new session	
5	Case where: ODI_VAR_SESS1	✓	ODI_VAR_SESS1.VAR		
6	Value = 9	✓			
7	CUST_NORTH	✓	CUST_NORTH Version 001	Restart from new session	
8	Else	✓			
9	CUST_SOUTH	✓	CUST_SOUTH Version 001	Restart from new session	
10	Serial	✓		Restart from failure	
11	LOAD_GEO_ZONES_DIM	✓	LOAD_GEO_ZONES_DIM Ver...	Restart from new session	CTX_DTC
12	LOAD_COUNTRIES_DIM	✓	LOAD_COUNTRIES_DIM Vers...	Restart from new session	
13	LOAD_CITIES_DIM	✓	LOAD_CITIES_DIM Version 002	Restart from new session	
14	Parallel	✓		Restart all children	
15	LOAD_SALES_FACT	✓	LOAD_SALES_FACT Version ...	Restart from new session	
16	LOAD_MARKETING_FACT	✓	LOAD_MARKETING_FACT Ve...	Restart from new session	

Overview

LOAD_TIME_DIM - Property Inspector

Find

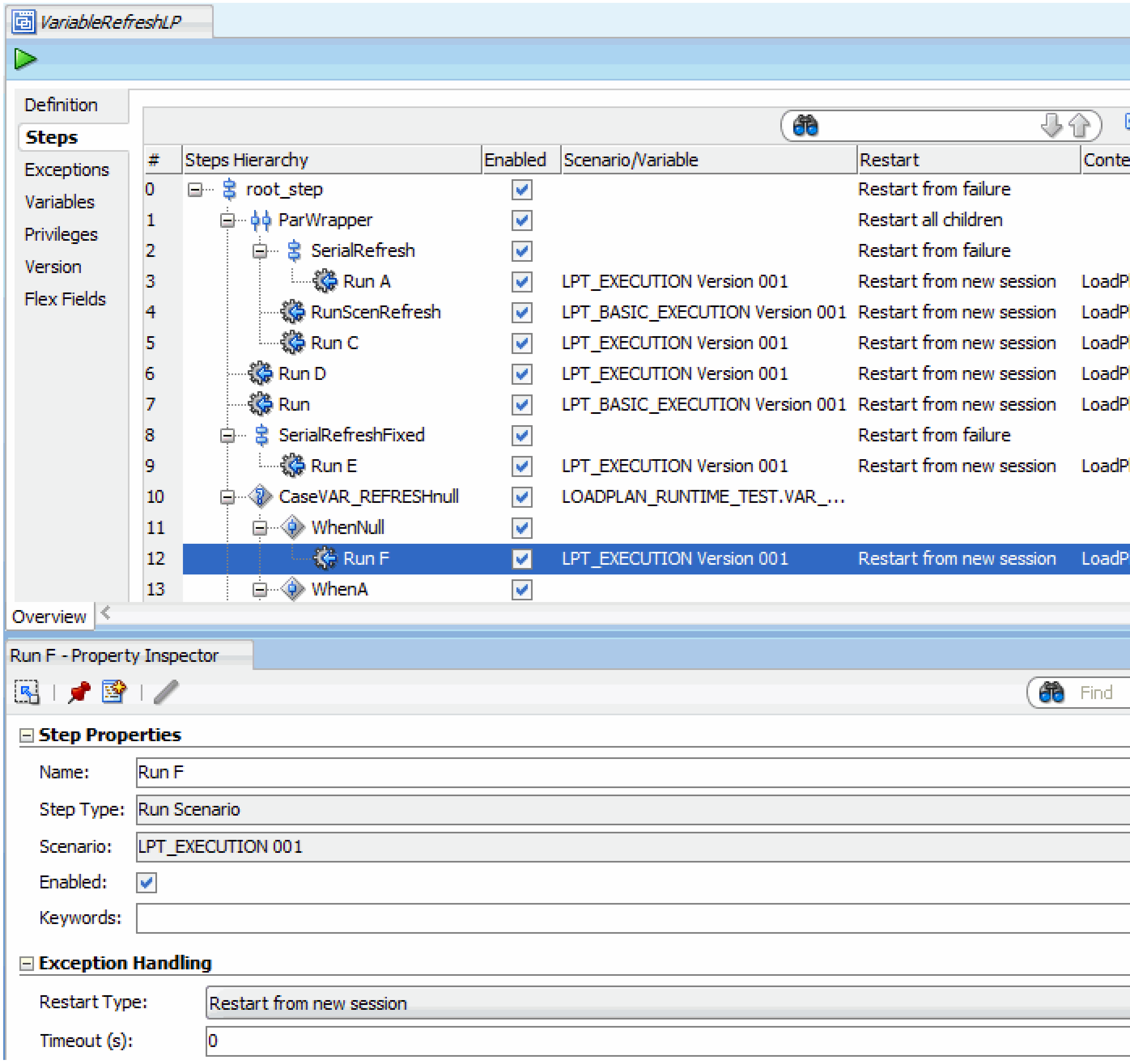
Step Properties

Name:	LOAD_TIME_DIM
Step Type:	Run Scenario
Scenario:	LOAD_TIME_DIM 001
Enabled:	✓
Keywords:	

Introduction to the Load Plan Editor

The Load Plan Editor provides a single environment for designing Load Plans. The figure below gives an overview of the Load Plan Editor.

Figure 16-2 Steps Tab of the Load Pan Editor



#	Steps Hierarchy	Enabled	Scenario/Variable	Restart	Conte
0	root_step	✓		Restart from failure	
1	ParWrapper	✓		Restart all children	
2	SerialRefresh	✓		Restart from failure	
3	Run A	✓	LPT_EXECUTION Version 001	Restart from new session	LoadP
4	RunScenRefresh	✓	LPT_BASIC_EXECUTION Version 001	Restart from new session	LoadP
5	Run C	✓	LPT_EXECUTION Version 001	Restart from new session	LoadP
6	Run D	✓	LPT_EXECUTION Version 001	Restart from new session	LoadP
7	Run	✓	LPT_BASIC_EXECUTION Version 001	Restart from new session	LoadP
8	SerialRefreshFixed	✓		Restart from failure	
9	Run E	✓	LPT_EXECUTION Version 001	Restart from new session	LoadP
10	CaseVAR_REFRESHnull	✓	LOADPLAN_RUNTIME_TEST.VAR_...		
11	WhenNull	✓			
12	Run F	✓	LPT_EXECUTION Version 001	Restart from new session	LoadP
13	WhenA	✓			

Step Properties

Name:	Run F
Step Type:	Run Scenario
Scenario:	LPT_EXECUTION 001
Enabled:	✓
Keywords:	

Exception Handling

Restart Type:	Restart from new session
Timeout (s):	0

The Load Plan steps are added, edited and organized in the Steps tab of the Load Plan Editor. The *Steps Hierarchy table* defines the organization of the steps in the Load Plan. Each row in this table represents a step and displays its main properties.

You can drag components such as packages, integration mappings, variables, procedures, or scenarios from the Designer Navigator into the Steps Hierarchy table for creating Run Scenario steps for these components.

You can also use the Add Step Wizard or the Quick Step tool to add Run Scenario steps and other types of steps into this Load Plan. See Adding Load Plan Steps for more information.

The *Load Plan Editor toolbar*, located on top of the Steps Hierarchy table, provides tools for creating, organizing, and sequencing the steps in the Load Plan. The figure below details the different toolbar components.

