Oracle® Cloud

Using Data Integration Platform Cloud

E83692-22
August 2019

Oracle Cloud Using Data Integration Platform Cloud,

E83692-22

Primary Author: Oracle Corporation

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

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

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

Contents

Preface

Part I Before You Begin

1 What is Oracle Data Integration Platform Cloud?

2 Get Started

3 Before You Perform Tasks in Data Integration Platform Cloud

Part II Perform Tasks in Data Integration Platform Cloud

4 Synchronize Data

10 Create Policies

11 Maintain Oracle Data Integration Platform Cloud

Part III Perform Tasks on Hosted VMs

12 Replicate Data to Oracle Autonomous Data Warehouse Cloud

13 Analyzing Data Streams

14 Integrating Data

18 Configure GoldenGate

Part V Appendix

A Terminology

B What's Certified for Hosted VM Tasks

C Frequently Asked Questions for Data Integration Platform

D Kafka Connect Client Libraries

Preface

Topics

Audience

Using Oracle Data Integration Platform Cloud is intended for database administrators who are responsible for managing the organization’s databases, E-TL developers who move and transform data to make it more accessible to users, data engineers who analyze data streams, and data stewards who ensure data elements are being used, are fit, and have adequate documentation on usage.

Documentation Accessibility

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

Access to Oracle Support

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

Related Resources

See these Oracle resources:

- Getting Started with Oracle Cloud
- Oracle Public Cloud

  `http://cloud.oracle.com`

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.

Part I

Before You Begin

Oracle Data Integration Platform Cloud is a unified platform for real-time data replication, data transformation, data quality, and data governance. Before you get started, understand what you can do with Data Integration Platform Cloud, obtain an Oracle Cloud account, and create a Data Integration Platform Cloud instance. When your instance is ready, you can then log in to Data Integration Platform Cloud to download and configure your agents and create connections to your data sources.

Topics:

1 What is Oracle Data Integration Platform Cloud?

Oracle Data Integration Platform Cloud (DIPC) is a unified platform for real-time data replication, data transformation, data quality, and data governance. Learn about Oracle Data Integration Platform Cloud features, its common use cases, and the different components of its architecture. Understand what features are available in different types of Data Integration Platform Cloud instances and editions, and choose the ones that are most suitable for your data integration needs.

Topics:

Why Use Oracle Data Integration Platform Cloud?

Data Integration Platform Cloud Data Integration Platform Cloud helps migrate and extract value from data by bringing together capabilities of a complete Data Integration, Data Quality and Data Governance solution into a single unified cloud based platform. You can use Data Integration Platform Cloud to move, transform, cleanse, integrate, replicate, analyze, and govern data.

With Data Integration Platform Cloud you can:

- Perform seamless batch and real-time data movement among cloud and on-premises data sources.
- Synchronize an entire data source or coup high volumes of data in batches to a new Oracle Database Cloud deployment. You can then access, profile, transform, and cleanse your data.
- Stream data in real time to new data sources, perform data analysis on streaming data, and keep any number of data sources synchronized.
- Perform bulk transformation by importing and executing Oracle Data Integrator scenarios.
- Copy or move data from flat files or on-premises data sources to Oracle Data Lake. This is applicable only on the Data Integration Platform Cloud Classic instance.

- Integrate with big data technologies.

The most common uses of Data Integration Platform Cloud are:

- Accelerate Data Warehouses
- Automate Data Mart generation
- Migrate data without any down time (zero down time migration)
- Support redundancy through active-active model
- Integrate Big Data
- Synchronize data
- Replicate data to Kafka and bootstrap stream analytics.
- Monitor data health
- Profile and validate data

Data Integration Platform Cloud Certifications

The following sections provide information on data sources, operating systems and platforms that you can use to as connections to create tasks and run jobs for Data Integration Platform Cloud (DIPC) and Data Integration Platform Cloud Classic (DIPC-C) instances.

What’s Certified for each of the Data Integration Platform Cloud Tasks

Before you create tasks, you must have an agent set up and running. Review what’s certified for agents first, before you review what’s certified for your tasks.

- Agent Certifications
- What’s Certified for Synchronize Data?
- What’s Certified for ODI Execution?
- What’s Certified for Replicate Data?
- What’s Certified for Add Data to Data Lake?
- What’s Certified for Data Preparation

Other Links

If none of these tasks work as a data integration solution for you, then refer to What's Certified for Hosted VM Tasks for a do-it-yourself approach.

High Level Concepts in Oracle Data Integration Platform Cloud

Data Integration Platform Cloud offers a number of different tasks to meet your data integration needs. Here’s a high-level overview of each task.

Data Integration Platform Cloud’s elevated integration tasks enable you to organize, consolidate, cleanse, transform, synchronize, replicate, and store data in the cloud. Supported data sources for these tasks include:

- Oracle databases
- Oracle Object Storage Classic
- Autonomous Data Warehouse
- Flat files
- MySQL databases
- SQL Server databases
- Kafka
- Salesforce

You create Connections to identify your data source in Data Integration Platform Cloud. After a Connection is created, Data Integration Platform Cloud harvests the metadata from the schema associated with that Connection, then profiles the data and calculates the popularity. This metadata is referred to as a **Data Entity**. You can view and edit data entities in the **Catalog**.

To learn about creating Connections, see Create a Connection.

Integration tasks that you can perform in Data Integration Platform Cloud include:

- Synchronizing Data
- Replicating Data
- Executing an ODI Scenario
- Preparing Data
- Adding Data to a Data Lake

Synchronize Data

The Synchronize Data Task enables you to synchronize data between two Cloud databases, between an on-premises database and a cloud database, or between two on-premises databases. You can configure your Synchronize Data task to include Initial Load and/or Replication. You can also select the data entities to include in your task.

To learn about the data sources supported for this task, see What's Certified for Synchronize Data?.

For information about creating this task, see Create a Synchronize Data Task.

Replicate Data

The Replicate Data Task of Data Integration Platform Cloud captures changes in your data source and updates the target in real time.

To learn about the data sources supported for this task, see What's Certified for Replicate Data?

For more information about creating this task, see Replicate Data.

ODI Execution

Perform data transformations and bulk operations using new or existing Scenarios from ODI.

To learn about the data sources supported for this task, see What's Certified for ODI Execution?

For more information about creating this task, see ODI Execution.

Prepare Data

The Prepare Data Task enables you to harvest data from a source Connection, then perform various transformations on that data.

To learn about the data sources supported for this task, see What’s Certified for Data Preparation.

For information about creating this task, see Create a Data Preparation Task.

Add Data to Data Lake

A Data Lake is a repository used to store vast amounts of data in its natural form until you're ready to use it. When you create a Data Lake in Data Integration Platform Cloud, you can find it in the Catalog as a Data Asset. You can then create an Add Data to Data Lake task, where you can add data from a variety of sources, including relational databases or flat files. You can also create an Execution Environment to run the Add Data to Data Lake task from a Spark on Big Data Cloud or YARN.

To learn about the data sources supported for this task, see What’s Certified for Add Data to Data Lake?

For information about creating this task, see Add Data to Data Lake.

Monitor Jobs

A Job is created when a task is run. You can monitor Job actions in the Monitor dashboard. Select a Job to see its details as it’s running or review Job history after its completed.

For information about monitoring jobs, see Monitor Jobs

Tour Data Integration Platform Cloud

Take a tour through Data Integration Platform Cloud, where you can perform all your real-time data replication, integration, transformation, streaming, analytics and data quality management from a single platform.

Using Data Integration Platform Cloud, you can move your data quickly and easily to Oracle Autonomous Data Warehouse Cloud, smoothly create connections between data sources and targets, and execute a process for bulk data movement. This product tour takes you through the Data Integration Platform Cloud interfaces and various features.

▶ Video

Data Integration Platform Cloud versus Data Integration Platform Cloud Classic

Oracle offers Data Integration Platform Cloud in two forms, Oracle-managed Data Integration Platform Cloud and user-managed Data Integration Platform Cloud Classic.

The following table outlines the differences between Data Integration Platform Cloud and Data Integration Platform Cloud Classic. It also provides information about the services that are available in each of them. Read this information carefully and choose the one that’s appropriate for you.

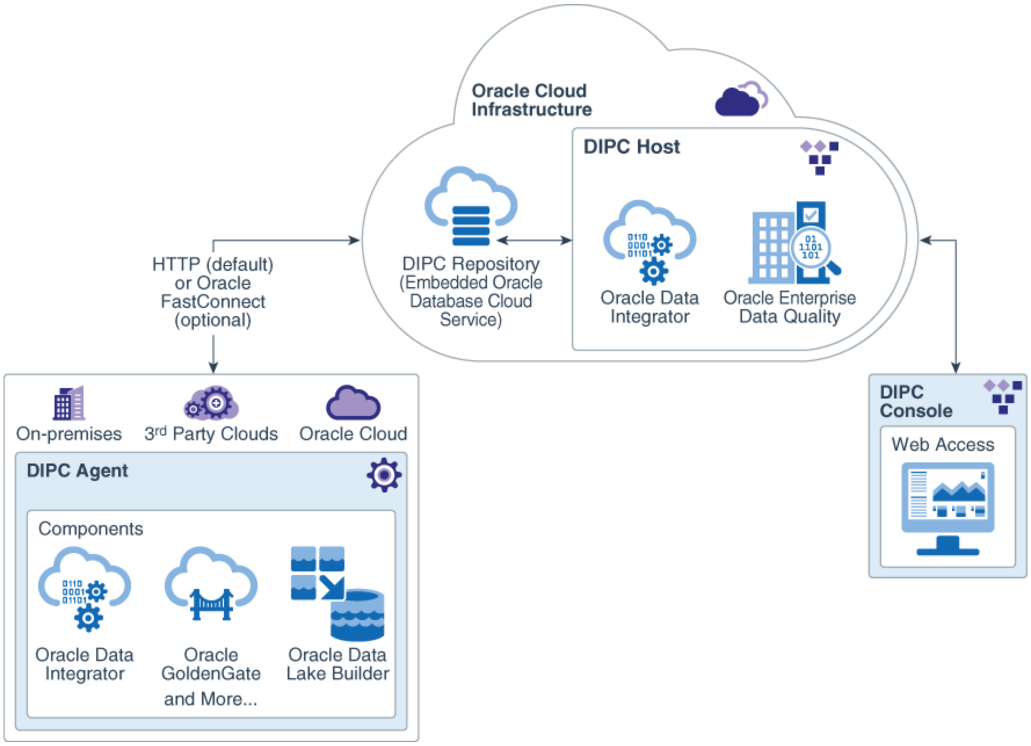
Feature	Data Integration Platform Cloud	Data Integration Platform Cloud Classic
Infrastructure	Oracle Cloud Infrastructure hosts the DIPC instances. Oracle Cloud Infrastructure is designed for automating, scaling, and managing containerized applications. Data regions are automatically selected based on your location.	Oracle Cloud Infrastructure Classic hosts the DIPC Classic instance. See Data Regions to select a proper region for your instance.
Management	Instance patching, upgrade, rollback, start, stop, and restart operations are automatic. The Instance Administration page provides the details of the Oracle-managed backups.	You have to manage instance patching, upgrade, rollback, start, stop, and restart operations. For example, if you need backups, you have to create an object storage container either before or during instance creation.
Access to Data Integration applications	VM access is not available with Oracle-managed instances. The DIPC host includes Oracle Data Integrator and Oracle Enterprise Data Quality installations. You can access Oracle Enterprise Data Quality Director using any web browser. The Oracle-managed DIPC instance does not provide access to the Oracle Data Integrator Studio on the instance itself. You won't have access to the database and storage associated with your instance.	The DIPC host includes the Oracle GoldenGate, Oracle Data Integrator, and Oracle Enterprise Data Quality installations. You can access Oracle Data Integrator Studio and Oracle GoldenGate using a VNC Server, and Oracle Enterprise Data Quality Director using any supported web browser.
Connection Type	By default, the DIPC instance connects with the DIPC agent using HTTP protocol. You can choose to use Oracle FastConnect instead of HTTP.	By default, the DIPC instance connects with the DIPC Agent using HTTP protocol. You can choose to use Oracle FastConnect or Oracle VPNaaS instead of HTTP.
Delivering Data to Big Data and MySQL	You can deliver Big Data using binaries obtained from Big Data (OGG) agent.	You can deliver Big Data and MySQL by setting up Oracle GoldenGate's Big Data and MySQL binaries for heterogeneous replication through the VM hosted on the DIPC instance. For a complete list of supported sources and targets see, Oracle GoldenGate (OGG) Certifications.

Feature	Data Integration Platform Cloud	Data Integration Platform Cloud Classic
Scaling up and down, the compute shape of a node	Not applicable.	Supported.
Stream Analytics	Not applicable.	Available.

Data Integration Platform Cloud Architecture

The Data Integration Platform Cloud instance is set up on Oracle Cloud Infrastructure. The following diagram shows the relationship between Oracle Data Integration Platform Cloud and the on-premises and cloud data sources.

Figure 1-1 Data Integration Platform Cloud Architecture



The Data Integration Platform Cloud (DIPC) instance architecture includes the following components:

- **DIPC host** - The DIPC host is located on Oracle Cloud Infrastructure. The DIPC host includes Oracle Data Integrator and Oracle Enterprise Data Quality installations. You can access Oracle Enterprise Data Quality Director using any web browser. The Oracle-managed DIPC instance does not provide access to the Oracle Data Integrator Studio on the instance itself.
- **DIPC repository** - The DIPC repository is located on an embedded Oracle Database Cloud Service within Oracle Cloud Infrastructure. The DIPC repository

stores metadata, such as connection details of the data sources, data objects, tasks, jobs, catalog, and so on. This metadata is required to run the DIPC tasks. DIPC does not store any type of user information.