Table 16-2 Load Plan Editor Toolbar

Icon	Name	Description
🔍 ⇩ ⇧	Search	Searches for a step in the Steps Hierarchy table.
⊞	Expand All	Expands all tree nodes in the Steps Hierarchy table.
⊟	Collapse All	Collapses all tree nodes in the Steps Hierarchy table.
✚▾	Add Step	Opens a Add Step menu. You can either select the Add Step Wizard or a Quick Step tool to add a step. See Adding Load Plan Steps for more information.
✖	Remove Step	Removes the selected step and all its child steps.
⇧ ⇩ ⇦ ⇨	Reorder arrows: Move Up, Move Down, Move Out, Move In	Use the reorder arrows to move the selected step to the required position.

The *Properties Panel*, located under the Steps Hierarchy table, displays the properties for the object that is selected in the Steps Hierarchy table.

Creating a Load Plan

This section describes how to create a new Load Plan in ODI Studio.

1. Define a new Load Plan. See Creating a New Load Plan for more information.

2. Add Steps into the Load Plan and define the Load Plan Sequence. See Defining the Load Plan Step Sequence for more information.
3. Define how the exceptions should be handled. See Handling Load Plan Exceptions and Restartability for more information.

Creating a New Load Plan

Load Plans can be created from the Designer or Operator Navigator.

To create a new Load Plan:

1. In Designer Navigator or Operator Navigator, click **New Load Plan** in the toolbar of the Load Plans and Scenarios accordion. The Load Plan Editor is displayed.
2. In the Load Plan Editor, type in the **Name**, **Folder Name**, and a **Description** for this Load Plan.
3. Optionally, set the following parameters:
   - **Log Sessions**: Select how the session logs should be preserved for the sessions started by the Load Plan. Possible values are:
     – **Always**: Always keep session logs (Default)
     – **Never**: Never keep session logs. Note that for Run Scenario steps that are configured as *Restart from Failed Step* or *Restart from Failed Task,* the agent will behave as if the parameter is set to **Error** as the whole session needs to be preserved for restartability.
     – **Error**: Only keep the session log if the session completed in an error state.
   - Log Session Step: Select how the logs should be maintained for the session steps of each of the session started by the Load Plan. Note that this applies only when the session log is preserved. Possible values are:
     – **By Scenario Settings**: Session step logs are preserved depending on the scenario settings. Note that for scenarios created from packages, you can specify whether to preserve or not the steps in the advanced step property called *Log Steps in the Journal*. Other scenarios preserve all the steps (Default).
     – **Never**: Never keep session step logs. Note that for Run Scenario steps that are configured as *Restart from Failed Step* or *Restart from Failed Task,* the agent will behave as if the parameter is set to **Error** as the whole session needs to be preserved for restartability.
     – **Errors**: Only keep session step log if the step is in an error state.
   - **Session Tasks Log Level**: Select the log level for sessions. This value corresponds to the *Log Level* value when starting unitary scenarios. Default is 5. Note that when Run Scenario steps are configured as *Restart from Failed Step* or *Restart From Failed Task*, this parameter is ignored as the whole session needs to be preserved for restartability.
   - **Keywords**: Enter a comma separated list of keywords that will be set on the sessions started from this load plan. These keywords improve the organization of ODI logs by session folders and automatic classification. Note that you can overwrite these keywords at the level of the child steps. See the "Managing the Log" section in *Administering Oracle Data Integrator* for more information.

4. Optionally, modify the **Concurrent Execution Controller** options:
   - Enable the **Limit Concurrent Executions** check box if you do not want to allow multiple instances of this load plan to be run at the same time. If **Limit Concurrent Executions** is disabled (unchecked), no restriction is imposed and more than one instance of this load plan can be running simultaneously.
   - If **Limit Concurrent Executions** is enabled, set your desired **Violation Behavior**:
     - **Raise Execution Error**: if an instance of the load plan is already running, attempting to run another instance will result in a session being created but immediately ending with an execution error message identifying the session that is currently running which caused the Concurrent Execution Control error.
     - **Wait to Execute**: if an instance of the load plan is already running, additional executions will be placed in a wait status and the system will poll for its turn to run. The session's status is updated periodically to show the currently running session, as well as all concurrent sessions (if any) that are waiting in line to run after the running instance is complete.

       If you select this option, the **Wait Polling Interval** sets how often the system will check to see if the running instance has completed. You can only enter a **Wait Polling Interval** if **Wait to Execute** is selected.

       If you do not specify a wait polling interval, the default for the executing agent will be used: in ODI 12.1.3, the default agent value is 30 seconds.
5. Select the **Steps** tab and add steps as described in Defining the Load Plan Step Sequence.
6. If your Load Plan requires conditional branching, or if your scenarios use variables, select the **Variables** tab and declare variables as described in Declaring Load Plan Variables.
7. To add exception steps that are used in the event of a load plan step failing, select the **Exceptions** tab and define exception steps as described in Defining Exceptions Flows.
8. From the **File** menu, click **Save**.

The Load Plan appears in the Load Plans and Scenarios accordion. You can organize your Load Plans by grouping related Load Plans and Scenarios into a Load Plan and Scenarios folder.

Defining the Load Plan Step Sequence

Load Plans are an organized hierarchy of child steps. This hierarchy allows conditional processing of steps in parallel or in series.

The execution flow can be configured at two stages:

- At Design-time, when defining the Steps Hierarchy:
  - When you add a step to a Load Plan, you select the step type. The step type defines the possible child steps and how these child steps are executed: in parallel, in series, or conditionally based on the value of a variable (Case step). See the table on Load Plan Steps in Load Plan Structure for more information on step types.

  - When you add a step to a Load Plan, you also decide where to insert the step. You can add a child step, a sibling step after the selected step, or a sibling step before the selected step. See Adding Load Plan Steps for more information.
  - You can also reorganize the order of the Load Plan steps by dragging the step to the wanted position or by using the arrows in the Step table toolbar. See the table on Load Plan Edit Toolbar in Introduction to the Load Plan Editor for more information.
- At design-time and run-time by enabling or disabling a step. In the Steps hierarchy table, you can enable or disable a step. Note that disabling a step also disables all its child steps. Disabled steps and all their child steps are not executed when you run the load plan.

This section contains the following topics:

- Adding Load Plan Steps
- Editing Load Plan Steps
- Deleting a Step
- Duplicating a Step

Adding Load Plan Steps

A Load Plan step can be added by using the Add Step Wizard or by selecting the Quick Step tool for a specific step type. A load plan step can be also createdby dragging an object (such as a scenario, package, etc.) and dropping it onto a container step. The step will be created as a child of the selected step. See the table on Load Plan Steps in Load Plan Structure for more information on the different types of Load Plan steps. To create Run Scenario steps, you can also drag components such as packages, mappings, variables, procedures, or scenarios from the Designer Navigator into the Steps Hierarchy table. Oracle Data Integrator automatically creates a Run Scenario step for the inserted component.

When a Load Plan step is added, it is inserted into the Steps Hierarchy with the minimum required settings. See Editing Load Plan Steps for more information on how to configure Load Plan steps.

Adding a Load Plan Step with the Add Step Wizard

To insert Load Plan step with the Add Step Wizard:

1. Open the Load Plan Editor and go to the **Steps** tab.
2. Select a step in the Steps Hierarchy table.
3. In the Load Plan Editor toolbar, select **Add Step** > **Add Step Wizard**.
4. In the Add Step Wizard, select:
   - **Step Type**. Possible step types are: Serial, Parallel, Run Scenario, Case, When, and Else. See the table on Load Plan Steps in Load Plan Structure for more information on the different step types.
   - **Step Location**. This parameter defines where the step is added.
     - **Add a child step to selection**: The step is added under the selected step.

- **Add a sibling step after selection**: The step is added on the same level after the selected step.
- **Add a sibling step before selection**: The step is added on the same level before the selected step.

Note:

Only values that are valid for the current selection are displayed for the **Step Type** and **Step Location**.

5. Click **Next**.
6. Follow the instructions in the table below for the step type you are adding.

Table 16-3 Add Step Wizard Actions

Step Type	Description and Action Required
Serial or Parallel step	Enter a **Step Name** for the new Load Plan step.

Table 16-3 (Cont.) Add Step Wizard Actions

Step Type	Description and Action Required
Run Scenario step	**a.** Click the **Lookup Scenario** button. **b.** In the Lookup Scenario dialog, you can select the scenario you want to add to your Load Plan and click **OK**. Alternately, to create a scenario for an executable object and use this scenario, select this object type in the Executable Object Type selection box, then select the executable object that you want to run with this Run Scenario step and click **OK**. Enter the new scenario name and version and click **OK**. A new scenario is created for this object and used in this Run Scenario Step. **Tip:** At design time, you may want to create a Run Scenario step using a scenario that does not exist yet. In this case, instead of selecting an existing scenario, enter directly a Scenario Name and a Version number and click **Finish**. Later on, you can select the scenario using the Modify Run Scenario Step wizard. See the section **Change the Scenario of a Run Scenario Step** in Editing Load Plan Steps for more information. Note that when you use the version number `-1`, the latest version of the scenario will be used, based on the string's lexical sorting order. **c.** The **Step Name** is automatically populated with the name of the scenario and the **Version** field with the version number of the scenario. Optionally, change the Step Name. **d.** Click **Next**. **e.** In the Add to Load Plan column, select the scenario variables that you want to add to the Load Plan variables. If the scenario uses certain variables as its startup parameters, they are automatically added to the Load Plan variables. See Working with Variables in Load Plans for more information.
Case	**a.** Select the variable you want to use for the conditional branching. Note that you can either select one of the load plan variables from the list or click **Lookup Variable** to add a new variable to the load plan and use it for this case step. See Working with Variables in Load Plans for more information. **b.** The **Step Name** is automatically populated with the step type and name of the variable. Optionally, change the Step Name. See Editing Load Plan Steps for more information.

Table 16-3 (Cont.) Add Step Wizard Actions

Step Type	Description and Action Required
When	a. Select the **Operator** to use in the WHEN clause evaluation. Possible values are: • Less Than (<) • Less Than or Equal (<=) • Different (<>) • Equals (=) • Greater Than (>) • Greater Than or Equal (>=) • Is not Null • Is Null b. Enter the **Value** to use in the WHEN clause evaluation. c. The **Step Name** is automatically populated with the operator that is used. Optionally, change the Step Name. See Editing Load Plan Steps for more information.
Else	The **Step Name** is automatically populated with the step type. Optionally, change the Step Name. See Editing Load Plan Steps for more information.

7. Click **Finish**.
8. The step is added in the steps hierarchy.

Note:

You can reorganize the order of the Load Plan steps by dragging the step to the desired position or by using the reorder arrows in the Step table toolbar to move a step in the Steps Hierarchy.

Adding a Load Plan Step with the Quick Step Tool

To insert Load Plan step with the Quick Step Tool:

1. Open the Load Plan editor and go to the **Steps** tab.
2. In the Steps Hierarchy, select the Load Plan step under which you want to create a child step.
3. In the Steps toolbar, select **Add Step** and the Quick Step option corresponding to the Step type you want to add. The table below lists the options of the Quick Step tool.

Table 16-4 Quick Step Tool

Quick Step tool option	Description and Action Required
⇣	Adds a serial step as a child of the selected step. Default values are used. You can modify these values in the Steps Hierarchy table or in the Property Inspector. See Editing Load Plan Steps for more information.
⇊	Adds a parallel step as a child of the selected step. Default values are used. You can modify these values in the Steps Hierarchy table or in the Property Inspector. See Editing Load Plan Steps for more information.
⚙	Adds a run scenario step as a child of the selected step. Follow the instructions for Run Scenario steps in the **Add Step Wizard Actions** table.
◆	Adds a Case step as a child of the selected step. Follow the instructions for Case steps in the **Add Step Wizard Actions** table.
◆	Adds a When step as a child of the selected step. Follow the instructions for When steps in the **Add Step Wizard Actions** table.
◆	Adds an Else step as a child of the selected step. Follow the instructions for Else steps in the **Add Step Wizard Actions** table.

Note:

Only step types that are valid for the current selection are enabled in the Quick Step tool.

Editing Load Plan Steps

To edit a Load Plan step:

1. Open the Load Plan editor and go to the **Steps** tab.
2. In the Steps Hierarchy table, select the Load Plan step you want modify. The Property Inspector displays the step properties.

3. Edit the Load Plan step properties according to your needs.

The following operations are common tasks when editing steps:

Change the Scenario of a Run Scenario Step

To change the scenario:

1. In the Steps Hierarchy table of the Steps or Exceptions tab, select the Run Scenario step.
2. In the Step Properties section of the Properties Inspector, click **Lookup Scenario**. This opens the Modify Run Scenario Step wizard.
3. In the Modify Run Scenario Step wizard, click **Lookup Scenario** and follow the instructions in the Add Step Wizard Actions table in Adding Load Plan Steps corresponding to the Run Scenario step.

Set Advanced Options for Run Scenario Steps

You can set the following properties for Run Scenario steps in the Property Inspector:

- **Priority**: Priority for this step when the scenario needs to start in parallel. The integer value range is from 0 to 100 (100 being the highest priority). Default is 0. The priority of a Run Scenario step is evaluated among all runnable scenarios within a running Load Plan. The Run Scenario step with the highest priority is executed first.
- **Context**: Context that is used for the step execution. Default context is the Load Plan context that is defined in the Start Load Plan Dialog when executing a Load Plan. Note that if you only specify the Context and no Logical Agent value, the step is started on the same physical agent that started the Load Plan, but in this specified context.
- **Logical Agent**: Logical agent that is used for the step execution. By default, the logical agent, which is defined in the Start Load Plan Dialog when executing a Load Plan, is used. Note that if you set only the Logical Agent and no context, the step is started with the physical agent corresponding to the specified Logical Agent resolved in the context specified when starting the Load Plan. If no Logical Agent value is specified, the step is started on the same physical agent that started the Load Plan (whether a context is specified for the step or not).

Open the Linked Object of Run Scenario Steps

Run Scenario steps can be created for packages, mappings, variables, procedures, or scenarios. Once this Run Scenario step is created, you can open the Object Editor of the original object to view and edit it.

To view and edit the linked object of Run Scenario steps:

1. In the Steps Hierarchy table of the Steps or Exceptions tab, select the Run Scenario step.
2. Right-click and select **Open the Linked Object**.

The Object Editor of the linked object is displayed.

Change the Test Variable in Case Steps

To change the variable that is used for evaluating the tests defined in the WHEN statements:

1. In the Steps Hierarchy table of the Steps tab or Exceptions tab, select the Case step.
2. In the Step Properties section of the Properties Inspector, click **Lookup Variable**. This opens the Modify Case Step Dialog.
3. In the Modify Case Step Dialog, click **Lookup Variable** and follow the instructions in the Add Step Wizard Actions table in Adding Load Plan Steps corresponding to the Case step.

Define the Exception and Restart Behavior

Exception and Restart behavior can be set on the steps in the Steps Hierarchy table. See Handling Load Plan Exceptions and Restartability for more information.

Regenerate Scenarios

To regenerate all the scenarios of a given Load Plan step, including the scenarios of its child steps:

1. From the Steps Hierarchy table of the Steps tab or Exceptions tab, select the Load Plan step.
2. Right-click and select **Regenerate**. Note that this option is not available for scenarios with the version number `-1`.
3. Click **OK**.

> **Caution:**
>
> Regenerating a scenario cannot be undone. For important scenarios, it is better to generate a scenario with a new version number.