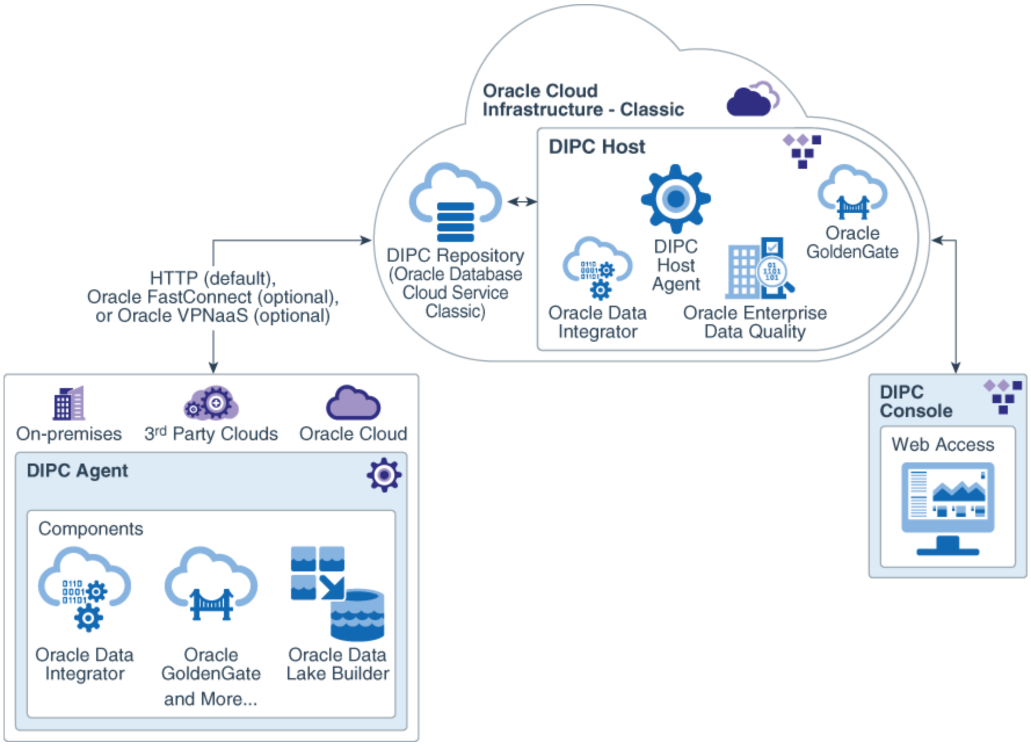
- **DIPC agent** - The DIPC agent establishes a connection between the DIPC host and the on-premises and cloud data sources. The DIPC agent also orchestrates the jobs that you run on the data sources from the DIPC console. You can download the DIPC agent from the DIPC console and install it on any machine, on-premises or cloud that has access to your data sources. For example, you can install the DIPC agent on premises, or on cloud, such as Oracle Cloud VMs or Third-Party Cloud VMs.
- **Components** - The Components comprise a set of binary files and agent properties file that you need to set up to run the DIPC tasks. Based on the DIPC tasks you want to run, you need to select the appropriate components when you download the DIPC agent from the DIPC console. Depending on the components you select, appropriate files are included in the DIPC agent package, which you download as a zip file. You need to copy this zip file and unzip it on the machine where you want to configure the DIPC agent and the components.
- **Connection type** - By default, the DIPC instance connects with the DIPC agent using HTTP protocol. You can choose to use Oracle FastConnect instead of HTTP.
- **DIPC console** - The DIPC console provides an easy-to-use graphical user interface that lets you perform the data integration tasks on your data sources. You can access the DIPC console using any supported web browser.

Data Integration Platform Cloud Classic Architecture

This topic only applies to Data Integration Platform Cloud Classic.

The Data Integration Platform Cloud Classic instance is set up on Oracle Cloud Infrastructure Classic. The following diagram shows the relationship between Oracle Data Integration Platform Cloud and the on-premises and cloud data sources.

Figure 1-2 Data Integration Platform Cloud Classic Architecture



The Data Integration Platform Cloud Classic instance architecture includes the following components:

- **DIPC host** - The DIPC host is located on Oracle Cloud Infrastructure. The DIPC host includes the DIPC host agent, Oracle GoldenGate, Oracle Data Integrator, and Oracle Enterprise Data Quality. You can access Oracle Data Integrator Studio and Oracle GoldenGate using a VNC Server, and Oracle Enterprise Data Quality Director using any supported web browser.
- **DIPC repository** - The DIPC repository is located on the Oracle Database Cloud Service Classic within the Oracle Cloud Infrastructure. The DIPC repository stores metadata, such as connection details of the data sources, data objects, tasks, jobs, catalog, and so on. This metadata is required to run the DIPC tasks. DIPC does not store any type of user information.
- **DIPC agent** - The DIPC agent establishes a connection between the DIPC host and the on-premises and cloud data sources. The DIPC agent also orchestrates the jobs that you run on the data sources from the DIPC console. You can download the DIPC agent from the DIPC console and install it on any machine, on-premises or cloud that has access to your data sources. For example, you can install the DIPC agent on premises, or on cloud, such as Oracle Cloud VMs or Third-Party Cloud VMs.
- **DIPC host agent** - The DIPC host agent is a pre-configured ready-to-use agent that is located on the DIPC host. This agent works in the same manner as the DIPC agent, but only with the cloud data sources and components. If you want to use on-premises data sources or components, you must either download and use the DIPC agent, or set up a VPN connection so that the DIPC host agent can access the on-premises data sources.

- **Components** - The Components comprise a set of binary files and agent properties file that you need to set up to run the DIPC tasks. Based on the DIPC tasks you want to run, you need to select the appropriate components when you download the DIPC agent using the DIPC console. Depending on the components you select, appropriate files are included in the DIPC agent package, which you download as a zip file. You need to copy this zip file and unzip it on the machine where you want to configure the DIPC agent and the components.
- **Connection type** - By default, the DIPC instance connects with the DIPC Agent using HTTP protocol. You can choose to use Oracle FastConnect or Oracle VPNaaS instead of HTTP.
- **DIPC console** - The DIPC console provides an easy-to-use graphical user interface that lets you perform the data integration tasks on your data sources. You can access the DIPC console using any supported web browser.

What are the Oracle Data Integration Platform Cloud Editions?

Data Integration Platform Cloud is available in three editions: Standard, Enterprise, and Governance. Each edition provides a different set of features. Understand what features are available in each of the editions and choose the one most suitable for your requirements.

Edition:	Features
Standard	High-performance Extract-Transform-Load (ETL) functions. You can bulk copy your data sources, and then extract, enrich, and transform your data.
Enterprise	In addition to the features of the Standard edition, you have access to execute all the integration tasks from the Data Integration Platform Cloud console, Stream Analytics, and Big Data technologies.
Governance	In addition to the features of the Enterprise edition, you have access to the data quality, data profiling, and data governance features. You can profile, cleanse, and govern your data sources using customized dashboards.

2 Get Started

To get started with Oracle Data Integration Platform Cloud, subscribe to Oracle Cloud, create a Data Integration Platform Cloud instance, add users, and then provide access to your users.

Topics:

Before You Begin with Oracle Data Integration Platform Cloud

Before you begin with Oracle Data Integration Platform Cloud, you should have:

1. An Oracle Cloud account. See About Oracle Cloud Accounts.
2. Your Oracle Cloud account details. You can locate your Oracle Cloud account details in the post-activation email that you receive on creating the account.
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 Integration Platform Cloud

Here’s how to get started with free Oracle Data Integration Platform 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 Integration Platform Cloud service.

   See Access Oracle Data Integration Platform Cloud.

To grant access to others:

- Learn about user accounts and roles.

  See Assign Roles for Data Integration Platform Cloud Classic Instances.

- 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*

Access Oracle Data Integration Platform Cloud

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

To access Oracle Data Integration Platform Cloud:

1. Sign in to your Oracle Cloud account. If this is your first time signing in, refer to the Welcome email you received when you purchased your subscription for more information.
2. Open the navigation menu in the upper left corner, select **Platform Services**, and then **Data Integration Platform** or **Data Integration Platform Classic**, depending on the type of service you have.
3. You can either create a new Data Integration Platform Cloud instance, or if there are instances available, select one to access.

Assign Oracle Data Integration Platform Cloud Roles

This topic only applies to Data Integration Platform Cloud Classic.

Data Integration Platform Cloud is integrated with Oracle Identity Cloud Service for security and access management. You can use this service to create users and assign roles to them.

Topics:

- Assign Roles for the Data Integration Platform Cloud Classic Service
- Assign Roles for Data Integration Platform Cloud Classic Instances

Assign Roles for the Data Integration Platform Cloud Classic Service

This topic only applies to Data Integration Platform Cloud Classic.

Service roles determine what a user can do within the Data Integration Platform Cloud Classic service, such as create and delete instances.

Assign the following roles to users who will create and delete instances:

- **DICS_ENTITLEMENT_ADMINISTRATOR**: This role enables users to create and delete Data Integration Platform Cloud service instances and perform all tasks available in the **Services**, **Activity** and **SSH** pages of Data Integration Platform Cloud.
- **Compute.Compute_Operations**: For the Compute service, this role enables users to create Oracle Public Cloud service instances.

To assign roles to the service:

1. Log in to Oracle Cloud as a user with the IDCS Administrator role.
2. From the navigation menu in the upper left corner, select **Identity**, **Federation**, and then your Identity Cloud Service provider.
3. Select a user, and then click **Manage Service Roles**.
4. Click the menu for **DICS**, and then select **Manage service access**.
5. Select **DICS_ENTITLEMENT_ADMINISTRATOR**, and then click **Save Role Selections**.
6. Click the menu for **Compute**, and then select **Manage service access**.
7. Select **Compute.Compute_Operations**, and then click **Save Role Selections**.
8. Click **Apply Service Role Settings**.

Assign Roles for Data Integration Platform Cloud Classic Instances

This topic only applies to Data Integration Platform Cloud Classic.

Instance roles to determine what a user can do within a specific Data Integration Platform Cloud instance.

You can access the following with each Data Integration Platform Cloud instance:

- **Data Integration Platform** console, available through the action menu of any Data Integration Platform Cloud instance
- **Oracle Data Integrator** console, available through the user menu of Data Integration Platform Cloud console
- **Enterprise Data Quality** console (available with Governance edition), available through the user menu of Data Integration Platform Cloud console
- **WebLogic Server** console, available through the action menu of any Data Integration Platform Cloud instance
- **Fusion Middleware** console, available through the action menu of any Data Integration Platform Cloud instance

The following table lists the instance level roles that you can assign to your users.

Role in Identity Cloud Service	Equivalent ODI Profile	Description	Is Admin?
Administrator	Profiles ending in ADMIN	Service application administrator role	Yes
Developer	DESIGNER (Use in ODI Studio for ELT designs)	Service application developer role	No

Role in Identity Cloud Service	Equivalent ODI Profile	Description	Is Admin?
User	OPERATOR (Use in ODI console for job execution and status review)	Service application user role	No. For example, this user will not have the **Management** tab in the ODI console available to them.

To assign roles to instances:

1. Log in to Oracle Cloud as a user with the IDCS Administrator role.
2. From the navigation menu in the upper left corner, select **Identity**, **Federation**, and then your Identity Cloud Service provider.
3. Select a user, and then click **Manage Service Roles**.
4. Click the menu for **DICS**, and then select **Manage instance access**.
5. Select the roles to assign.
6. Click **Save Instance Settings**, and then **Apply Service Role Settings**.

For more information, see Manage Security for Service Instances.

Create Instances for Data Integration Platform Cloud

There are several ways that you can create a Data Integration Platform Cloud instance, depending on your requirements and experience level.

Topics:

- Create Data Integration Platform Cloud Instances
- QuickStart Data Integration Platform Cloud Classic Instances
- Create Data Integration Platform Cloud Classic Instances

Create Data Integration Platform Cloud Instances

This topic does not apply to Data Integration Platform Cloud Classic. It applies only to Data Integration Platform Cloud.

Follow these steps to create a Data Integration Platform Cloud instance.

You can also refer to the Create a Data Integration Platform Cloud Instance tutorial.

1. Log in to Oracle Cloud.
2. In the console, open the navigation menu in the upper left corner, and select **Platform Services**, and then **Data Integration Platform**.
3. On the Instances page, click **QuickStarts**.
4. Select a Data Integration Platform Cloud service edition, and then click **Create**.

QuickStart Data Integration Platform Cloud Classic Instances

This topic only applies to Data Integration Platform Cloud Classic.

Get started with Data Integration Platform Cloud Classic quickly and easily using the QuickStart option on your Service Overview page.

The QuickStart option provisions a Database Cloud Service instance followed by a Data Integration Platform Cloud Classic instance with the following details:

- Database Cloud Service: Oracle Database 12.1.0.2, Enterprise Edition. 1 PDB, 1 OCPU, 15 GB Memory
- Data Integration Platform Cloud Classic: Enterprise Edition, 1 OCPU, 15 GB Memory

Each of these instances are configured on one virtual machine. These two instances are configured to see each other and the schemas for the Data Integration Platform Cloud Classic instance are stored in the database instance. Neither of these instances have a backup option.

These instances are typically used for creating sandbox environments or in cases that don't require backup and recovery. The QuickStart templates give you the fastest and easiest way to create a Data Integration Platform Cloud Classic instance. All you need to do is provide an instance name and then click **Create**.

> **Note:**
>
> QuickStart instances are offered through **Oracle Cloud Stack**, which is an Oracle Cloud service that allows several Oracle Cloud instances stacked and created together.

QuickStart a user-managed instance:

1. Log in to Oracle Cloud.
2. In the console, open the navigation menu in the upper left corner, select **Platform Services**, and then **Data Integration Platform Classic**.
3. On the Data Integration Platform Cloud Classic Instances page, click **QuickStarts**.
4. On the QuickStarts page, enter a name for your instance in the **Instance Name** field, and then click **Create**.
5. In the Confirmation dialog, click **Download** to download the credential file that you'll need if you want to access your database and Data Integration Platform Cloud Classic VMs, and then click **Create**.
6. Click **Navigation Menu** to access the navigation pane, and then select **Cloud Stack**.
7. On the Stacks page, confirm that your instance appears in the list.

Allow some time to pass before using your QuickStart instance. When you refresh the page and the **Status: Creating Instance** no longer appears, you'll know your instance is ready to use.

Create Data Integration Platform Cloud Classic Instances

This topic only applies to Data Integration Platform Cloud Classic.

To create Data Integration Platform Cloud Classic instances, you must first provision an Oracle Database Cloud Service instance and optionally an Oracle Storage Classic container (for backup and recovery).

Topics:

- Find Your User-Managed Services and Backup Location
- Create an Object Storage Classic Container
- Provision Oracle Database and Exadata Cloud Instances
- Provision and Access an Oracle Data Integration Platform Cloud Classic Service Instance
- Configure VPN as a Service (VPNaaS)

Find Your User-Managed Services and Backup Location

This topic only applies to Data Integration Platform Cloud Classic.

Before you provision Storage Classic, Database, and Data Integration Platform Cloud instances, it may be helpful to determine the REST Endpoint url that you'll use for your Storage Classic container.

1. Log in to Oracle Cloud.
2. From the navigation menu, select **Storage Classic**, and then click **Account**.
3. Under **Account Information**, copy the **REST Endpoint** URL and paste it into a text editor to use later.

Create an Object Storage Classic Container

This topic only applies to Data Integration Platform Cloud Classic.

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 Selecting a Replication Policy for Your Service Instance

Interface	Resources
Web Console (Not available on Oracle Cloud at Customer)	Creating a Container Using the Web Console
RESTful API	Creating Containers 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 for Oracle Cloud Infrastructure Object Storage Classic File Transfer Manager*.
File Transfer Manager CLI	See Create Container in Command-Line Reference for *Oracle Cloud Infrastructure Object Storage Classic*.

To create an `archive` container, you must set the X-Storage-Class header to Archive. For more information, see Creating Archive Containers. (Not available on Oracle Cloud at Customer)

Creating a Container Using the Web Console

(Not available on Oracle Cloud at Customer)

1. Log in to the Oracle Cloud Infrastructure Object Storage Classic console.
2. Click **Create Container.**

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

   > **Note:**
   >
   > Ensure that the container name complies with the input restrictions mentioned in Character Restrictions.
4. Select **Standard** in the **Storage Class** field.
5. 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.
- For the syntax of `accountURL`, see About REST URLs for Oracle Cloud Infrastructure Object Storage Classic Resources.
- `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.