Refresh Scenarios to Latest Version

To modify all the scenario steps of a given Load Plan step, including the scenarios of its child steps, and set the scenario version to the latest version available for each scenario:

1. From the Steps Hierarchy table of the Steps tab or Exceptions tab, select the Load Plan step.
2. Right-click and select **Refresh Scenarios to Latest Version**. Note that the latest scenario version is determined by the Scenario Creation timestamp. While during the ODI agent execution, the latest scenario is determined by alphabetical ascending sorting of the Scenario Version string value and picking up the last from the list.

   > **Note:**
   >
   > This option is not available for scenarios with the version number `-1`.

3. Click **OK**.

Deleting a Step

To delete a step:

1. Open the Load Plan Editor and go to the **Steps** tab.
2. In the Steps Hierarchy table, select the step to delete.
3. In the Load Plan Editor toolbar, select **Remove Step**.

The step and its child steps are removed from the Steps Hierarchy table.

Note:

It is not possible to undo a delete operation in the Steps Hierarchy table.

Duplicating a Step

To duplicate a step:

1. Open the Load Plan Editor and go to the **Steps** tab.
2. In the Steps Hierarchy table, right-click the step to duplicate and select **Duplicate Selection**.
3. A copy of this step, including its child steps, is created and added as a sibling step after the original step to the Step Hierarchy table.

You can now move and edit this step.

Working with Variables in Load Plans

Project and Global Variables used in a Load Plan are declared as Load Plan Variables in the Load Plan editor. These variables are automatically available in all steps and their value passed to the Load Plan steps.

The variables values are passed to the Load Plan on startup as startup parameters. At a step level, you can overwrite the variable value (by setting it or forcing a refresh) for this step and its child steps.

Note:

Load plan variables are copies of Project and Global variables. Thus, changes to the definition of the original project and global variables are not automatically propagated to corresponding variables that are already created in a load plan. You can use the **Refresh Variable Definition** option on the right-click context menu to update the definition of a load plan variable with the current value from the corresponding Project or Global variable.

Because a load plan variable is a copy of the original project or global variable, at startup, Load Plans do not take into account the default value of the original project or global variable, or the historized/latest value of the variable in the execution context. The value of the variable is either the one specified when starting the Load Plan, or the value set/refreshed within the Load Plan.

You can use variables in Run Scenario steps - the variable values are passed as startup parameters to the scenario - or in Case/When/Else steps for conditional branching.

This section contains the following topics:

- Declaring Load Plan Variables
- Setting Variable Values in a Step

Declaring Load Plan Variables

To declare a Load Plan variable:

1. Open the Load Plan editor and go to the **Variables** tab.
2. From the Load Plan Editor toolbar, select **Add Variable**. The Lookup Variable dialog is displayed.
3. In the Lookup Variable dialog, select the variable to add your Load Plan.
4. The variable appears in the Variables tab of the Load Plan Editor and in the Property Inspector of each step.

Setting Variable Values in a Step

Variables in a step inherit their value from the value from the parent step and ultimately from the value specified for the variables when starting the Load Plan.

For each step, except for Else and When steps, you can also overwrite the variable value, and change the value used for this step and its child steps.

Variable values overwritten or refreshed at a given step are available to all the step's descendants, until the value is overwritten or refreshed again for a descendant branch. Similarly, a variable value overwritten or refreshed at a given step does not affect the value for sibling or parent steps.

To override variable values at step level:

1. Open the Load Plan editor and go to the **Steps** tab.

2. In the Steps Hierarchy table, select the step for which you want to overwrite the variable value.
3. In the Property Inspector, go to the Variables section. The variables that are defined for this Load Plan are listed in this Variables table. You can modify the following variable parameters:

   Select **Overwrite**, if you want to specify a variable value for this step and all its children. Once you have chosen to overwrite the variable value, you can either:

   - Set a new variable value in the **Value** field.
   - Select **Refresh** to refresh this variable prior to executing the step. The Refresh option can be selected only for variables with a Select Query defined for refreshing the variable value.

     Note:

     If the refresh SQL of a Global or Project variable has changed, the variable refresh SQL of the corresponding load plan variable is not updated automatically. You can update the load plan variable refresh SQL by selecting the **Refresh Variable Definition** option from the right-click context menu for a load plan variable on the Variables tab of the load plan editor.

Handling Load Plan Exceptions and Restartability

Load Plans provide two features for handling error cases in the execution flows: *Exceptions* and *Restartability*.

Exceptions

An *Exception Step* contains a hierarchy of steps that is defined on the Exceptions tab of the Load Plan editor.

You can associate a given exception step to one or more steps in the Load Plan. When a step in the Load Plan errors out, the associated exception step is executed automatically.

Exceptions can be optionally raised to the parent step of the failing step. Raising an exception fails the parent step, which can consequently execute its exception step.

Restartability

When a Load Plan Run is restarted after a failure, the failed Load Plan steps are restarted depending on the Restart Type parameter. For example, you can define whether a parallel step should restart all its child steps or only those that have failed.

This section contains the following topics:

- Defining Exceptions Flows
- Using Exception Handling
- Defining the Restart Behavior

Defining Exceptions Flows

Exception steps are created and defined on the Exceptions tab of the Load Plan Editor.

This tab contains a list of *Exception Steps*. Each Exception Step consists in a hierarchy of Load Plan steps.The Exceptions tab is similar to the Steps tab in the Load Plan editor. The main differences are:

- There is no root step for the Exception Step hierarchy. Each exception step is a separate root step.
- The Serial, Parallel, Run Scenario, and Case steps have the same properties as on the Steps tab but do not have an Exception Handling properties group. An exception step that errors out cannot raise another exception step.

An Exception step can be created either by using the Add Step Wizard or with the Quick Step tool by selecting the **Add Step** > **Exception Step** in the Load Plan Editor toolbar. By default, the Exception step is created with the Step name: *Exception*. You can modify this name in the Steps Hierarchy table or in the Property Inspector.

To create an Exception step with the Add Step Wizard:

1. Open the Load Plan Editor and go to the **Exceptions** tab.
2. In the Load Plan Editor toolbar, select **Add Step** > **Add Step Wizard**.
3. In the Add Step Wizard, select **Exception** from the **Step Type** list.

   Note:

   Only values that are valid for the current selection are displayed for the **Step Type**.

4. Click **Next**.
5. In the **Step Name** field, enter a name for the Exception step.
6. Click **Finish**.
7. The Exception step is added in the steps hierarchy.

You can now define the exception flow by adding new steps and organizing the hierarchy under this exception step.

Using Exception Handling

Defining exception handling for a Load Plan step consists of associating an Exception Step to this Load Plan step and defining the exception behavior. Exceptions steps can be set for each step except for When and Else steps.

To define exception handling for a Load Plan step:

1. Open the Load Plan Editor and go to the **Steps** tab.
2. In the Steps Hierarchy table, select the step for which you want to define an exception behavior. The Property Inspector displays the Step properties.

3. In the **Exception Handling** section of the Property Inspector, set the parameters as follows:
   - **Timeout (s)**: Enter the maximum time (in seconds) that this step takes before it is aborted by the Load Plan. When a time-out is reached, the step is marked in error and the Exception step (if defined) is executed. In this case, the exception step never times out. If needed, a timeout can be set on a parent step to safe guard such a potential long running situation.

     If the step fails before the timeout and an exception step is executed, then the execution time of the step plus the execution time of the exception step should not exceed the timeout, otherwise the exception step will fail when the timeout is reached.

     Note that the default value of zero (`0`) indicates an infinite timeout.
   - **Exception Step**: From the list, select the Exception step to execute if this step fails. Note that only Exception steps that have been created and defined on the Exceptions tab of the Load Plan Editor appear in this list. See Defining Exceptions Flows for more information on how to create an Exception step.
   - **Exception Behavior**: Defines how this step behaves in case an exception is encountered. Select one of the following:
     - **Run Exception and Raise**: Runs the Exception Step (if any) and raises the exception to the parent step.
     - **Run Exception and Ignore**: Runs the Exception Step (if any) and ignores the exception. The parent step is notified of a successful run. Note that if an exception is caused by the exception step itself, the parent step is notified of the failure.

   For Parallel steps only, the following parameters may be set:

   Max Error Child Count: Displays the maximum number of child steps in error that is accepted before this step is to be considered in error. When the number of failed child steps exceeds this value, the parallel step is considered failed. The currently running child steps are continued or stopped depending on the Restart Type parameter for this parallel step:
   - If the Restart type is **Restart from failed children**, the Load Plan waits for all child sessions (these are the currently running sessions and the ones waiting to be executed) to run and complete before it raises the error to the parent step.
   - If the Restart Type is **Restart all children**, the Load Plan kills all running child sessions and does not start any new ones before it raises the error to the parent.

Defining the Restart Behavior

The Restart Type option defines how a step in error restarts when the Load Plan is restarted. You can define the **Restart Type** parameter in the Exception Handling section of the Properties Inspector.

Depending on the step type, the **Restart Type** parameter can take the values listed in the table below.

Table 16-5 Restart Type Values

Step Type	Values and Description
Serial	• **Restart all children**: When the Load Plan is restarted and if this step is in error, the sequence of steps restarts from the first one. • **Restart from failure**: When the Load Plan is restarted and if this step is in error, the sequence of child steps starts from the one that has failed.
Parallel	• **Restart all children**: When the Load Plan is restarted and if this step is in error, all the child steps are restarted regardless of their status. This is the default value. • **Restart from failed children**: When the Load Plan is restarted and if this step is in error, only the failed child steps are restarted in parallel.
Run Scenario	• **Restart from new session**: When restarting the Load Plan and this Run Scenario step is in error, start the scenario and create a new session. This is the default value. • **Restart from failed step**: When restarting the Load Plan and this Run Scenario step is in error, restart the session from the step in error. All the tasks under this step are restarted. • **Restart from failed task**: When restarting the Load Plan and this Run Scenario step is in error, restart the session from the task in error. The same limitation as those described in the "Restarting a Session" section in *Administering Oracle Data Integrator* apply to the sessions restarted from a failed step or failed task.

Running Load Plans

You can run a Load Plan from Designer Navigator or Operator Navigator in ODI Studio.

Caution:

Unless concurrent execution has been limited by using the **Concurrent Execution Controller** options on the **Definition** tab of a load plan, no restriction is imposed to prevent multiple instances of a load plan from running simultaneously. It is possible for two or more instances of a load plan to perform data read/write operations on the same data sources and targets simultaneously. Use the **Limit Concurrent Executions** option to disallow this behavior programmatically if concurrent execution would be undesirable.

See Creating a New Load Plan for details.

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

1. In the Load Plans and Scenarios accordion, select the Load Plan you want to execute.
2. Right-click and select **Execute**.

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 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** dialog is displayed.
6. Click **OK**.

The Load Plan execution starts: a Load Plan instance is created along with the first Load Plan run. You can review the Load Plan execution in the Operator Navigator.

> **Note:**
>
> OracleDIAgent executes purge jobs based on the value of 'Keep Log History (days)' parameter defined at the individual Load Plans level. The default load plan purge value for a work repository is 7 days and you can set the parameter to a higher value, if you need to keep the history for a longer time. Such purge jobs do not appear in Operator panel as they are internal jobs that ODI executes automatically.

For more information, see the Monitoring Integration Processes chapter in *Administering Oracle Data Integrator*, and see also the Running Integration Processes chapter in *Administering Oracle Data Integrator* for more information on the other run-time operations on Load Plans.

Using Load Plans in Production

Using Load Plans in production involves the following tasks:

- Scheduling, starting, monitoring, stopping and restarting Load Plans. See Scheduling and Running Load Plans in Production for information.
- Moving Load Plans across environments. See Exporting, Importing, and Versioning Load Plans for information.

Scheduling and Running Load Plans in Production

The Running Integration Processes chapter in *Administering Oracle Data Integrator* describes how to schedule and run load plans, including executing, restarting, and stopping load plan runs.

Exporting, Importing, and Versioning Load Plans

A Load Plan can be exported and then imported into a development or execution repository. This operation is used to deploy Load Plans in a different repository, possibly in a different environment or site.

The export (and import) procedure allows you to transfer Oracle Data Integrator objects from one repository to another.

17
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 17-1.

Table 17-1 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 17-1 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/oraclediagent
```

On Windows

```
startscen.cmd "-INSTANCE=OracleDIAgent1" PKG001 001 GLOBAL "-
SESSION_NAME=RUN1" "-AGENT_URL=http://localhost:20910/oraclediagent"
```

Table 17-2 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 17-2 Startscen 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 scenario (mandatory).
`<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`
`[-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/oraclediagent`, `https://mySSLHost:8002/oraclediagent`
`[-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.

Table 17-2 (Cont.) Startscen command Parameters

Parameters	Description
`[-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 17-3 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 17-3 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 17-4 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 17-4 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 17-5 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 17-5 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 17-5 (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 17-6 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 17-6 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 17-7 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 17-7 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).

Table 17-7 (Cont.) Stoploadplan Command Parameters

Parameters	Description
[-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.

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.
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 17-1 Using the View Schedule Dialog

The schedule is displayed in form of a Gantt diagram. Table 17-8 lists the details of the **Schedule** dialog.

Table 17-8 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).
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 17-1 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
- Executing a Scenario Using a Web Service
- Monitoring a Session Status Using a Web Service
- Restarting a Session Using a Web Service
- Executing a Load Plan Using a Web Service
- Stopping a Load Plan Run Using a Web Service

- Restarting a Load Plan Instance Using a Web Service
- Monitoring a Load Plan Run Status Using a Web Service
- Accessing the Web Service from a Command Line
- Using the Run-Time Web Services with External Authentication
- Using WS-Addressing
- Using Asynchronous Web Services with Callback

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>
```

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 17-9 lists the different parameters of this command, both mandatory and optional.

Table 17-9 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.
`<remote_agent_url>`	URL of the run-time agent that will run this session.
`<odi_user>`	Name of the user used to run this sessions.
`<odi_password>`	This user's password.
`-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).
`-n <session_name>`	Name of the session
`-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.

Table 17-9 (Cont.) Startscenremote command Parameters

Parameters	Description
`-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 can be repeated 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.
`-i <interval>`	Polling interval for session status if sync_mode = 2.
`-h <http_timeout>`	HTTP timeout for the web services calls.
`-v`	Verbose mode.