> **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 Error Code Reference for Object Storage Classic

cURL Command Example

Sample Cloud account with the following details:

- Account name: `acme`
- REST Endpoint URL: `https://acme.storage.oraclecloud.com/v1/Storage-acme`
- REST Enpoint (Permanent) URL: `https://storage-7b16fede61e1417ab83eb52e06f0e365.storage.oraclecloud.com/v1/Storage-7b16fede61e1417ab83eb52e06f0e365`

> **Note:**
>
> The REST Enpoint (Permanent) URL is displayed for accounts created after November 2017.

- Using the REST Enpoint URL obtained from the **REST Endpoint** field:

```
curl -v -X PUT \
     -H "X-Auth-Token: AUTH_tkb4fdf39c92e9f62cca9b7c196f8b6e6b" \
     https://acme.storage.oraclecloud.com/v1/Storage-acme/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
```

- Using the Service Permanent REST Endpoint URL from the **REST Endpoint (Permanent)** field:

  ```
  curl -v -X PUT \
       -H "X-Auth-Token: AUTH_tkb4fdf39c92e9f62cca9b7c196f8b6e6b" \
       https://storage-7b16fede61e1417ab83eb52e06f0e365.storage.oraclecloud.com/v1/
  Storage-7b16fede61e1417ab83eb52e06f0e365/FirstContainer
  ```

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

  ```
  > PUT /v1/Storage-7b16fede61e1417ab83eb52e06f0e365/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: storage-7b16fede61e1417ab83eb52e06f0e365.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 informationa bout setting the replication policy in the container, see Selecting a Replication Policy Using the REST API.

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

Provision Oracle Database and Exadata Cloud Instances

This topic only applies to Data Integration Platform Cloud Classic.

Learn how to create Oracle Database Cloud and Oracle Exadata Cloud instances.

Topics:

- Create an Oracle Database Cloud Service Instance
- Create an Oracle Database Exadata Cloud Service Instance

Create an Oracle Database Cloud Service Instance

This topic only applies to Data Integration Platform Cloud Classic.

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

Before You Begin

An SSH public/private key pair (Optional)

An SSH public key is used for authentication when you use an SSH client to connect to a compute node associated with the deployment. When you connect, you must provide the private key that matches the public key.

You can have the wizard create a public/private key pair for you, or you can create one beforehand and upload or paste its private key value. If you want to create a key pair beforehand, you can use a standard SSH key generation tool. See Generating a Secure Shell (SSH) Public/Private Key Pair.

After Your Database Deployment Is Created

After your database deployment is created, you should perform the following actions:

- **Update cloud tooling**

  While the base images used to create Database Cloud Service database deployments are updated regularly, it is possible that even more recent updates to the cloud tooling are available. Therefore, you should check for and apply any updates to the cloud tooling. See Updating the Cloud Tooling on Database Cloud Service.

- **Apply database patches**

  While the base images used to create Database Cloud Service database deployments are updated regularly, it is possible that a newer patch set update (PSU) or bundle patch (BP) is available. Therefore, you should check for and apply any database patches that are available. See Applying a Patch.

Create an Oracle Database Exadata Cloud Service Instance

This topic only applies to Data Integration Platform Cloud Classic.

Oracle Data Integration Platform Cloud supports capturing from, and delivering to Exadata Cloud Service. The replication is based on the remote capture and delivery architecture. Oracle Data Integration Platform Cloud captures from on-premises Oracle Databases and sends data to a Data Integration Platform Cloud instance. This instance connects to Database Cloud Service in Exadata Service through SQL*Net and performs remote delivery (integrated delivery).

You must create an Exadata Cloud Service database deployment that is properly configured for use as a Data Integration Platform Cloud replication target before you create a Data Integration Platform Cloud instance.

Before you can create a database deployment, you must have an active Exadata Cloud Service instance in place.

> **Note:**
>
> If you do not have an active service instance in place, then the Create Service wizard will not show the options required to create a deployment on Exadata Cloud Service.

Review Creating an Exadata Cloud Service Instance

You must complete the following fields when using the wizard.

- Service level: Oracle Database Exadata Cloud Service.
- Database release: This selection determines the version of Oracle Grid Infrastructure that is configured.

Database releases	Version of Oracle Grid Infrastructure
Oracle Database 11g Release 2 or Oracle Database 12c Release 1	Oracle Grid Infrastructure 12c Release 1 is installed and can only be used to support version 11.2 or version 12.1 database deployments. In this case, you cannot later use the Create Service wizard to create a version 12.2 database deployment.
Oracle Database 12c Release 2	Oracle Grid Infrastructure 12c Release 2 is installed and used to support all of your database deployments, including subsequent deployments that use an earlier Oracle Database release version.

  If you wish to deploy Oracle Database 12c Release 2 on a system that is already configured with Oracle Grid Infrastructure 12c Release 1, then you must manually upgrade to Oracle Grid Infrastructure 12c Release 2 and manually create the version 12.2 database deployment. For details see My Oracle Support note 2206224.1.

- Database edition: The only valid option for use with Exadata Cloud Service is **Enterprise Edition — Extreme Performance**.

Configure the database deployment for use as a replication database by setting the **Enable Oracle GoldenGate** option on the Service Details page of the wizard.

The target database must be network accessible on the listener port by:

- Specifically enabling network access to the Oracle Net Listener port. Review Enabling Access to a Compute Node Port.

  > **Note:**
  >
  > If you specifically enable access to the Oracle Net Listener port, ensure that you always use an encrypted Oracle Net connection. Review Using Network Encryption and Integrity.

- Or you can configure an IPSec VPN. This enables secure connectivity on any port between a customer network and the Oracle Public Cloud. Review Enabling IPSec VPN Access to Exadata Cloud Service.

Once you have created and properly configured an Exadata Cloud Service database deployment for use as a replication target, you can create an Oracle Data Integration Platform Cloud instance that uses it.

When provisioning your Database, you can select any Oracle Database Exadata Cloud Service you already have.

1. You first create the **Oracle Database Exadata Cloud Service** instance by launching the wizard. The wizard is invoked through the **Open Service Console** button on the Cloud My Services Dashboard related to the Exadata Cloud Service.
2. Provide all the Instance Details:

Instance Details fields	Description
Name	The name for your service instance is no longer than 25 characters, starts with a letter, not contains spaces or other special characters and is unique within the identity domain.
Data Center	The data center that will host your Exadata Cloud Service instance.
Plan	A plan is associated with a set of attributes that apply to a service. For Exadata Cloud Service only one plan is available.
Rack size	Select the option that describes the number of compute nodes and Exadata Storage Servers that you have.
Extra number of OCPUs (Cores)	Optionally you can enter a number of additional CPU cores that you want to enable.
Exadata System Name	The Exadata Database Machine environment. This name is also used as the cluster name for the Oracle Grid Infrastructure installation.
Database backups on Exadata Storage (Yes or No)	Check this option to configure the storage to enable local database backups on Exadata storage.
Administrator Details	Provide information about the administrator of your Exadata Database Machine environment

3. Create the Service instance by clicking in the **Create Service Instance** button.
4. In the Overview you can also review additional information: (like the Service Start Date and the Version).

> **Note:**
>
> When you run the Oracle Database Cloud Service provisioning wizard, always select an Oracle Database Exadata Cloud Service option. This action is to have the Oracle Database software pre-installed on an Oracle Exadata machine. Using this option, Database Service instances are created on available Exadata infrastructure.

5. When configuring your Oracle Database Cloud Service, you select Exadata System for your service instance.

6. Finally, when configuring your Oracle Data Integration Platform Cloud instance, you select the Database Cloud Service that has Exadata.

Follow the steps through the tutorial Get Started with Oracle GoldenGate Cloud Service

To enable Oracle Data Integration Platform Cloud replication, you need to file an SR to whitelist the GGCS IP address so that GGGCS can communicate with the Exadata DB instance with the SQL*Net connection. The SR would typically include the following information:

- IdentityDomain ID -xxxxxx of the GGCS VM
- Source Compute instance - GGCS VM IP address
- Exadata CS SQL*Net Port: 1521

You can find additional Exadata Cloud Service documentation at: Enabling Access to a Compute Node Port

> **Note:**
>
> - Oracle Data Integration Platform Cloud doesn't support Oracle database 12.2 capture yet.
> - With Exadata CS X6, you can complete the whitelist operation using the UI interface.

Provision and Access an Oracle Data Integration Platform Cloud Classic Service Instance

This topic only applies to Data Integration Platform Cloud Classic.

Learn how to create Data Integration Platform Cloud Classic instances, access a VM through SSH, and create access rules.

Topics:

- Create an Oracle Data Integration Platform Cloud Classic Service Instance
- Access a Virtual Machine Through a Secure Shell
- Configure Access Rules

Create an Oracle Data Integration Platform Cloud Classic Service Instance

This topic only applies to Data Integration Platform Cloud Classic.

You can provision an Data Integration Platform Cloud Classic service instance after you’ve created your Database Cloud Service instance.

To create a Data Integration Platform Cloud Classic service instance, you can either use the Create Service wizard as described here, or you can follow the steps in the

Create a Data Integration Platform Cloud Classic instance on Oracle Cloud Infrastructure Classic tutorial.

Before you Begin

When you create a Data Integration Platform Cloud Classic service instance, you provide information about the instance itself and its related resources. Therefore, make sure you have the required information about the following resources available:

- Your Database Cloud Service instance
- Your Oracle Cloud Storage container

Create an Oracle Data Integration Platform Cloud Classic Service Instance

To create a new Data Integration Platform Cloud Classic service instance:

1. Log in to Oracle Cloud.
2. In the Oracle Cloud Infrastructure console, open the navigation menu in the upper left corner, select **Platform Services**, and then **Data Integration Plaform Classic**.
3. Click **Create Instance**.
4. On the Service page, enter a name and description for the Data Integration Platform Cloud Classic service instance, and provide information about its high-level characteristics.
5. Click **Next**.
6. On the Service Details page, complete the following sections, and then click **Next**.
   - **Database Configuration**
     - **Oracle Database Cloud Service Instance Name:** Select an available Database Cloud Service instance.
     - **PDB Name:** Specify the pluggable database name (PDB) associated with the Database Cloud Service instance chosen from the **Associated DBAAS Service Name** list.
     - **Database Administrator Username** and **Password:** Enter the user name and password of the administration account for this Database Cloud Service instance.
   - **Backup and Recovery Configuration**
     - **Cloud Storage Container:** Enter the object storage location where backups of the service instance must be stored.

       Enter the URL of a container in Oracle Cloud Infrastructure Object Storage Classic using the format `rest_endpoint_url/containerName`. You can find the REST endpoint URL of the Oracle Cloud Infrastructure Object Storage Classic service instance in the Oracle Cloud My Services portal; for example, `https://acme.storage.oraclecloud.com/v1/MyService-acme/MyContainer`.

> **Note:**
>
> If the container that you specify doesn’t exist, then select the **Create Cloud Storage Container** check box to create the container automatically.

  - **Cloud Storage Username** and **Cloud Storage Password:** Enter the user name and password of the Oracle Cloud Infrastructure Object Storage Classic service user who created the container you specified earlier. If the container doesn’t exist, then enter the user name of a service administrator.
  - **Create Cloud Storage Container:** Select this check box to create the container automatically if the container that you specified doesn’t exist.
- **Weblogic Server Configuration**
  - **Compute Shape:** Select the compute shape for the Weblogic VMs.
  - **SSH Public Key:** Specify the Secure Shell (SSH) client public key that’s used for authentication when connecting to a node in your instance.

    Click **Edit** to display the SSH Public Key for VM Access dialog, and then specify the public key using one of the following methods:

    * Select **Key file name** and use your web browser to select a file on your machine that contains the public key.
    * Select **Key value** and paste the value of the public key into the text area. Be sure the value doesn’t contain line breaks or end with a line break.
    * Select **Create a New Key** if you want Oracle to generate a public/ private key pair for you.
  - **Administrator Username** and **Password:** Specify the administration user name and password for Weblogic.

    The password:

    * Must start with a letter.
    * Must be between 8 to 30 characters long.
    * Must contain at least one number.
    * Can include any number of dollar signs ($), hash signs (#), or underscores (_).

7. Click **Next.**

   On the Confirmation page, review the information listed. If you’re satisfied with the information, click **Create.**

Access a Virtual Machine Through a Secure Shell

This topic only applies to Data Integration Platform Cloud Classic.

You access the services and resources provided by the virtual machine (VM) of a cloud service instance through a Secure Shell (SSH) connection.

The SSH protocol uses two keys, a public key and a private key, to provide secure communication between two computers. You can use any SSH utility to generate a public/private key pair and later, to log in to your Oracle Data Integration Platform Cloud instance. For example, you can use PuTTY or you can use OpenSSH.

For Windows you can follow the steps through the tutorial:

- Connect to a Cloud VM on Windows with PuTTY

Configure Access Rules

This topic only applies to Data Integration Platform Cloud Classic.

Use access rules to define sources, destinations and ports to control network access, to or from Data Integration Platform Cloud Classic components.

To control network access to your Oracle Service instances, you can define access rules. For example, you can create rules that:

- Enable an Oracle Database node to access multiple ports on the WebLogic Managed Server nodes
- Enable public Internet access to a specific port on the WebLogic Administration Server node

Data Integration Platform Cloud Classic creates several default rules on a new service instance, such as public access to the WebLogic Administration Server node on port 22 for Secure Shell (SSH). Some of these are system rules, which cannot be disabled.

Make sure you consider the possible security implications before you open ports to external access.

1. Access your service console.
2. For the service that you want to modify, click **Manage this Service,** and then select **Access Rules.**
3. On the Access Rules page, click **Create Rule.**
4. Enter a name in the **Rule Name** field.
5. (Optional) Enter a description in the **Description** field.
6. In the **Source** field, select a source for the rule. The available source options depend on the topology of your service instance and may include:
   - **PUBLIC-INTERNET:** Any host on the Internet
   - **OTD_ADMIN_HOST:** The first Oracle Traffic Director (OTD) load balancer node
   - **OTD_OTD_SERVER:** All OTD load balancer nodes
   - **WLS_ADMIN_HOST:** The WebLogic Administration Server node
   - **WLS_ADMIN:** All WebLogic Administration Server nodes
   - **WLS_MS:** All WebLogic Server nodes
   - **DBaaS:*Name*:DB**: The database service named *Name*

   - **<custom>:** A custom list of addresses from which traffic should be allowed. In the field that appears below this one, enter a comma-separated list of subnets (in CIDR format, such as 10.11.12.13/24) or IPv4 addresses for which you want to permit access).

7. In the **Destination** field, select the destination node within this service instance.
   - **WSL_ADMIN:** The WebLogic Administration Server node
   - **WLS_ADMIN_HOST:** The WebLogic Administration Server node
   - **WLS_MS:** All WebLogic Server nodes
   - **OTD_ADMIN_HOST:** The Oracle Traffic Director (OTD) Administration Server node
   - **OTD_OTD_SERVER:** All Oracle Traffic Director (OTD) Manager Server nodes
8. In the **Destination Port(s)** field, enter the port or range of ports through which the source will be granted access to the destination.
9. In the **Protocol** field, select the TCP or UDP transport for this rule.
10. Click **Create**.

To return to either the Service page or the Overview page for the select service instance, click the locator links at the top of the page.

> **Note:**
>
> OTD has the same admin/managed server architecture as WebLogic Server. You can configure zero, one, or two load balancer nodes. The first load balancer node (OTD_ADMIN_HOST) runs the OTD admin server and managed server. The 2nd LB node (OTD_OTD_SERVER) runs an OTD managed server. OTD components can only be used as source, to allow traffic from them to other components.

To create custom access rules, refer to the Create Access Rules tutorial.

Configure VPN as a Service (VPNaaS)

This topic only applies to Data Integration Platform Cloud Classic.

You can connect your on-premises Oracle Database to Oracle Public Cloud (OPC) by using Virtual Private Network as a Service (VPNaaS) on Data Integration Platform Cloud Classic.

> **Note:**
>
> When you configure VPNaaS, you must provision Data Integration Platform Cloud Classic to use IP Network instead of a Shared Network.
> For details on how to configure IP Networks, see Configuring IP Networks.

Before you begin, you should already have your on-premises Oracle Database installed and configured, and instances for DIPC already provisioned on the IP network. For details on provisioning Data Integration Platform Cloud Classic instances, see Create an Oracle Data Integration Platform Cloud Classic Service Instance; or you can follow the steps in this tutorial, Create a Data Integration Platform Cloud Classic instance on Oracle Cloud Infrastructure Classic.

Topics:

- Create Public DVLBR and Mapping IP Networks
- Set up VPNaaS

Create Public DVLBR and Mapping IP Networks

This topic only applies to Data Integration Platform Cloud Classic.

You can create a parent Virtual Load Balancer Resource (VLBR) by using REST APIs, or from the Compute Classic service console. You can use a single VLBR (INTERNET_FACING) with multiple IPNetworks available in the same account. Here, you'll learn to create a child VLBR in the console.

Create a Child VLBR

1. Log in to the Oracle Cloud Instrastructure Cloud Classic console.
2. In the navigation menu, select **Compute Classic**.
3. Log in to the Compute Classic console, and then click the **Network** tab.
4. In the left navigation pane, click **Load Balancers** and then select **Create Load Balancer**.
5. Enter a **Name**, an **IPNetwork**, and a **Parent Load Balancer**.
6. Click **Create.**

After you create this child-load-balancer, you can create the JLS instances using this IPNetwork or you can use the IPNetwork (Primary) used in creating VLBR.

Finding IPNetwork used in creating the DVLBR

If you’ve already created a DVLBR (Parent VLBR) pointing to an IPNetwork, you can find it from the Compute Classic console.

1. Click the **Network** tab, and from the left navigation pane, click **IP Networks.**

   All the IP Networks are listed in the right pane.
2. Click the **Load Balancer**, and then select **Overview** to find the details.

Updating IPExchange

If a DVLBR already exists, select the IPNetwork that you pointed to, to create a parent VLBR (INTERNET_FACING). Update the IPExchange to point to an IPExchange used in the Parent VLBR IPNetwork.

> **Note:**
>
> The same applies for all other IPNetworks in the same account that you want to use to create any JLS (DIPC)-based services.

1. Click the **Network** tab, and from the left navigation pane, click **IP Exchanges.**
2. From the right pane, click **Create IPExchange.**
3. Enter a **Name** and a **Description**, and click **Create.**
4. Go back to the left navigation pane, click **IP Networks**, and then select **Create IP Network**
5. Select the **IPExchange** from the drop-down and click **Create.**

Set up VPNaaS

This topic only applies to Data Integration Platform Cloud Classic.

VPNaaS communicates with an on-premises network through a VPN gateway. You can use a third-party device to set up the VPN gateway.

For a list of supported third-party devices and configurations, see Setting Up a VPN Connection Using VPNaaS

Here are the high-level steps for configuring a VPN connection between an on-premises Oracle GoldenGate application and a Data Integration Platform Cloud Classic server by using VPNaaS.

1. OPC – IP Network and DIPC Instance Check
2. OPC – Provision a VPN Connection under VPNaaS
3. Create and Configure On-Premises Third-Party VPN Device
4. OPC and On-Premises – Verify Tunnel Status is Up
5. OPC and On-Premises – Verify VPN Connection

OPC – IP Network and DIPC Instance Check

This topic only applies to Data Integration Platform Cloud Classic.

From the Oracle Cloud Infrastructrure Compute Classic console, you can check the IP Network where Data Integration Platform Cloud Classic is provisioned.

1. In the Compute Classic console, and click the **Network** tab.
2. From the left navigation pane, click **IP Networks.**
3. Check the **IP Network** where Data Integration Platform Cloud Classic is provisioned.
4. Double-check Data Integration Platform Cloud Classic instance to see if it’s provisioned on the correct IP Network.

> **Note:**
>
> You must create the Internet Facing Load Balancer, before provisioning a DIPC Classic instance on an IP Network.
> You can create the Internet Facing Load Balancer by using REST APIs for Load Balancer services.
>
> For more details on the Load Balancer REST Endpoints, see REST API for Oracle Cloud Infrastructure Load Balancing Classic.

Here's an example of the **cURL** command syntax that can be used to create the **Load Balancer.**