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 <wait_mode> -t <timeout> -i
   <interval> -h <http_timeout> -v
   ```

Table 17-10 lists the different parameters of this command, both mandatory and optional.

Table 17-10 GetSessionStatusRemote command Parameters

Parameters	Description
`<session_number>`	Number of the session to monitor.
`<work_repository>`	Name of the work repository containing the scenario.
`<remote_agent_url>`	URL of the run-time agent that will run this session.
`<odi_user>`	Name of the user used to run this sessions.
`<odi_password>`	This user's password.
`-w <wait_mode>`	Wait mode: • 0: Do not wait for session completion, report current status. • 1: Wait for session completion then report status.
`-t <timeout>`	Timeout in seconds for waiting for session to complete if sync_mode = 2.
`-i <interval>`	Polling interval for session status if sync_mode = 2.

Table 17-10 (Cont.) GetSessionStatusRemote command Parameters

Parameters	Description
`-h <http_timeout>`	HTTP timeout for the web services calls.
`-v`	Verbose mode.

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 the Configuring External Authentication section in *Administering Oracle Data Integrator* for 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.

18

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.

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.
🐞	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.

Icon	Command	Description
	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.
	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**.

Debug Session Options

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

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

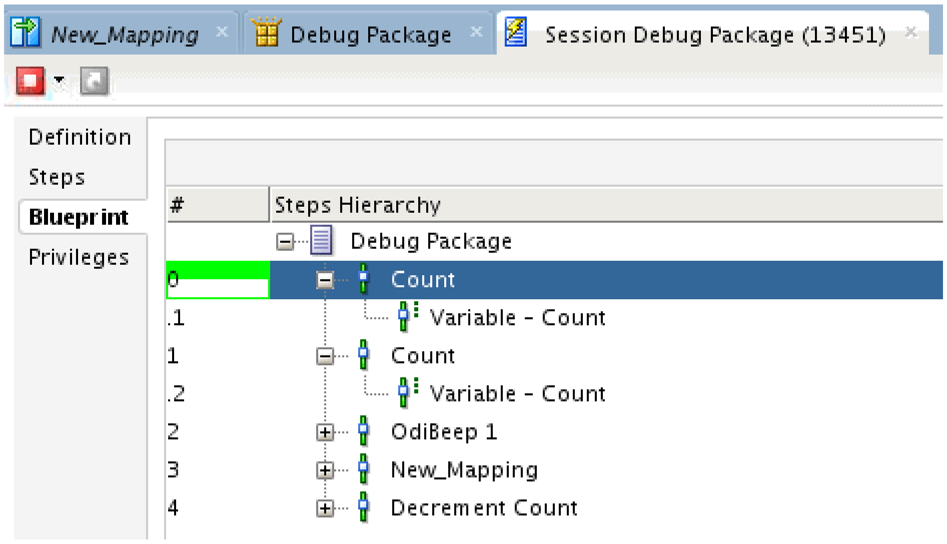
Each row in a blueprint represents a step or task. Tasks in a blueprint can be container tasks, and can execute serially or in parallel. Tasks inside a container task are leaf nodes that get executed. Sessions based on mappings have only one step, while sessions based on packages can have multiple steps.

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 18-1 shows a cursor position before the execution of a line:

Figure 18-1 Cursor Positioned Before a Step



#	Steps Hierarchy
	Debug Package
0	Count
.1	Variable - Count
1	Count
.2	Variable - Count
2	OdiBeep 1
3	New_Mapping
4	Decrement Count

Debug Actions

You can perform debug actions for a selected cursor position.

Multiple Cursors

You can use multiple cursors for in-session parallelism.

Using Breakpoints

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

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.

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.

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.

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.

19

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 19-1.

Table 19-1 Operator Navigator Toolbar Menu Items

Icon	Menu Item	Description
	Refresh	Click **Refresh** to refresh the trees in the Operator Navigator.
	Filter	Click **Filter** to define the filters for the sessions to display in Operator Navigator.
	Filter activated	
	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 Commands for 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 Folders for 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 19-2 lists the six possible status values:

Table 19-2 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 19-2 (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 19-3.

Table 19-3 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 19-3 (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.

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 19-4 for the sessions or Load Plan runs you want to delete.

Table 19-4 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.

Table 19-4 (Cont.) Purge Log Parameters

Parameter	Description
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 19-5 lists examples of how session folder keywords are matched.

Table 19-5 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 19-6.

Table 19-6 Log Export Parameters

Properties	Description
Export to directory	Directory in which the export file will be created.
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.

> **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 19-7 shows the mapping between the Java log levels and the Oracle Java Debugging Levels.

Table 19-7 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.

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.

20 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
- Using Oracle Data Integrator Console
- ODI Domain
- Oracle Enterprise Manager Fusion Middleware Control
- Management Pack for Oracle Data Integrator

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 Interface

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.

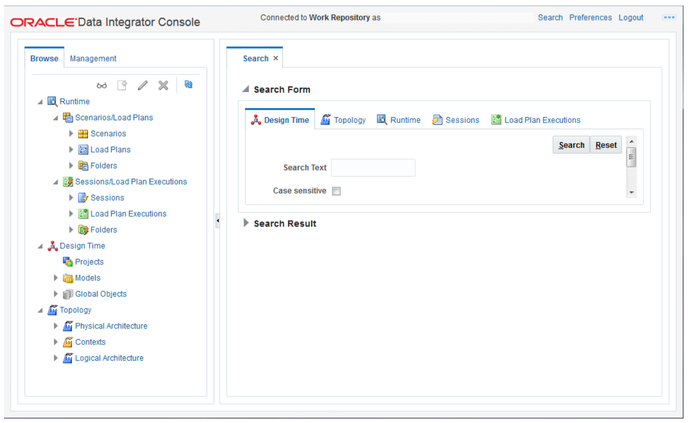
Note:

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

Figure 20-1 shows the layout of Oracle Data Integrator Console.

Figure 20-1 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 20-1.

Table 20-1 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
- Working with the Flow Map

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 the Administering Repositories chapter in *Administering Oracle Data Integrator* 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 the Topology for 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 is 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 Server for 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 the Topology for 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:

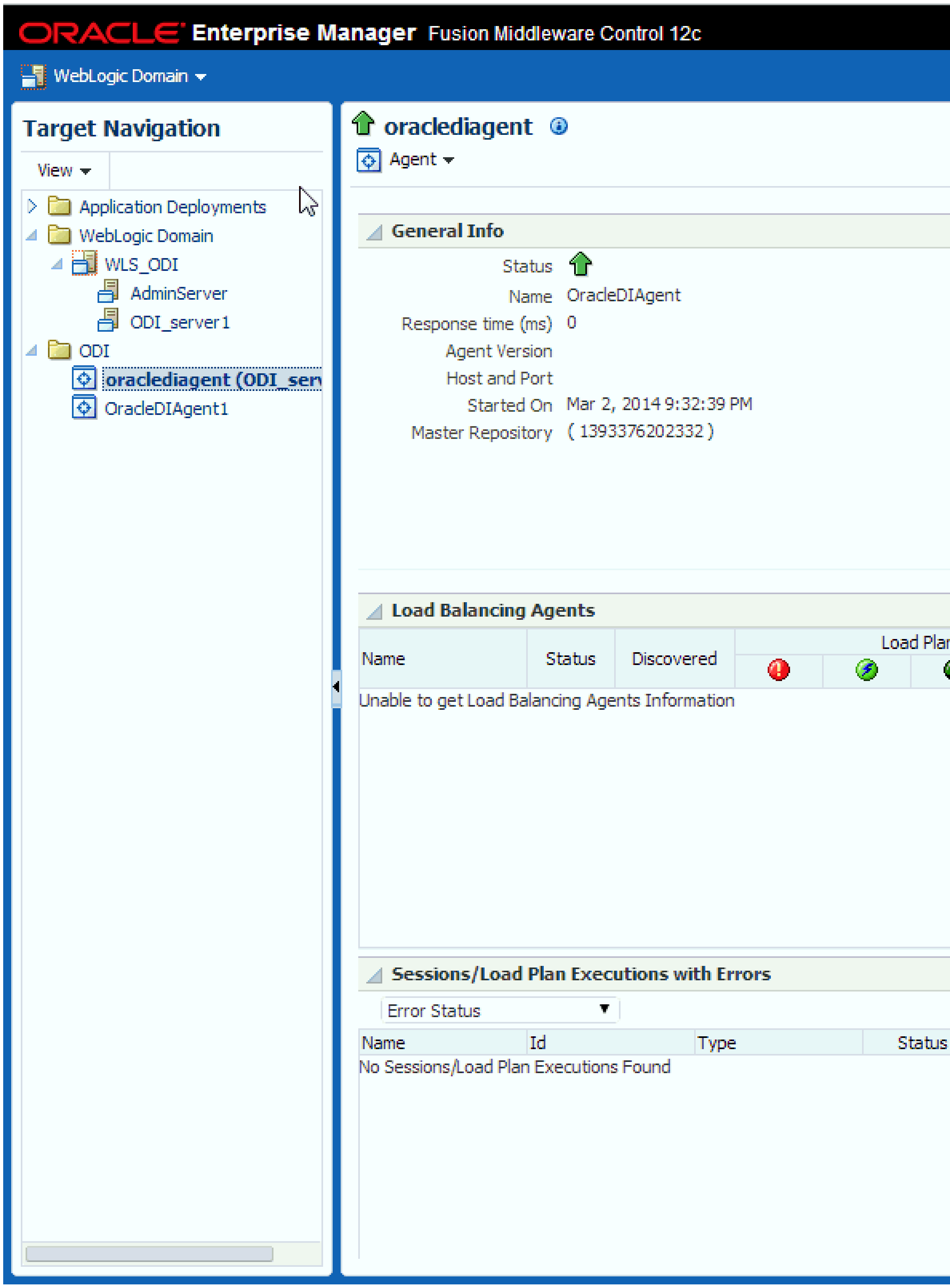
- Configuring Oracle Fusion Middleware Control with ODI Plug-in
- Searching Sessions
- Searching Load Plan Executions

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 20-2 shows the agents in the Deployment pane.

Figure 20-2 ODI Console Within Oracle Enterprise Manager Fusion Middleware Control



Status	
Name	OracleDIAgent
Response time (ms)	0
Agent Version	
Host and Port	
Started On	Mar 2, 2014 9:32:39 PM
Master Repository	(1393376202332)

Name	Status	Discovered	Load Plar		
Unable to get Load Balancing Agents Information					

Name	Id	Type	Status
No Sessions/Load Plan Executions Found			

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*.

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 Domain for 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*.

21

Managing Environments

This chapter provides information about managing the integration project environments.
This chapter includes the following topics:

- Managing Environments

Managing Environments

Integration projects exist in different environments during their lifecycle (development, test, production) and may even run in different environments in production (multiple site deployment). Oracle Data Integrator makes easier the definition and maintenance of these environments, as well as the lifecycle of the project across these environments using the Topology.

The Topology describes the physical and logical architecture of your Information System. It gives you a very flexible way of managing different servers, environments and agents. All the information of the Topology is stored in the master repository and is therefore centralized for an optimized administration. All the objects manipulated within Work Repositories refer to the Topology. That's why it is the most important starting point when defining and planning your architecture.

The Topology is composed of data servers, physical and logical schemas, and contexts.

Data servers describe connections to your actual physical application servers and databases. They can represent for example:

- Apache Hive
- An Oracle Instance
- An IBM DB2 Database
- A Microsoft SQL Server Instance
- A Sybase ASE Server
- A File System
- An XML File
- A Microsoft Excel Workbook
- and so forth.

At runtime, Oracle Data Integrator uses the connection information you have described to connect to the servers.

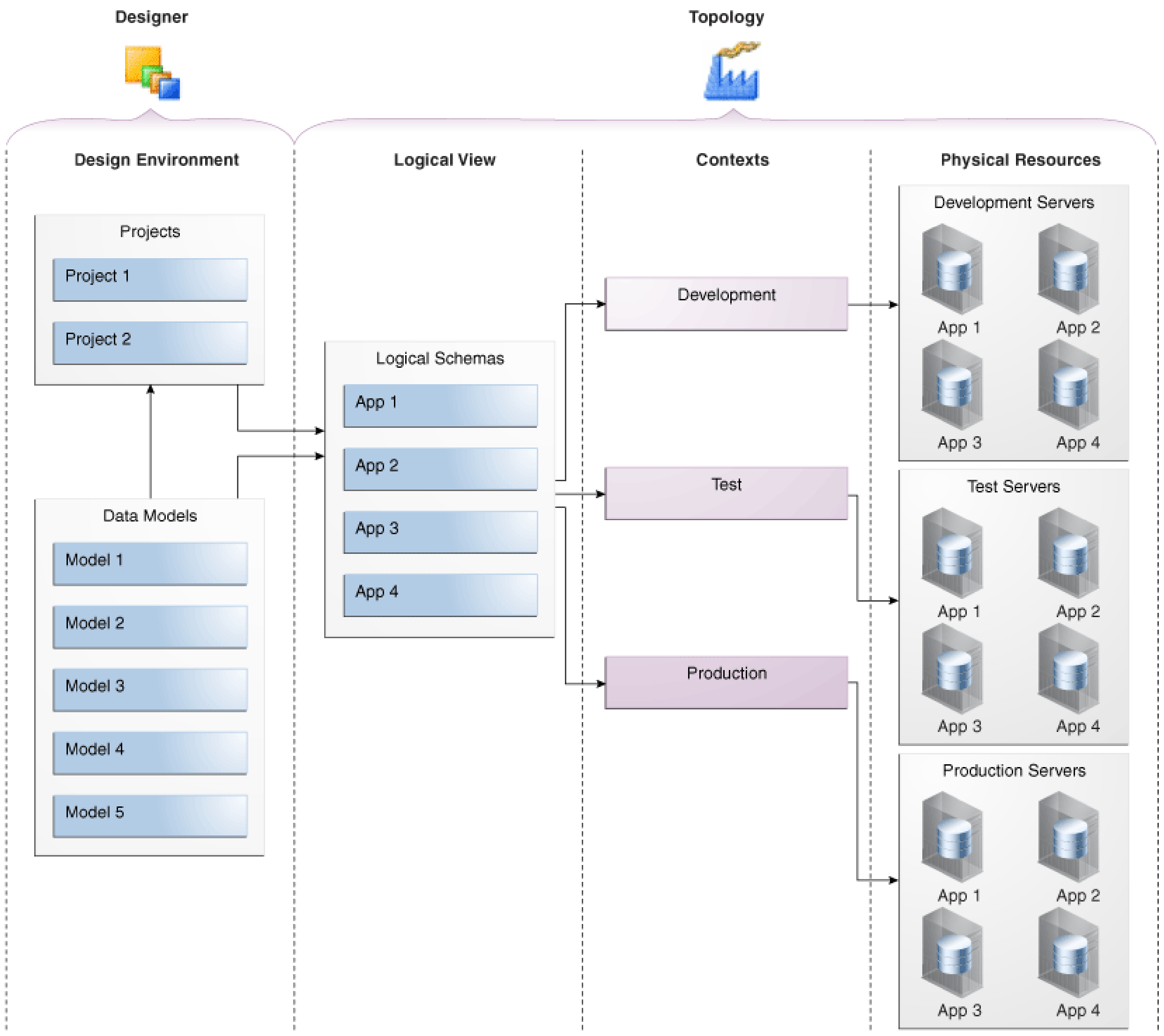
Physical schemas indicate the physical location of the datastores (tables, files, topics, queues) inside a data server. All the physical schemas that need to be accessed have to be registered under their corresponding data server, physical schemas are used to prefix object names and access them with their qualified names. When creating a physical schema, you need to specify a temporary, or work schema that will store temporary or permanent objects needed at runtime.

A logical schema is an alias that allows a unique name to be given to all the physical schemas containing the same datastore structures. The aim of the logical schema is to ensure the portability of procedures and models on different design-time and run-time environments.

A Context represents one of these environments. Contexts are used to group physical resources belonging to the same environment.

Typical projects will have separate environments for Development, Test and Production. Some projects will even have several duplicated Test or Production environments. For example, you may have several production contexts for subsidiaries running their own production systems (Production New York, Production Boston, and so forth). There is obviously a difference between the logical view of the information system and its physical implementation as described in Figure 21-1.

Figure 21-1 Logical and Physical View of the Infrastructure



The logical view describes logical schemas that represent the physical schemas of the existing applications independently of their physical implementation. These logical schemas are then linked to the physical resources through contexts.

Designers always refer to the logical view defined in the Topology. All development done therefore becomes independent of the physical location of the resources they address. At runtime, the logical information is mapped to the physical resources, given the appropriate contexts. The same scenario can be executed on different physical servers and applications simply by specifying different contexts. This brings a very flexible architecture where developers don't have to worry about the underlying physical implementation of the servers they rely on.

22

Life Cycle Management in Oracle Data Integrator

Oracle Data Integrator integrates with Version Control Systems (VCS) to enable you to version control ODI objects.
This chapter includes the following topics:

- Oracle Data Integrator integration with Version Control Systems
- Support for Separation of Deployment and Development Environments

Oracle Data Integrator integration with Version Control Systems

You can integrate ODI with an external Version Control Systems (VCS) to enable version control of ODI objects. You can store versioned copies of the ODI objects into an external VCS repository. As VCS relies on file-based storage, ODI objects are stored as XML files in the VCS repository.