```
curl -X POST -k -H 'X-ID-TENANT-NAME: gse00013735' -u
'cloud.admin:Pass1234' -H 'Content-Type: application/
vnd.com.oracle.oracloud.lbaas.VLBR+json' -i 'https://
lbaas-71bde0c0714f41cd9cea9a15f414ece3.balancer.oraclecloud.com/vlbrs'
--data ' {
 "name":"dipc-manl-public-lbr-central-ipnet",
 "disabled":"false",
 "region":"uscom-central-1",
 "scheme":"INTERNET_FACING",
 "compute_site":"uscom-central-1",
 "ip_network_name":"/Compute-588688908/cloud.admin/DIPCIPN"
}'| grep "{" | python -m json.tool
```

5. After you execute the **cURL** command, ensure that the status has changed to **Creation in Progress**.
6. To check the successful creation of the **Internet Facing Load Balancer**, execute this **REST API** example by using **cURL**.

```
curl -X GET -k -H 'X-ID-TENANT-NAME: gse00013735' -u
'cloud.admin:Pass1234' -i 'https://
lbaas-71bde0c0714f41cd9cea9a15f414ece3.balancer.oraclecloud.com/vlbrs'
| grep "{" | python -m json.to
```

The **Load Balancer** must display a**HEALTHY** status after it is created.

OPC – Provision a VPN Connection under VPNaaS

This topic only applies to Data Integration Platform Cloud Classic.

Create the VPN connection by using the Oracle Cloud Compute Classic console under the **VPN** menu option.

1. Navigate to the Compute Classic console, and click the **Network** tab.
2. From the left navigation pane, click **IP Networks**, and then **VPN**.
3. Click **VPN**, and then under **VPNaaS**, click **VPN Connections**.
4. Enter a **Name**, and set the following:

- **IP Network:** Select the IP network where Data Integration Platform Cloud Classic is provisioned.
- **Connected IP Networks:** Displays the IP networks that are reachable over this VPN connection.

  The VPN connection allows you to access all IP networks that are added to the same IP network exchange as the specified IP networks. For example, if there's only one IP Network and there's no IP Exchange attached to that network, this field is blank.
- **vNICsets:** Select the vNICsets that are associated with the DIPC instance.
- **Customer Gateway:** Enter the WAN IP or public facing IP address of the VPN device in the on-premises network.

  > **Note:**
  >
  > Sometimes, the WAN IP is different from the public IP address, especially if the VPN device has Network Address Translation (NAT). Therefore, you must enter the public facing IP address of the on-premises Gateway device.

- **Customer Reachable Routes:** Enter (in CIDR format) the subnet for the on-premises database server.
- **Pre-shared Key:** Enter the pre-shared key (PSK).

  The value is masked as you type it. The key here must match the key entered on the on-premises VPN gateway device.
- **IKE ID:** If you leave this option blank, the public IP address of the cloud gateway is used.

  The public IP is set during provisioning process.
- **Phase 1 IKE Proposal Options:** Specify Phase 1 IKE options.

  Leaving this blank tells the Gateway to let all possible values to be permitted.
- **Phase 2 ESP Proposal Options:** Specify Phase 2 Encapsulating Security Payload (ESP) options.

  Leaving this blank tells the Gateway to let all possible values to be permitted.
- **Require Perfect Forward Secrecy:** This option is selected by default.

5. Click **Create.**

   A message appears showing that the VPN connection is added. Its status on the web console is displayed as **Pending/Provisioning**.

> **Note:**
>
> It takes anywhere from 20 minutes to 45 minutes, for a VPNaaS instance creation. During this process, the public IP address assigned to the VPNaaS gateway and its corresponding private IP address are displayed. To set up an on-premises VPN Gateway device, you must provide the public IP address. Make a note of this IP address. Refresh the page until the VPNaaS instance is ready.

Related Topics

- Setting Up a VPN Connection Using VPNaaS

Create and Configure On-Premises Third-Party VPN Device

This topic only applies to Data Integration Platform Cloud Classic.

You can use a third-party VPN device for an on-premises environment. The following example uses pfSense VPN appliance. This section covers the VPN configuration required for the VPN appliance, for connecting to Oracle Public Cloud's VPNaaS.

To configure pfSense VPN appliance Phase 1 settings:

1. Click the **Tunnels** tab, and edit the Phase 1 **General Information.**
2. Enter the **Authentication** and **Algorithms** details.
3. For Advanced Options, select the **Responder Only** and the **Dead Peer Detection** options.
4. Click **Save.**

To configure pfSense VPN appliance phase 2 settings:

1. Click the **Tunnels** tab, and edit the Phase 2 **General Information.**
2. Enter the **SA/Key Exchange** details.
3. Set the Advanced Configuration details, and click **Save.**

> **Note:**
>
> Make sure that the Shared Key for the on-premises VPN device and the Oracle Public Cloud's VPNaaS match.

OPC and On-Premises – Verify Tunnel Status is Up

This topic only applies to Data Integration Platform Cloud Classic.

After you set up an on-premises third-party VPN device and create a VPNaaS instance, then you must establish a tunnel session between the on-premises environment and OPC. You have a tunnel session if both ends have a running status.

To verify if the tunnel has a running status:

1. Navigate to the Compute Classic console, and click the **Network** tab.
2. From the left navigation pane, click **IP Networks.**
3. Click **VPN,** then under **VPNaaS**, click **VPN Connections**.
4. In the right pane, check if the Tunnel status is **Up** for VPNaaS on the OPC side.
5. To check the on-premises side, go to the pfSense VPN Status page, and see if the status is **Established.**

OPC and On-Premises – Verify VPN Connection

This topic only applies to Data Integration Platform Cloud Classic.

Before you create a Data Integration Platform Cloud Classic Connection to an on-premises database, you must verify if data can flow between the on-premises database server and the DIPC server.

To verify if there is a VPN connection, do one of the following:

- From the on-premises database, log in to the DIPC Server, by using its private IP address.
- From the DIPC server, log in to the on-premises database server. Then verify the connection from the Data Integration Platform Cloud Classic VM to the on-premises database server listener port by using the `netcat` command, `nc`.

Configure DIPC Connection to On-Premises Database Server as a Data Source

This topic only applies to Data Integration Platform Cloud Classic.

You can create a Connection within Data Integration Platform Cloud Classic to an on-premises database server by using the Data Integration Platform Cloud console.

To create a Connection:

1. From the Data Integration Platform Cloud home screen, click **Create Connection** and enter the **General Information.**
2. In the **Connection Settings** section, set the following:
   - **Hostname:** Enter the Host IP address of the on-premises database server.
   - **Port:** Enter the Database Listener Port. For Oracle databases, the default value is `1521`.
   - **Username** and **Password:** Enter the database user name and password.
   - **Service Name:** Enter the Database Listener Service Name.
   - **Schema Name:** Enter a schema that's set up for the database or use the default one.
   - **CDB Connection:**Leave blank.
3. Click **Test Connection** and **Save** if the connection is successful.

   After you save the Connection, it appears in the Catalog, in the **Connection** category.

4. Click that Connection and check the summary information or metadata/tables attached to that connection source.

The Connection that you create for an on-premises database can now be used by any job to perform data synchronization, integration, or validation.

Understand the Information on the Home page

The Home page provides you with quick access to Agents, Connections, Tasks, and Jobs, as well as a summary of active components in your Data Integration Platform Cloud instance.

At the top of the page, the menu icon displays the navigation pane. Here, you can access the Catalog, Monitor, Agents, Policies, and Admin pages. Click **Home** to return to the Home page at any time. When you click the **Notifications** icon, you’re brought to the Notifications page, where you can review a list of system messages and alerts. Click the **user icon** to access context-sensitive help for any page that you’re on, tutorials, videos, information about Data Integration Platform Cloud, or sign out of your session.

There are three sections to the Home page, the Welcome carousel, the Summary, and the Getting Started section.

Welcome

When you log in to Data Integration Platform Cloud, you're brought to the Home page. If it's your first time, click **Watch Video** to tour the interface. Afterwards, you can download an Agent, or create Connections and Tasks from the tiles in the carousel.

Summary

The Summary section displays recent and active Tasks, Jobs, Connections, and Agents in your Data Integration Platform Cloud instance. Click the title of the tile to see a filtered view of the Catalog, which displays a full list of Tasks, Jobs, or Connections, or the Agents page. Click the name of any component, Task, Jobs, Connections, or Agents, in a tile to access its page and view its metadata.

Getting Started

The Getting Started section enables you to download an Agent, or create new Tasks or other system components. Here's a list of what you can download or create:

- **Add Data to Data Lake**
  Store and organize large volumes of data from different sources in its raw format into the Oracle Data Lake for later use.

  For more information, see Add Data to Data Lake.
- **Agents**
  Download a DIPC Agent that orchestrates data integration tasks on your data source.

  For more information, see Set up an Agent.
- **Connection**
  Create connections to define the source and target data sources.

  For more information, see Create a Connection.
- **Data Lake**

Create a repository on the Oracle Cloud to store a vast amount of data from different sources in its raw data.

For more information, see Create a Data Lake.

- **Data Preparation**
  Harvest data from a data source, perform a variety of transformations to organize and cleanse your data, and then write the resultant data to a new data entity.

  For more information, see Prepare Data.
- **Execution Environment**
  Use Spark Execution on Big Data Cloud or YARN to run the Add Data to Data Lake task.

  For more information, see Set Up an Execution Environment.
- **ODI Execution**
  Invoke an existing Oracle Data Integrator (ODI) Studio Scenario to perform bulk data transformations.

  For more information, see ODI Execution.
- **Policy**
  Create policies to stay informed about your job activities and to schedule jobs. Set conditions by using predefined, job-related metrics. You'll get notifications when the conditions are met.

  For more information, see What is a Policy?
- **Replicate Data**
  Capture new transactions in your source data source and replicate them to a target data source.

  For more information, see Replicate Data.
- **Synchronize Data**
  Copy data from a source data source to target data source, and then keep both data sources in sync. Use filter rules to include or exclude specific data entities in your job. The synchronize data task captures any change in the source schema and replicates it in the target and vice versa.

  For more information, see Synchronize Data.

3 Before You Perform Tasks in Data Integration Platform Cloud

To perform tasks in Data Integration Platform Cloud, you must first download, install, and configure your agent, and then create Connections. Agents enable the Data Integration Platform Cloud instance to connect with your data sources. Agents also orchestrate the tasks that you run on your data sources from the Data Integration Platform Cloud console. Connections enable you to define source, staging, or target data sources for your tasks.

Topics:

- Set up an Agent
- View and Edit Connections
- Set up Default Connections
- Understand the Catalog

Set up an Agent

Agents orchestrate data exchange from your data sources to Oracle Data Integration Platform Cloud.

> **Note:**
>
> Assuming that you have access to the file system on ExaCC, you can follow the instructions below on installing the remote agent of Data Integration Platform Cloud.
>
> The DIPC agent running on ExaCC needs a connection to the DIPC host, which runs either in Oracle Public Cloud (OPC) or own Cloud console (OCC).

Make sure that you have an IDCS account before you register the remote agent. DIPC does not support federated Single Sign-On (SSO).

Topics

- What is an Agent?
- Agent Certifications
- Import a Client Certificate to the Agent
- Download the Agent Package
- Add Your DIPC Agent as a Confidential Application

What is an Agent?

The DIPC agent connects the DIPC host to the on-premises and cloud data sources. The DIPC agents also orchestrate the jobs that you run on the data sources from the DIPC console.

You can download the DIPC agent from the DIPC console and install it on any machine, on-premises, or cloud that has access to your data sources. For example, you can install the DIPC agent on premises, or on cloud, such as Oracle Cloud VMs or Third-Party Cloud VMs.

DIPC Classic also provides the DIPC host agent. The DIPC host agent is a pre-configured ready-to-use agent that is located on the DIPC host. This agent works in the same manner as the DIPC agent, but only with the cloud data sources and components. If you want to use on-premises data sources or components, you must either download and use the DIPC agent, or set up a VPN connection so that the DIPC host agent can access the on-premises data sources.

For all the certified data sources that you connect to, you must download and run an agent with a component for that data source, so that the agent can communicate with the Data Integration Platform Cloud server. For example, for Oracle 12c, you can either download the Linux or Windows version of the Oracle 12c component, and then set up the agent.properties file with connection information, register it and run it.

An agent:

- Exchanges heartbeats with the server and reacts based on server availability
- Captures server health records of the host process and sends them to the server for monitoring
- Reduces installation requirements
- Makes maintenance and patching easier
- Reduces the number of processes running on the Virtual Machine (VM)

Agent Certifications

For all the tasks that you create in Data Integration Platform Cloud, you assign an agent. Agents must run only on their certified platforms.

Available Components

The Components comprise a set of binary files and agent properties file that you need to set up to run the DIPC tasks. Based on the DIPC tasks you want to run, you need to select the appropriate components, when you download the DIPC agent using the DIPC console. The components that you select are included in the agent package.You can select from the following components when you download the agent:

- Linux

  - Big Data (OGG)
  - Data Integrator (ODI)
  - Data Preparation
  - Data Lake
  - Oracle 11g (OGG)
  - Oracle 12c (OGG)
- Windows
  - Big Data (OGG) (This component is not certified with any of the tasks. You can use it to perform data integration through the hosted VMs.)
  - Oracle 12c (OGG)
  - SQL Server (OGG)
- All data sources must have x86_64, the 64 bit version of x86 operating systems, with the latest upgrade.
- Run your remote agents only on operating systems that are certified for the agents.
- Set up the remote or host agents for data sources certified for your task.

Operating Systems Certified for DIPC Agents

The following tables list the operating systems your agents can run on. If your certified data sources have the same version of operating system, then you can run the DIPC agent on the same machine as the data source. Otherwise, run the agents on their certified operating systems and ensure that they have access to your data sources. For example, if your Oracle 11g database is on a Windows machine, then you can't run the Oracle 11g component that you downloaded with the agent on the same machine as there is no Windows component for Oracle 11g. Instead, you must either download a DIPC agent's 11g component on a Linux machine or use the DIPC host agent. Then run your agent remotely with a setup that has access to the database. Alternatively, you can have a DIPC agent on Windows with the 12c component, connecting to an Oracle Database Cloud Service 12c Classic.

Agent Component	OEL	RHEL	SLES	Windows	Certified Data Source(s)	Applicable Task(s)
Oracle 12*c* (OGG)	6.x, 7.x	6.x, 7.x	11, 12	2012, 2016	Oracle Database Cloud Classic 12*c* and Oracle Database 12*c*	All tasks
Oracle 11*g* (OGG)	6.x, 7.x	6.x, 7.x	11, 12	no	Oracle Database Cloud Classic 1*g* and Oracle Database 11*g*	All tasks
Big Data (OGG)	6.x, 7.x	6.x, 7.x	11, 12	no	Kafka Connect	Replicate Data

Agent Component	OEL	RHEL	SLES	Windows	Certified Data Source(s)	Applicable Task(s)
Data Lake	6.x, 7.x	6.x, 7.x	11, 12	no	Oracle Database Cloud Classic 12*c* and Oracle Database 12*c*, Oracle Database Cloud Classic 1*g* and Oracle Database 11*g*, Object Storage Classic, Flat Files	Add Data to Data Lake
Data Integrator (ODI)	6.x, 7.x	6.x, 7.x	11, 12	no	Oracle Database Cloud Classic 12*c* and Oracle Database 12*c*, Oracle Database Cloud Classic 1*g* and Oracle Database 11*g*, SQL Server, MySQL, Flat Files	ODI Execution, Synchronize Data
Data Preparation	6.x, 7.x	6.x, 7.x	11, 12	no	Oracle Database Cloud Classic 12*c* and Oracle Database 12*c*, Oracle Database Cloud Classic 1*g* and Oracle Database 11*g*, Flat Files	Data Preparation

Tasks Certified for Host Agents

The host agent available only on the Data Integration Platform Cloud Classic server is on a Linux platform, with OEL 7.x operating system and the latest patch installed.

Agent Component	Certified Data Source	Applicable Task(s)	GoldenGate Version to Use
Oracle 12*c* (OGG)	Oracle Database Cloud Classic 12*c*, Oracle Database 12*c*, Autonomous Data Warehouse Cloud (ADWC-Supported only for Replicate Data)	All tasks	12.3.0.1.4
Oracle 11*g* (OGG)	Oracle Database Cloud Classic 1*g* and Oracle Database 11*g*	All tasks	12.3.0.1.4
Big Data (OGG)	Kafka Connect	Replicate Data	12.3.2.1.0
MySQL (OGG)	MySQL	ODI Execution	12.2.0.1.1

If you want to use MySQL for the ODI Execution task, then you must modify the agent.properties file to use GoldenGate 12.2. The default version is GoldenGate 12.3.

Import a Client Certificate to the Agent

The Data Integration Platform Cloud agent and server communicates via the Transport Layer Security (TLS) protocol. If the remote agent has to trust the Data Integration Platform Cloud server, the public key (certificate) used by Data Integration Platform Cloud server should be recognized/trusted by the on-premises agent client.

To import a client certificate:
The Data Integration Platform Cloud client side default certificate store (trust store) already contains the root certificate of the accredited certifying authority, so the client can verify and trust the signed certificate. Hence, it’s not mandatory to import a client certificate, before starting the agent. Certificate import is required, if any error occurs. Otherwise, it’s optional to import a certificate to the Agent’s JAVA_HOME.

If the following error occurs, you should manually import the Data Integration Platform Cloud server's certificate to the on-premises agent client side JRE trust store. This error occurs only when the client side default Java certificate store fails to verify or recognize the server's certificate.

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: No
trusted certificate found
```

1. Log in to Data Integration Platform Cloud.
2. In the address bar of your web browser, click **View/Show site information** (the security padlock icon).
   - If you’re using Chrome, select **Certificate** and then in the **Details** tab of the Certificate dialog, select **Subject** and click **Copy to File**. Follow the prompts in the Certificate Export Wizard to save the certificate.
   - If you’re using Firefox, click **Show connection details** (the right arrow next to the URL), and then click **More information**. In the Page Info dialog, click **View Certificate,** and then click **Export** in the **Details** tab. Save the certificate.
3. Navigate to `$JAVA_HOME`, using the keytool present under `$JAVA_HOME/bin/keytool`

4. Do one of the following
   - Import the client side certificate into the cacerts present under `$JAVA_HOME`
     ```
     $JAVA_HOME/bin/keytool -import -alias ocptest -keystore /
     scratch/<user>/JDK/jdk1.8.0_171/jre/lib/security/cacerts -
     file /scratch/ocp_test.crt
     ```
     In general, keystore can be either private or Java default keystore.
   - Go to the on-premises agent hosting local machine and run the following command:

     For Data Integration Platform Cloud Classic:
     ```
     keytool -import -alias dipc_cert -keystore  $JAVA_HOME/jre/lib/
     security/cacerts -file dipc_cert.crt
     ```
     For Data Integration Platform Cloud:
     ```
     # ./keytool -import -v -alias adipc_cert -file
     adipcocporaclecloudcom.der -keystore <KEYSTORE>
     ```

     > **Note:**
     >
     > If KEYSTORE is `cacerts`, then pass parameter `-storepass changeit`. If KEYSTORE is `DemoTrust.jks`, then pass parameter `-storepass DemoTrustKeyStorePassPhrase` in the command above. For example:
     >
     > ```
     > # ./keytool -import -alias adipcocporaclecloudcom -
     > keystore /u01/app/12.2.0.1/grid/jdk/jre/lib/security/
     > cacerts -file /home/opc/adipcocporaclecloudcom.crt
     > ```

5. Enter `keystore password`.

   The default password is:

   `changeit`

   You can now list down the certificates from the local machine trust store to see the imported certificate:
   ```
    echo 'changeit' | keytool -list -v -keystore $(find $JAVA_HOME -name
   cacerts) | grep 'Owner:'  > out
   ```

Download the Agent Package

You can download the Agent package from the DIPC console. Alternatively, you can download the Agent from DIPC URL using the cURL (Client for URLs) call.

To download the Agent Package:

1. Log in to Data Integration Platform Cloud.

2. Click **Download** in the Agents tile on the Home page, or click **Download Installer** on the Agents page.
3. Select the components you want to include in your Agent package, and then click **Download**.

   DIPC supports the following Agent Components:

Agent Component	Supported Platform	Description
Oracle 12c (OGG)	Windows Linux	Provides Oracle GoldenGate 12c connector for Synchronize Data and Replicate Data tasks.
Oracle 11g (OGG)	Linux	Provides Oracle GoldenGate 11g connector for Synchronize Data and Replicate Data tasks.
Data Integrator (ODI)	Linux	Enables you to connect to Oracle Data Integrator, and perform data integration tasks. Required for Synchronize Data with Initial Load Advanced Option.
Data Lake	Linux	Enables you to connect to Data Lake, where you can store vast amounts of data for later use.
Data Preparation	Windows Linux	Enables you to perform Data Preparation transforms such as Ingest, Sample, Profile.
Big Data (OGG)	Windows Linux	Provides Oracle GoldenGate Big Data connector for Replicate Data task.
SQL Server (OGG)	Windows	Provides Oracle GoldenGate SQL Server connector for Synchronize Data and Replicate Data tasks.

Before You Download an Agent Using cURL Command

Make sure that you have agent configuration command handy before you proceed.

The agent configuration command is as follows:

```
dicloudConfigureAgent.sh -dipchost=<DIPCHOSTNAME> -dipcport=<DIPCPORT>
      -idcsServerUrl=<IDCSERVERURL> -agentIdcsScope=<IDCSSCOPE> -
user=<USERNAME>
      -password=<PASSWD> -agentClientId=<CLIENTID>
    -agentClientSecret=<CLIENTSECRET>
```

1. Go to `base64encode.org`, and encode the following value to `base64` (note the colon between the two strings):

   ```
   <agentClientId>:<agentClientSecret>
   ```

2. Get the `Authorization header` details as follows:

   ```
   curl -X POST<idcs-url>/oauth2/v1/token -H 'authorization:
   Basic<base64encodedstringfrom
   ```

```
step1>' -H 'content-type: application/x-www-form-
urlencoded;charset=UTF-8' -d
'grant_type=client_credentials&scope=<agentIdcsScope>'
```

This will generate Bearer token required for the agent download. For example:

```
curl -X POST https://<idcs-url>/oauth2/v1/token -H
'authorization: Basic
MmFkNDFhMGZhZmYyNGYyYWJhOWUxMTcxYjg3Y2U0OTc6ZmJmN2MwNzItZGI5Zi00MzY2LThl
NjIt
ZTMxNjg4NTgzNWVlCg==' -H 'content-type: application/x-www-form-
urlencoded;charset=UTF-8' -
d
'grant_type=client_credentials&scope=<agentIdcsScope>'
```

3. Go to postman, and add the following details:

   Sample URI:

   ```
   https://adipc1913tagint22inst-adipc2bppreprod.adipc.ocp.oc-
   test.com/dicloud/agentdeployment/v1/packages/current/configurationforms
   ```

   Request: `POST`

   Output of authorization token from Step 2 for postman request.

   Headers:

   ```
   Content-Type:application/x-www-form-urlencoded
   Accept:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
   Accept-Encoding:gzip, deflate
   Connection:keep-alive
   Authorization:Bearer <output of step 2>
   ```

   Request Body:

   `x-www.form-urlencoded`

   For example,

   ```
   cfgJson:{"operatingSystem":"linux.64.bit","connectors":["ODI"]}
   ```

4. Go to **Code** on the right-hand side.
5. Click the dropdown on the left-hand side, and select **cURL.**
6. Copy the content to a text editor, and trigger this command on your host. At the end of the command, enter the name of the file where the output should be saved .
7. Run the command in the Linux location where you want to download the agent.