Note:

Currently ODI supports only Apache™ Subversion®.

You can add un-versioned ODI objects to VCS, retrieve older versions of ODI objects from VCS, and view the differences between two versions of ODI objects from within ODI.

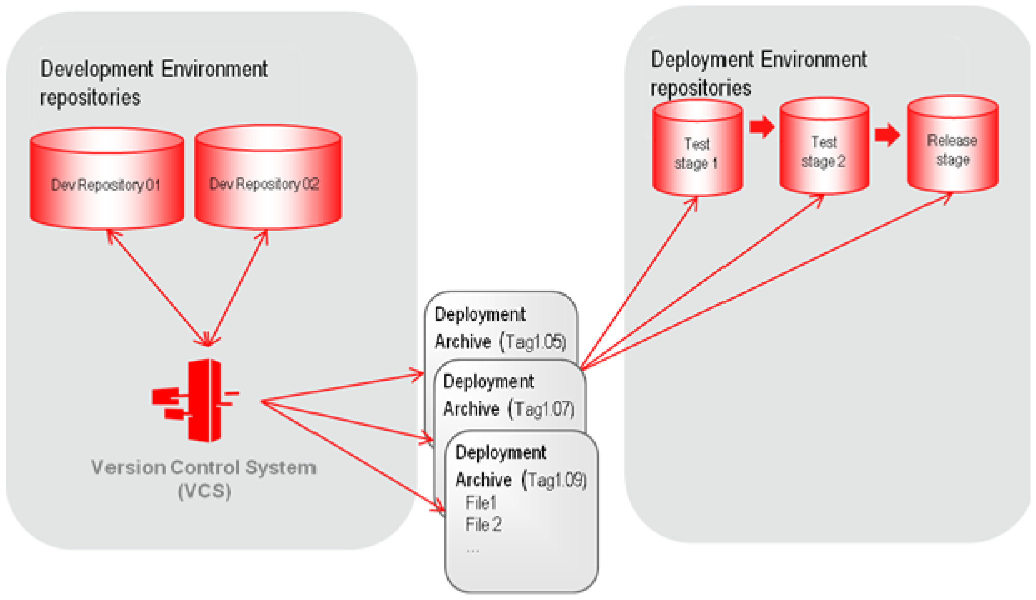
For more information, see the Integrating ODI with Version Control Systems section in *Developing Integration Projects with Oracle Data Integrator*.

Support for Separation of Deployment and Development Environments

ODI supports separation of the development environment and deployment environment, such as testing or production, where the ODI objects are not modified. ODI object updates in the deployment environments are now managed using Deployment Archives, which are the vehicle to release ODI objects from development to testing or production.

The following diagram depicts typical repositories in development and deployment environments.

Figure 22-1 Repositories in Development and Deployment Environments


Development Environment repositories
Dev Repository 01
Dev Repository 02
Version Control System (VCS)
Deployment Archive (Tag1.05)
Deployment Archive (Tag1.07)
Deployment Archive (Tag1.09)
File1
File 2
...
Deployment Environment repositories
Test stage 1
Test stage 2
Release stage

The repositories in the development environment are connected to the version control system. These repositories are used to actively develop the ODI objects. Once the ODI objects are ready for testing or production, the deployment archives are created with them. These deployment archives are applied to a test repository for verification and upon successful testing are promoted to the production repositories.

The patches applied to the deployment environment using the deployment archives can be managed through the ODI UI, which allows you to audit the list of applied patches and also lets you to rollback any bad patches.

For detailed information, see the Release Management section in *Developing Integration Projects with Oracle Data Integrator*.

Part IV

Maintaining Your Oracle Data Integrator Cloud Service

Advice on how to maintain Oracle Data Integrator Cloud.

Topics:

23 Troubleshooting Oracle Data Integrator

Learn techniques for troubleshooting issues with Oracle Data Integrator (ODI).

General Troubleshooting Tips

Troubleshoot most common issues with Oracle Data Integrator installation.

If you encounter an error during installation:

- Read the *Oracle Fusion Middleware Release Notes* for the latest updates. The most current version of the release notes is available on the Oracle Technology Network (OTN) at:

  `http://www.oracle.com/technetwork/documentation/index.html#middleware`

- Verify your system and configuration is certified. Refer to the *System Requirements and Supported Platforms for Oracle Fusion Middleware* document on the Oracle Fusion Middleware Supported System Configurations page:

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

- Verify your system meets the minimum system requirements. Refer to the System Requirements and Specifications document:

  `http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-requirements-100147.html`

- If you entered incorrect information on one of the installation screens, return to that screen by clicking **Back** until you see the screen.
- If an error occurred while the installer is copying or linking files:
  1. Note the error and review the installation log files.
  2. Remove the failed installation by following the steps in: .

     Deinstalling or Reinstalling Oracle Data Integrator
  3. Correct the issue that caused the error.
  4. Restart the installation.

Troubleshooting Using Installation and Configuration Log Files

Use the installation and configuration log files to troubleshoot Oracle Data Integrator issues.

The installation and configuration log files contain useful information to help troubleshoot any issues you may encounter. For more information about these log files and their contents, see Configuring Installation and Configuration Log Files in *Installing Software with the Oracle Universal Installer*.

Verifying ODI Client and Repository Compatibility

Troubleshoot potential compatibility issues between Oracle Data Integrator (ODI) and Repository versions.

If you are using Oracle Data Integrator with other Oracle Fusion Middleware products, make sure you have read Oracle Data Integration Interoperability with Other Fusion Middleware Products in *Understanding Interoperability and Compatibility*.

Oracle strongly recommends that you use identical versions of different ODI components such as ODI Studio, ODI Console, standalone and J2EE Agents, and executable scripts for a specific site or platform. In addition, you should make sure your Master and Work Repository versions are compatible with these components (Table 23-1).

Table 23-1 ODI and Repository Compatible Versions

ODI Version	Repository Version
12*c* (12.2.1.2.0)	05.01.02.02
12*c* (12.2.1.1.0)	05.01.02.02
12*c* (12.1.2)	05.01.01.16
11*g* Release 1 (11.1.1.7.0)	04.03.04.02
11*g* Release 1 (11.1.1.6.0)	04.03.04.02

Need More Help?

Access additional help through My Oracle Support.

If this appendix does not solve the problem you encountered, try looking for a solution on My Oracle Support (formerly Oracle*MetaLink*):

`https://support.oracle.com/`

If you are unable to find a solution for your problem, open a service request.