   Sample command:

   ```
   curl -X POST \
   https://adipc1913tagint22inst-adipc2bppreprod.adipc.ocp.oc-
   test.com/dicloud/agentdeployment/v1/packages/current/configurationforms
   \
   ```

```
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8' \
-H 'Accept-Encoding: gzip, deflate' \
-H 'Authorization: Bearer
eyJ4NXQjUzI1NiI6InZpSmVfeTFGdUJJalhiYXpFOXg2ZUFZa1lxYXZseFM3bm41SWlDVENw
Q0kiLCJ4NX
QiOiI1b1ZISVhwNlpuSUJJUW8tZ0w5Mkx1WV9ES00iLCJraWQiOiJTSUdOSU5HX0tFWSIsIm
FsZyI6IlJTM
jU2In0.eyJzdWIiOiIyYWQ0MWEwZmFmZjI0ZjJhYmE5ZTExNzFiODdjZTQ5NyIsInVzZXIud
GVuYW50Lm5
hbWUiOiJpZGNzLTI1ZWYwZDRkODY4MTRlNDY5YzA4MTc1MGZkZTMxZDY2Iiwic3ViX21hcHB
pbmdh
dHRyIjoidXNlck5hbWUiLCJwcmltVGVuYW50Ijp0cnVlLCJpc3MiOiJodHRwczpcL1wvaWRl
bnRpdHkub3J
hY2xlY2xvdWQuY29tXC8iLCJ0b2tfdHlwZSI6IkFUIiwiY2xpZW50X2lkIjoiMmFkNDFhMGZ
hZmYyNGYyY
WJhOWUxMTcxYjg3Y2U0OTciLCJjYV9ndWlkIjoiY2FjY3QtNGEwNTkzM2Y3YTJkNDM0Y2JiM
DcyYzc3Yz
RiMTA3ZDciLCJhdWQiOlsiaHR0cHM6XC9cLzVFRDRBNzc5ODhFRTRBOEVCRTVBMzFBMzNFND
MyRT
hBLmFkaXBjLm9jcC5vYy10ZXN0LmNvbTo0NDMiLCJ1cm46b3BjOmxiYWFzOmxvZ2ljYWxndW
lkPTVFR
DRBNzc5ODhFRTRBOEVCRTVBMzFBMzNFNDMyRThBIl0sInN1Yl90eXBlIjoiY2xpZW50Iiwic
2NvcGUiOiJ
1cm46b3BjOnJlc291cmNlOmNvbnN1bWVyOjphbGwiLCJjbGllbnRfdGVuYW50bmFtZSI6Iml
kY3MtMjVl
ZjBkNGQ4NjgxNGU0NjljMDgxNzUwZmRlMzFkNjYiLCJleHAiOjE1NTg5NzI1MTksImlhdCI6
MTU1ODk2
ODkxOSwidGVuYW50X2lzcyI6Imh0dHBzOlwvXC9pZGNzLTI1ZWYwZDRkODY4MTRlNDY5YzA4
MTc1
MGZkZTMxZDY2LmlkZW50aXR5LnByZXByb2Qub3JhY2xlY2xvdWQuY29tIiwiY2xpZW50X2d1
aWQiOiJk
NjQ1MDkwYmQ3NjU0N2QwOTgyOWM1Nzk0NDI0M2YzZiIsImNsaWVudF9uYW1lIjoidGVzdHBy
ZXBv
ZCIsInRlbmFudCI6ImlkY3MtMjVlZjBkNGQ4NjgxNGU0NjljMDgxNzUwZmRlMzFkNjYiLCJq
dGkiOiJlMzY
1MjYzZC1jNjk4LTQ2NGItYWJmMS0xZDc4ZDU1ZTkzZjgifQ.aX9KRs67v4K7ZMqotk0JvkGs
ZPUadu8M8
hDYCywMa-97k2QvO7_pH6BPnpQcngGycmGuirezPiZEKpz67588Rhh-j16masq-
avzlON5j5v_psglxIh3GNTw0PxfGin1506AkSe1pws4k2yU5EIkjo-
FzuY0rS2XkZ2nstpuZQpIfX-8ltydD--
mrlsTbP1ZKrN-
Bu2q5KkzsPOJD50NFz6onqwc_Cz4fhz2c1qLX8NBWzyh8cHjzwSbssTU2GJBiUyPcV5VJicL
zXoOp9uxks
wq9wrF0TJvII1VUdC29wFNEpbXQsdCGaWMuIEnnF51rZGYbMPVsybLCV9Am35_6YA' \
-H 'Cache-Control: no-cache' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'Postman-Token: b63ecdf0-60b0-499e-bba2-193aa83a68ea' \
-d
cfgJson=%7B%22operatingSystem%22%3A%22linux.64.bit%22%2C%22connectors
%22%3A%5B%22
ODI%22%5D%7D \
--output test.zip
```

Add Your IDCS OAUTH2 Token to Download an Agent Using REST cURL Call

To use cURL to access the REST APIs, authorization headers must be sent.

```
-H "Authorization: Bearer <TOKEN_FROM_PREVIOUS_STEP>"
```

`TOKEN_FROM_PREVIOUS_STEP` can be set up as follows:

1. From the **Identity Cloud Service** console, get the **Client ID** and **Client Secret**. See Add Your DIPC Agent as a Confidential Application.
2. Add the **Scope** URL for the Agent application added as a **Confidential Application**. Note down this **Scope** URL for use in later steps.
3. Get a `Base64` encoded string of the **Client ID** and **Client Secret**. For example, on the command line execute:

   ```
   echo "<CLIENT_ID>:<CLIENT_SECRET>" | base64
   ```

4. Get the `Authorization Token` using the above `Base64` output:

   ```
   curl -X POST https://<idcs-url>/oauth2/v1/token -H 'authorization: Basic
         <BASE64_OUTPUT>' -H 'content-type: application/x-www-form-
   urlencoded;charset=UTF-8' -d
         'grant_type=client_credentials&scope=<SCOPE_URL>'
   ```

5. Use the `Authorization Token`:

   ```
   curl -v -X GET <REST_ENDPOINT_URL> -H "Authorization: Bearer
         <TOKEN_FROM_PREVIOUS_STEP>"
   ```

   You can also test if a REST URL for GET data is correct or not by entering it directly in a web browser, and then typing the username and password when the web page prompts for login information. Once the REST URL is verified to be correct, it can then be used in a cURL command along with the authorization header info as described above.

API to Download Agent through REST/cURL Call

You can download the Agent from the DIPC URL, using the following cURL call.

```
curl -X POST
http://<host:port>/dicloud/agentdeployment/v1/packages/current/
configurationforms
-H "Authorization: Bearer <TOKEN_FROM_PREVIOUS_STEP>" \
-F
    'cfgJson={"operatingSystem":"linux.64.bit","connectors":
["GGMON_BIGDATA"]}'
```

HTTP

```
POST /dicloud/agentdeployment/v1/packages/current/configurationforms
HTTP/1.1
```

```
Host: <host:port>

Content-Type: multipart/form-data;

Content-Disposition: form-data; name="cfgJson"

Content-Disposition: form-data; name="cfgJson"

{"operatingSystem":"linux.64.bit","connectors":["GGMON_BIGDATA"]}
```

Possible values for `operatingSystem` are :

- `Linux.64.bit`
- `Windows.64.bit`

Connectors are representative of the check boxes that appear on the DIPC download UI page. So possible values are:

For Linux:

- GGMON_ORACLE_12c
- GGMON_ORACLE_11g
- GGMON_BIGDATA
- DATALAKE
- DATAPREPARATION
- ODI

For Windows:

- GGMON_ORACLE_12c
- GGMON_SQLSERVER

Connectors:

"GGMON_ORACLE_12c":"GoldenGate for Oracle 12c"

"GGMON_ORACLE_11g":"GoldenGate for Oracle 11g"

"GGMON_SQLSERVER":"GoldenGate for SQL Server"

"GGMON_BIGDATA":"GoldenGate for Big Data"

"ODI":"Oracle Data Integrator"

"DATALAKE":"DataLake"

"DATAPREPARATION":"DataPreparation"

Before You Register Your Agent

Adding your agent as a confidential application enables you to get certain parameters (`agentIdcsScope`, `agentClientId`, `agentClientSecret`) required to register your agent. However, you must perform some additional steps to obtain the values of the other parameters required for the registration.

Make sure that you have an IDCS account before you register the remote agent. DIPC does not support federated Single Sign-On (SSO).

You need the values of the following parameters to register your agent, and connect to DIPC in OAuth mode:

- agentIdcsScope
- agentClientId
- agentClientSecret
- idcsServerUrl
- dipchost
- dipcport
- username
- password

To get agentIdcsScope, agentClientId, and agentClientSecret parameter values, see Add Your DIPC Agent as a Confidential Application.

Obtain the idcsServerUrl

1. In the navigation menu, go to **Identity**, and then select **Federation**.
2. Copy the value for **Oracle Identity Cloud Service Console**. This is your idcsServerUrl.

Obtain the value of the idcs_Service_ID

You can derive the idcs_Service_ID from the idcsServiceUrl. The idcs_Service_ID is the first part of the service URL. For example: https://<idcs_Service_ID>.identity.oraclecloud.com/ui/v1/adminconsole.

Obtain the value of dipchost

Get the value of dipchost from the address bar of the DIPC console.

The dipcport value

Use the default dipcport 443. If your on-premises system doesn't allow any outgoing internet traffic, you must make an exception rule for port 443 of the DIPC host machine.

Add Your DIPC Agent as a Confidential Application

DIPC application server is a protected server. You must add your DIPC Agent as a confidential application to get the values for the parameters required to register your Agent. Adding the DIPC Agent as a confidential application ensures secure interaction between the DIPC Agent and the DIPC application server. After you add the DIPC Agent as a confidential application, you can connect it to the DIPC application server in OAuth mode.

When you add the DIPC Agent as a confidential application, you get the values of the agentIdcsScope, agentClientId, and agentClientSecret parameters. Make sure that you note down the values of these parameters, as they are required to register the DIPC agent. Also, take note of your Data Integration Platform Cloud instance.

Add DIPC Agent as a Confidential Application for DIPC Classic

DIPC Agent must be added as a confidential application to establish secure interaction between the DIPC agent and the DIPC application server. After you add the DIPC agent as a confidential application, you can connect it to the DIPC application server in OAuth mode.

To add DIPC Agent as a confidential application for DIPC Classic:

1. Log in to Oracle Cloud as the application administrator, or if you have your IDCS Service URL, you can log in directly to the IDCS Admin console and continue with Step 4.
2. Open the navigation menu in the upper left corner, select **Platform Services**, and then **Identity Cloud**.
3. Click **Open Service Console**.
4. From the navigation menu, select **Applications**.
5. Click **Add,** then select **Confidential Application.**
6. On the Add Confidential Application **Details** page, enter a **Name** in the **App Details,** and then click **Next.**
7. On the **Client** page, select **Configure this application as a client now**.
8. Under the **Authorization** section, select **Resource Owner**, **Client Credentials**, and **Refresh Token** for **Allowed Grant Types**.
9. Under **Token Issuance Policy**, go to the **Resources** section, and click **Add Scope** for **Resources**.
10. In the **Select Scope** dialog, select your Data Integration Platform Cloud Classic application (for example, DIPCAUTO_adipc-numeric-Agentdipcauto), and then click the corresponding right-arrow button.
11. Select the listed scope name, and then click **Add.**

    The Application you added appears under the Allowed Scope. Take note of the `agentIdcsScope`. This value needs to be copied to the command prompt, while registering your agent.
12. Click **Next** until you reach the **Authorization** page, and then click **Finish.**

    The Application Added confirmation message appears, displaying the Client ID and Client Secret to configure your remote Agent.

    Take note of the Client ID and Client Secret before closing this message.

    If you need to regenerate the Client ID and Client Secret again later, return to the Identity Console Applications page, select your application, and under the Configuration tab, click Show Secret and Regenerate.
13. Click **Activate.**
14. In the Activate Application dialog, click **Activate Application**.

Add DIPC Agent as a Confidential Application for DIPC

DIPC does not provide an Agent on its host. You can deploy an agent to any type of infrastructure where you need to execute jobs, either on-premises, Database Cloud Service, Oracle Cloud Infrastructure as a Service, and so on.

To add DIPC Agent as a Confidential Application for DIPC

1. Log in to Oracle Cloud as the application administrator, or if you have your IDCS Service URL, you can log in directly to the IDCS Admin console and continue with Step 4.
2. Open the navigation menu in the upper left corner, select **Platform Services**, and then **Identity Cloud**.
3. Click **Open Service Console**.
4. From the navigation menu, select **Applications.**
5. Click **Add,** then select **Confidential Application.**
6. On the Add Confidential Application **Details** page, enter a **Name** and **Description** in the **App Details** section.
7. Leave the other fields as is, and then click **Next.**
8. On the **Client** page, select **Configure this application as a client now**.
9. Under the **Authorization** section, select **Resource Owner**, **Client Credentials**, **JWT Assertion**, and **Refresh Token** for **Allowed Grant Types**.
10. Under **Accessing APIs from Other Applications**, for **Trust Scope**, select **Allowed Scopes**.
11. Under **Allowed Scopes**, click **Add**.
12. In the **Add Scope** dialog, select the DIPC instance (for example, DIPCINST_ADIPCInstanceName), and then click the corresponding right-arrow button.
13. In the dialog that opens, select the listed scope name, click the **Back** button, and then click **Add.**

    The Application you added appears under the Allowed Scope. Note down the `agentIdcsScope`. This value needs to be copied to the command prompt, while registering your agent.
14. Click **Next** until you reach the **Authorization** page, and then click **Finish.**

    The Application Added confirmation message appears, displaying the Client ID and Client Secret to configure your remote Agent.

    Note down the `Client ID` and `Client Secret` before closing this message.

    Now, you've all the parameters ready to go to the agent instance location and register your agent. See Register Your Agent.

    If you need to regenerate the Client ID and Client Secret again later, return to the Identity Console Applications page, select your application, and under the Configuration tab, click Show Secret and Regenerate.
15. Click **Activate.**
16. In the Activate Application dialog, click **Activate Application**.

Register Your Agent

After downloading the agent package, you must unzip and run the register script to register it with Data Integration Platform Cloud.

Make sure that the time you set on the machine where the DIPC agent resides and the DIPC server are the same and correct/current time. If the time on both the machines is not in sync, the DIPC server doesn’t process the messages send by the DIPC agent, and shows the following error:

(DIPC-AGTMED-0009 : Agent Mediator operation timed out at QUEUED status before reaching SUCCESS/WARNING/ERROR status.

To register your agent with Data Integration Platform Cloud:

1. Create a directory called `dicloud`, move the agent package to that directory and then unzip the agent package.
2. After the files are unzipped, navigate the `dicloud` directory, and then set the parameters in the agent registration script command:

   ```
   ./dicloudConfigureAgent.sh <agentInstanceDirectory> -recreate -debug -
   dipchost=<dipc.example.host.com> -dipcport=<port> -user=<diuser> -
   password=<dipassword> -authType=<BASIC/OAUTH2> -idcsServerUrl=<idcs
   server url> -agentIdcsScope=<agent IDCS Client Scope> -
   agentClientId=<Agent IDCS clientID> -agentClientSecret=<Agent IDCS
   clientSecret>
   ```

   The parameters for the registration script command are as follows:

   > **Note:**
   >
   > If a value is not provided for the parameters for which you don't have defaults, you will be prompted to enter it in the command line . For example , "user" , "dipcPort", "dipcHost", "idcsServerURL," and so on are critical parameters. If you provide some defaults to these, it will not work.

Parameter	Description
`<agentInstanceDirectory>`	Indicates the name of the agent instance directory. This is optional. All Agent instances are always created in : $ `{agent_unzip_loc}/dicloud/agent` The default value is `dipcagent001`.
`-user=<diuser>`	You must provide the username with which the agent will connect to the Data Integration Platform Cloud host/IP. This is mandatory. For example: `-user=dipcOperator`

Parameter	Description
`-password=<dipassword>`	You must provide the password with which the agent will connect to Data Integration Platform Cloud host/IP. `-password=password`
`-recreate`	Specifies that you're recreating the agent. This is optional. The default value is `false`.
`-dipcport=<port>`	You must provide the port to which this agent will connect and register. This is mandatory. For example: `-dipcport=443`

`-dipchost=<dipc.example.host.com>`	You must provide the Data Integration Platform Cloud host/IP to which this agent will connect and register.
	This is mandatory.
	Note: Do not prefix with https://
	For example:
	`-dipchost=dipc.example.host.com`
	For Data Integration Platform Cloud, it looks like this:
	`dipcinstancename-dispaas.dipc.ocp.examplecloud.com`

`-debug`	Specifies that you're running this script in debug mode and will also generate a debug log file. This is optional. The default value is `false`.
`-authType=<OAUTH2>`	This property determines the authentication mode with the server. It is OAUTH2. This is optional. Default value is OAUTH2. • OAUTH2: uses Identity Cloud Service (IDCS) server to perform OAUTH2 protocol-based authentication for agent requests. For example: `-authType=OAUTH2`

`-idcsServerUrl=<idcs_server_url>`	Required if `-authType=OAUTH2`. This is mandatory.
	When `authType=OAUTH2`, provide the full URL of the IDCS server that the agent will connect to for authentication tokens.
	For Data Integration Platform Cloud, it looks like this:
	`-idcsServerUrl=https://idcs.example.oracle.com`
	For Data Integration Platform Cloud, it looks like this:
	`-idcsServerUrl=https://idcs-alphanumericDigits.identity.examplecloud.com`

Parameter	Description

`-agentIdcsScope=<client_scope:443external>`	Required if `-authType=OAUTH2`. This is mandatory.
	When `authType=OAUTH2`, provide the Client Scope of this agent obtained from the IDCS server.
	For Data Integration Platform Cloud, it looks like this:
	`-agentIdcsScope=https://idcs.example.identity.com:443external`
	See Add DIPC Agent as a Confidential Application for DIPC Classic
	For Data Integration Platform Cloud, it looks like this:
	`agentIdcsScope=https://alphanumericdigits.adipc.ocp.examplecloud.com:443urn:opc:resource:Consumer::all`
	See Add DIPC Agent as a Confidential Application for DIPC

`-agentClientId=<client_ID>`	Required if `-authType=OAUTH2`. This is mandatory. When `authType=OAUTH2`, provide the IDCS Client ID of this agent, obtained from the IDCS admin console. This information is stored in the agent wallet. For example: `-agentClientId=alphanumeric-digits`
`-agentClientSecret=<client_secret>`	Required if -authType=OAUTH2. This is mandatory. When authType=OAUTH2, provide the IDCS Client Secret of this agent, obtained from the IDCS admin console. This information is stored in the agent wallet. For example: `-agentClientSecret=Alphanumeric-digits`
`—odiRemote`	Use `–odiRemote` to set up your DIPC remote agent with the ODI plugin for Synchronize Data with Initial Load and ODI Execution Tasks. Agent will be registered with ODI, only if the `–odiRemote` parameter is passed. This parameter is optional if you are not using ODI. See also, Set up a Remote Agent for ODI.

3. Before executing the script, make sure to:
   - Install JDK.
   - Run the right version of Java (1.8.0_91).
   - Set $Java_Home (`-/Public/jdk/bin/java`)

     This is important, because when you create the confidential application, it will create certain things in your security store of your JDK environment.
4. Execute the agent registration script `./dicloudConfigureAgent.sh` to register your agent.

   When the script is run, it reads all the parameters you set in that file. You can also see your JDK Trust Store `/home/oracle/public/jdc/jre/lib/security/`

`cacerts`. If you've not set up your `$Java_Home` correctly, it will show an error here.

5. Go to the agents directory by running the script `$ cd agent`, and then run `$ cd agent directory name`.

   You can see a bin directory, and a conf directory. And inside the conf directory, you've the agent properties file. Proceed with Set Your Agent Properties.

You'll find the registration script log file, `dicloudRegisterAgent.log`, in the same directory where the script is run.

Set Your Agent Properties

After you've registered your Agent with Data Integration Platform Cloud, you can set your agent properties.

The `agent.properties` file for each agent created is located in the `conf` directory. For example, `${agent_unzip_loc}/dicloud/agent/<agentInstanceName>/conf/agent.properties`. This file enables you to perform advanced configuration of the agent. It contains the configuration properties, their default values, and a detailed explanation of each property. If it's a clean agent, and GoldenGate wasn't running before, you can use the default values, without any changes. If you want to run multiple GoldenGate instances or agents, make sure that all the ports are pointing to different agents.

`If required, you can set proxy details in agent.properties to connect to Object Storage Connection.`

- Go to dicloud/agent/dipcagent/conf/agent.properties.
- Add proxy details in this format:
  - `agentUseProxy=`
  - `proxyHost=`
  - `proxyPort=`

The Agent ports should be configured as unique port of the system, especially when there are more than one agent on the same operating system.

Make sure to set gghome in the path to access ggsci prompt:

`PATH=%12CGGHOME%;%12CGGHOME%\lib12;%12CGGHOME%\crypto;%PATH%`

For Windows , you must set it as `PATH=%12CGGHOME%;%12CGGHOME%\crypto;%PATH%`

Here's an example:

```
# agentPort
#   : This is the port at which the DIPC agent will be running.
#     Default Value : 7005
agentPort=7005

# agentMessageProcessorConnectionPoolSize
#   : This is the number of threads in the message processor thread pool
#     that this agent uses to handle messages sent from server.
#     Default Value : 5
```

```
#agentMessageProcessorConnectionPoolSize=5

# agentHeartBeatInterval
#   : This is the interval in seconds at which
#     heart-beat is sent to the server
#     Default Value : 10

#agentHeartBeatInterval=10

# ggInstanceHost
#   : HostName of the machine where Data Integration Platform is running.
#     Default Value : gghost.example.com
# ggInstancePort
#   : Manager Port of the Data Integration Platform instance that
#     you want to connect
#     Default Value : 7809
# ggCoreVersion
#   : Version of the Data Integration Platform instance that you are 
connecting.
#     The following versions with the values as below are
#     supported
#     1> V_12_2_0
#     2> V_12_1_2
#     Default Value : V_12_2_0
ggInstanceHost=localhost
ggInstancePort=7809
ggCoreVersion=V_12_2_0
# ggccServiceHost
#   : HostName of the machine where Data Integration Platform Cloud
#     Control/Service is running.
#     Default Value : ggcchost.example.com
# ggccServicePort
#   : Port number where Data Integration Platform Cloud Control Service is 
running.
#     If SSL is being used then set this to the SSL port of the server.
#     Default Value : 8001
ggccServiceHost=ggcchost.example.com
ggccServicePort=8001

# ggccServerConnectTimeout
#   : Connection Timeout to DIPC Server in milliseconds.
#     Default : 60000 milliseconds = 1 minute.
# ggccServerReadTimeout
#   : Read Timeout to DIPC Server in milliseconds.
#     Default : 60000 milliseconds = 1 minute.
# ggccServerConnectPoolSize
#   : Connection Pool Size to DIPC server.
#     Default : 5

#ggccServerConnectTimeout=120000
#ggccServerReadTimeout=120000
#ggccServerConnectPoolSize=5

# agentUseProxy
#   : Set this to true if Proxy is to be used for this agent.
#     Default Value : false
# proxyHost
#   : HostName where Proxy Server is configured.
#     Default Value : ggccProxy.example.com
# proxyPort
#   : Proxy Port configured in Proxy Server
#     Default Value : 3128
# proxyUsingAuth
#   : Check if authentication is required.
#     Default Value : false

#agentUseProxy=false
#proxyHost=ggccProxy.example.com
#proxyPort=3128
#proxyUsingAuth=false

# agentUseSSL
#   : Set this to true if SSL needs to be used between
#     GGCC agent and Sever.
#     Default Value : false
# agentSSLAlgoName
#   : SSL Algorithm name to be used.
#     https://docs.oracle.com/javase/8/docs/technotes/guides/security/
StandardNames.html#SSLContext
#     Default Value : TLSv1.2
# agentUse2WaySSL
#   : Set this to true and configure  agentIDStorePath if DIPC server is 
using 2 way SSL.
#     Default Value : false
# agentIDStorePath
#   : Path to the agent ID store file.
#     Default Value : demoAgentId.jks, located in agent conf directory.
# agentTrustStorePath
#   : Path to the agent ID store file.
#     Default Value : demoAgentTrust.jks, located in agent conf directory.
# agentIDStoreType
#   : Type of the agent ID keystore
#     https://docs.oracle.com/cd/E19509-01/820-3503/ggfen/index.html
#     Default Value : JKS
# agentTrustStoreType
#   : Type of the agent ID keystore
#     https://docs.oracle.com/cd/E19509-01/820-3503/ggfen/index.html
#     Default Value : JKS
# agentIDStoreKeyManagerFactoryAlgorithm
#   : ID Store Key Manager Factory algorithm name
#     https://docs.oracle.com/javase/8/docs/technotes/guides/security/
StandardNames.html#KeyManagerFactory
#     Default Value : SunX509
# agentTrustStoreKeyManagerFactoryAlgorithm
#   : Trust Store Key Manager Factory algorithm name
#     https://docs.oracle.com/javase/8/docs/technotes/guides/security/
StandardNames.html#KeyManagerFactory
#     Default Value : SunX509

#agentUseSSL=false
```

```
#agentSSLAlgoName=TLSv1.2
#agentUse2WaySSL=false
#agentIDStorePath=
#agentTrustStorePath=
#agentIDStoreType=JKS
#agentTrustStoreType=JKS
#agentIDStoreKeyManagerFactoryAlgorithm=SunX509
#agentTrustStoreKeyManagerFactoryAlgorithm=SunX509

# Enumerate Proxy's used for remote output trails
# In the following format
# proxy-<proxyHost>=<actualHost>
#
# proxyHost  -  is the IP/hostname of the proxy used for the remore trail
configuration.
# actualHost -  is the actual remote IP/host of the output trail
configuration.

# proxy-myproxy.example.com=host.example.com

#ODI configuration
odiGeneralConfig=aSampleString

# agentManagedBy
# Indicates if this agent is managed by oracle cloud or On-premises
# Possible values : ONPREMISE , ORACLE_CLOUD
# Default         : ONPREMISE
agentManagedBy=ONPREMISE

# agentPlatform
# Agent meta-data information , denotes the platform of this installed
agent.
# Possible values : Linux, Windows
# Default         : Linux
agentPlatform=Linux
```

Register Your Agent in OAuth2 Mode

The following is an example for the Agent Instance creation with OAUTH2 Mode:

```
.${AGENT_UNZIP_LOC}/dicloud/dicloudConfigureAgent.sh -
dipchost=dipc.example.host.com -dipcport=443 -idcsServerUrl=https://
idcs.identity.example.com agentIdcsScope=https://
idcs.identity.example.com:443external -user=name.example.com -
password=example -agentClientId=example -agentClientSecret=example
```

When configuring an on-premises agent in OAuth mode, you'll need the following properties added to your agent.properties file:

```
# agentAuthMode
# Indicates what is the authentication mode this agent will be using to
connect to server.
# Possible values : OAUTH2, BASIC, LOCAL
#          OAUTH2 : OAUTH2 uses IDCS server to perform OAUTH2 protocol
based authentication for agent requests.
```

```
# Default         : OAUTH2 agentAuthMode=OAUTH2

# agentIdcsServerUrl
# URL of the IDCS server to connect to for authentication in this agent.
# Default : None
# Example : https://idcs.example.identity.com

agentIdcsServerUrl=https://idcs.example.identity.com

# agentIdcsClientScope
# Client Scope of this agent obtained from IDCS server.
# Default : None
# Example : https://idcs.example.identity.com:443external

agentIdcsClientScope=https://idcs.example.identity.com:443external
agentTrustStorePath = /ade_autofs/gd29_3rdparty/nfsdo_generic/JDK8/MAIN/
LINUX.X64/160622.1.8.0.101.0B13/jdk8/jre/lib/security/cacerts

# agentGoldenGateHome
# GoldenGate Home to be configured with the agent.
# Default : <AGENT_INSTALL_LOC>/dicloud/gghome
# For 11g set this to: <AGENT_INSTALL_LOC>/dicloud/gghome11g
```

Configure Agent Secure Sockets Layer (SSL)

An advanced configuration requires configuring Agent Secure Sockets Layer (SSL). You must configure the SSL settings in the agent.properties file to establish an encrypted link between the DIPC agent and the DIPC server.

Set the following in the agent.properties file:

1. `agentUseSSL = true`
2. `agentTrustStorePath= <path to SSL trust store>`
3. Execute and set the unlock password for the trust store configured above.
4. To create a SSL Trust-Store unlock password:

   ```
   .${AGENT_INSTANCE_HOME}/bin/createTrustStoreUnlockPassword.sh
   ```

   OR
5. To update a SSL Trust-Store unlock password:

   ```
   .${AGENT_INSTANCE_HOME}/bin/updateTrustStoreUnlockPassword.sh
   ```

Updating or viewing the configured server (DIPC) credentials
To view the configured username for the agent to talk to the DIPC server:

```
.${AGENT_INSTANCE_HOME}/bin/viewServerCred.sh
```

To update the configured username for the agent to talk to DIPC:

```
.${AGENT_INSTANCE_HOME}/bin/updateServerCred.sh
```

Viewing and updating IDCS client configuration
To update the IDCS Client ID and Client Secret configured for this agent:

```
.${AGENT_INSTANCE_HOME}/bin/updateAgentIdcsClientDetails.sh
```

To view the IDCS Client ID and Client Secret configured for this agent:

```
.${AGENT_INSTANCE_HOME}/bin/viewAgentIdcsClientDetails.sh
```

Configuring Agent through Proxy
Set the following in the agent.properties file:

```
agentUseProxy = true
```

```
proxyHost=<dicloudProxy.example.com>
```

```
proxyPort=<80>
```

If proxy needs authentication, execute the following commands as required:

Creating the credentials for the agent to talk to proxy

```
.${AGENT_INSTANCE_HOME}/bin/createProxyCred.sh
```

Updating the credentials for the agent to talk to proxy

```
.${AGENT_INSTANCE_HOME}/bin/updateProxyCred.sh
```

Viewing the proxy username configured for this agent to talk to a proxy

```
.${AGENT_INSTANCE_HOME}/bin/viewProxyCred.sh
```

Start and Stop the Agent

You can use these commands to start and stop the Agent.

After you install and configure your DIPC agent, run the following command to start it:

```
${agent_unzip_loc}/dicloud/agent/dipcagent001/bin/startAgentInstance.sh
```

Do the following to verify that the agent is running correctly:

- Go to the Oracle Data Integration Platform Cloud console.
- Click the green arrow icon next to your registered agent to see the drill-down status of all the agent components that are running.

Run the following script to view how your agent is running in the background:

`$ ssh your on-premises system IP`.

Do the following to view information of your GoldenGate instances:

- Go to the Oracle Data Integration Platform Cloud console.
- Click the green arrow icon next to your registered agent to see the drill-down status of all the agent components that are running.
- Copy or make a note of the GoldenGate Home directory.
- Run the following scripts in the order listed:
  - `$ cd your GoldenGate Home directory`
  - `$ ./ggsci, and info all`.

Run the following command if you want to stop the agent:

```
${agent_unzip_loc}/dicloud/agent/dipcagent001/bin/stopAgentInstance.sh
```

Monitor Your Agents

The Agents page displays a list of all agents registered with your Data Integration Platform Cloud instance along with the current status of each agent. You can select an agent to view its details, as well as the agent's Drill-down Status that displays the components of the agent and whether they are running or not.

Monitoring the status of your agent enables you to know the current state of your agent, and act on it accordingly.

Status	Description
RUNNING	When the Data Integration Platform Cloud instance receives heart-beat messages from the remote Agent at a regular interval, as specified by the Agent's property "agentHeartBeatInterval" in the [agent-home]`/conf/agent.properties` file. By default, the agent sends a heart-beat message every 10 seconds.
ERROR	When the Data Integration Platform Cloud instance receives heart-beat messages from the remote Agent at a regular interval, and the Agent sends error messages, indicating errors at its end.
UNREACHABLE	When the Data Integration Platform Cloud instance has not received a heart-beat message from the remote Agent for a duration exceeding twice the agent's hear-beat interval.
STOPPED	When the remote Agent has been stopped.

Create a Connection

Connections enable you to specify the connection details to your source, staging, or target data sources. You then select the source and target connections to use as part of your tasks in Data Integration Platform Cloud.

To create a new Connection:

1. Click **Create** from the Getting Started section of the Home page, or select **Connection** from the Create menu in the Catalog. You can also create Connections from any Create Task screen.
2. Complete the fields in the General Information section.
   - For **Agent Name**, select from a list of available agents.
   - For **Type**, select the type of connection you want to create.
3. In the Connection Settings section, enter the relevant connection details for your data source.

   For Connection Details specific to your data source type see:

   - Create a File Connection
   - Create a Kafka Connect Connection
   - Create a Microsoft SQL Server Connection

- Create a MySQL Connection
- Create an Autonomous Data Warehouse Cloud (ADWC) Connection
- Create an Oracle BI Cloud Connector Connection
- Create an Oracle Database Connection
- Create an Oracle CDB Connection
- Create an Oracle Object Storage Connection
- Create an Oracle Object Storage Classic Connection
- Create an Amazon S3 Connection
- Create a Salesforce Connection

4. Click **Test Connection**. If successful, click **Save**, otherwise review your connection details and try again.

View and Edit Connections

After creating your connections, you can find them in the Catalog, along with the data entities harvested from the Connection. From there, you can view, edit, test, or delete them.

You can select any connection to view its details. The details are grouped under following categories:

- **Summary** - contains details of the connection such as its name, description, type, user name, URI, Schema, and tags. You can create new tags on this page. To create a new tag, enter tag name in the **Tags** field and press the enter key.
- **History** - contains details of the actions performed on the connection, so you have a record of how the connection has changed over time.

Set up Default Connections

Setting default connections is a one-time task and necessary to perform certain tasks in Data Integration Platform Cloud. You set Default Connections on the Admin page.

Select an Object Storage Classic Connection

You need to select an Object Storage Classic Connection to:

- To run a Data Preparation Task on a remote file accessible from Data Integration Platform Cloud
- To connect to an Oracle Autonomous Data Warehouse connection
- To connect to an Oracle Object Storage connection

Before you can select a default Connection for Object Storage Classic, you must first create an Object Storage Classic Connection.

See Create an Oracle Object Storage Classic Connection.

To select an Object Storage Classic Connection:

1. In the Data Integration Platform Cloud navigation menu, click **Admin**.

2. From the Object Storage Classic menu, select an available Object Storage Classic connection.
3. Click **Save**.

Select a Default Stage Connection

To use the Transformation Creation Tool when creating a Data Preparation Task, you need to configure a default stage connection.

Data Integration Platform Cloud uses the default stage connection to temporarily store data harvested from your source connection for use with the Transformation Creation Tool.

If you haven't created a default stage Connection, you must first create one.

See Create a Connection.

To set a default stage connection:

1. In the Data Integration Platform Cloud navigation menu, click **Admin**.
2. From the Oracle menu, select a default stage connection.
3. Click **Save**.

Understand the Catalog

The Catalog enables you to access components such as Tasks, Connections, Data Entities, Data Assets, and Execution Environments, that exist in Data Integration Platform Cloud. You can also create and edit these components from the Catalog.

Use the Catalog to:

- Create Connections, Data Assets, Tasks, and Execution Environments.
- Access existing Connections, Data Entities, Data Assets, Tasks, and Execution Environments.
- Filter, sort, tag favorites, and search objects.
- Edit, delete, or refresh Connections.
- Review component details.
- Execute, edit, or delete Tasks.
- Edit or delete Data Entities.
- View a history of actions performed on all the components in the Catalog.

There are several ways to filter the list of components in the Catalog. To view only Connections, click the menu next to Catalog and select **Connections**. Likewise, you can select **Tasks**, **Data Entities**, **Data Assets**, or **Execution Environments** to see only Tasks, Data Entities, Data Assets, or Execution Environments respectively. Click **Filter Favorites** to view all Connections, Tasks, Data Entities, and Data Assets that you marked as Favorite. You can use the **Sort By** menu to sort the list by Most Recent (default), Alphabetical, and Popularity. Click **Show/Hide Filters** to toggle the filter selectors. Filter selectors change depending on the view you've set (either All, Connections, Tasks, Data Entities, Data Assets, or Execution Environments).

Search the Components in the Catalog

When you enter a text string into the **Search** field, the search engine returns any Connections, Tasks, and Data Entities that meet your criteria, searching all fields of these components for a match. As you type in your text string, the search engine actively returns components, and lists them underneath the search bar for you to quickly access what you're looking for.

Search Components by Properties

You can also enter property name and value pairs into the **Search** field to limit your search to specific fields and values. You can search on the following properties:

- id
- name
- description
- tags
- createdBy
- connectionType
- entityType
- taskType
- technology
- connection
- schema
- attributes

The property names are case sensitive, so be sure to enter the property names in the **Search** field as you see them in Data Integration Platform Cloud.

Enter a search string in the following format to perform a property search:

`<property-name>:<value>`.

You can also use operators such as OR (||) and AND (&&) to build on your search query. The default operation is OR. For example, `category:task name:dis_view` is the same as `category:task OR name:dis_view`.

Use parentheses around property name and value pairs to indicate the order of operations to be performed. For example,

```
(category:task || name:dis_view) && createdBy:Joe Smith
```

View Component History

You can find past actions, who performed them, and when in the History tab of any component in the Catalog.

All components listed in the Catalog such as Tasks, Data Entities, or Connections have the following actions in common: First you create them, then you can update or delete them. When you select any item listed in the Catalog, you view its Summary page. From here, you can access the component's History. The History tab is a read-only page that displays a table with three columns:

- **Action**: Displays either Create, Update, or Delete.
- **User**: Displays the name of the user who performed the action.

- **Last Updated**: Displays the date when the user performed the action.

Data Entities

Data Entities are the data objects you can use as your sources or targets.

Data Entities are listed in the Catalog along with other components such as Connections and Tasks. To view only Data Entities in the Catalog, select **Data Entities** from the Type filter menu. You can select any data entity to view its details. The details are grouped under the following categories:

- **Summary**: Contains details of the data entity, such as its name, identifier, description, type, and popularity.
- **Metadata**: Contains details of the data in the data entity along with examples. The examples are seen in the Sample Values column with five sample values.

  You can also select each metadata object to see its profiling metrics. By examining the data, you can:

  - Determine how easily the existing data can be used.
  - Evaluate data quality.
  - Review the risks of integrating data in new applications, including joins.
  - Access source database metadata, key candidates, foreign-key candidates, patterns and distributions, and functional dependencies.
- **Data**: Contains the content of the data entity. You can also access the profiling metrics from the data view.

  > **Note:**
  >
  > Profiling through the DIPC Agent is not supported. If the Connection is not available within the DIPC console, profiling is unavailable.

- **History**: Contains details of the actions performed on the data entity, so you have a record of how the definition of the data entity has changed over time.

Data Entity Editor

After data entities are harvested, their summary details can be edited. To do this, click **Edit** on the data entity’s Summary page. The fields that you can edit are:

- **Name**: Name of the data entity.
- **Identifier**: While the initial value of this field is auto-generated based on the value of the Name field, you can still edit it, but it can only contain capital letters, numbers, and underscores (_).
- **Description**: Description of the data entity.
- **Contact**: E-mail address or phone number of the data entity’s owner or another Data Integration Platform Cloud user. Click **Add New Contacts** to add more contacts.
- **Tags**: Tags let you group and organize components in the Catalog so that you can find them easily.

Part II

Perform Tasks in Data Integration Platform Cloud

Perform various tasks in Data Integration Platform Cloud, such as replicating and synchronizing your data, cleansing and preparing your data, importing and executing ODI scenarios, and adding data to a data lake to name a few.

Topics

Create a task, run tasks, and monitor the jobs created from your tasks.

- Synchronize Data
- ODI Execution
- Replicate Data
- Add Data to Data Lake
- Prepare Data
- Monitor Jobs
- Create Policies
- Maintain Oracle Data Integration Platform Cloud

4

Synchronize Data

Synchronize Data enables you to keep your data sources in sync.

Topics:

- What is Synchronize Data?
- What’s Certified for Synchronize Data?
- Before You Synchronize Data
- Create a Synchronize Data Task
- Use the Data Entity Selector
- Synchronize Data Job Actions

What is Synchronize Data?

The Synchronize Data Task enables you to sync data between two cloud data sources or from your on-premises data source to your cloud data source. You can also select specific data entities to synchronize.

Generally, when you synchronize data between a source and a target data source that have the same database version, only one agent is sufficient to complete the task. However, there are times where you'd need two different agents:

- Running the capture and delivery processes close to the Source Database and Target data source respectively.
- When the source and target databases have different major release versions, for example, the source database is 11g and target is 12c.

In both cases, you select one agent for your source and another for your target Connection, and then use those Connections in the same Synchronize Data Task. Data Integration Platform Cloud automatically initiates the GoldenGate pump action when you run the task and starts the job.

For more information about selecting agents, see Data Source Connection Details.

What's Certified for Synchronize Data?

Review the supported agents, data sources, and limitations before you choose your source and target for the Synchronize Data Task in Oracle Data Integration Platform Cloud.

> **Note:**
>
> - All data sources must have x86_64, the 64-bit version of x86 operating systems, with the latest update.
> - You need a VPN if you include the Initial Load option for the Synchronize Data Task, unless the Connection is Oracle Database Cloud Classic.

Connection type	Version	OEL	RHEL	SLES	Windows	Source	Target
Oracle Database Cloud Classic	12.2	6.x	no	no	no	yes	yes
Oracle Database Cloud Classic	12.1	6.x	no	no	no	yes	yes
Oracle Database Cloud Classic	11.2	6.x	no	no	no	yes	yes
Oracle Database	12.2	6.x	6.x, 7.x	11, 12	2012, 2016	yes	yes
Oracle Database	12.1	6.x	6.x, 7.x	11, 12	2012, 2016	yes	yes
Oracle Database	11.2.0.4	6.x	6.x, 7.x	11, 12	2012	yes	yes

Ensure that you set up the agents only for data sources that are certified for your tasks.

See Agent Certifications.

Before You Synchronize Data

To create a Synchronize Data Task, you must first download and configure your Agent, and then create Connections to your source and target data sources.

Download and configure your agents

- Set up an Agent
- Set up a Remote Agent for ODI

- Set up an External Library Path for GoldenGate 12c

Create Connections

Connections enable you to enter the connectivity details for your source and target data sources, so you can use them for different tasks when needed. Only Oracle sources and targets are supported for the Synchronize Data Task.

See:

- Create a Connection
- Create a PDB Connection
- Create an Oracle Database Connection

Configure GoldenGate

Depending on the version of GoldenGate you have, you may need to perform additional configuration steps.

See Configure GoldenGate.

Create a Synchronize Data Task

You can create a Synchronize Data task from the Getting Started section of the Home page or from the **Create** menu in the Catalog.

1. In the General Information section, enter a **Name** and **Description** for your Task.

   > **Note:**
   >
   > You'll use the value you provide in the **Name** field to identify it in the Catalog. The **Identifier** is auto-generated based on what you enter into the Name field and is used when you search for this Task in menus and other collections. You can edit the Identifier later, but it can only include capital letters, numbers, and underscores (_).

2. In the Source Configuration section, select the **Connection** and the corresponding **Schema** that serves as the source for this task. Click **Add Connection** if you haven't set one up yet.
3. In the Target Configuration section, select the **Connection** and the corresponding **Schema** that serves as the target for this task. Click **Add Connection** if you haven't set one up yet.
4. For Advanced Options, select at least one of the following:
   - **Include Initial Load**: Executes the Initial Load actions as part of the Task execution.
   - **Include Replication**: Executes the Replication actions as part of the Task execution.

   Note: You can select both of the options if you want to perform Initial Load and Replication actions during the Task execution. Select one of the options to perform only that part of the task in the Task execution, and then perform the other option by some other means, such as in a separate Task.

You must avoid running an Initial Load and a Replication for the same set of tables simultaneously in two different Tasks. An Initial Load removes the table it's copying at the target and recreates it using the metadata from the source table. Since Replication copies the data in real time, it'll find that the tables are removed from that target and it'll stop. There may also be instances where the data conflicts as Initial Load and Replication copies the same data.

After completing the Synchronize Data fields, you can do one of the following:

- Click **Save** to save your task
- Click **Configure Data Entities** to refine data entities included in your Synchronize Data Task
- Click **Save & Run** to save and start the synchronization task (executes the job)

Once your Synchronize Data task is executed, you'll find details on the Jobs pages.

Use the Data Entity Selector

When you click **Configure Entities** from the Create Synchronize Data Task page, you're brought to the Data Entity Selector. The Data Entity Selector enables you to set rules that include or exclude data entities in your Synchronize Data Task.

There are four areas of the Data Entity Selector:

- **Setup:** Lists the source Connection and schema to be used for this Synchronize Data Task. Click **Edit** to return to the Synchronize Data Task page to modify your source or target details.
- **Available Data Entities**: Lists all available Data Entities harvested from your source Connection, by default. Enter Filter Rule patterns into the **Filter Rules** field (click **Include** or **Exclude** as needed) to filter the list of data entities that satisfy your rule. You can also click **Include** or **Exclude** to explicitly include or exclude them from your task. This results in a rule added to the Rules table, with the Data Entity name in double-quotes.
- **Selected Data Entities**: Lists the Data Entities that fulfill the rules you’ve entered. Click **Options** for any Data Entity to exclude it from the list, which then adds an Exclude rule to your Rules table.
- **Rules**: Lists all the user-defined filter rules. Use the **Options** menu for any rule you want to reorder or delete.

The guidelines for using Filter Rules in the Data Entity Selector are as follows:

Patterns

You can enter character expressions into the **Filter available data entities** field to filter Data Entities. Acceptable character expressions are:

- A question mark (?) in the pattern matches exactly one character in the data entity string value.
- An asterisk (*) in the pattern can match zero or more characters in the data entity string value.
- A backslash (\ ) in the pattern is an escape character. You can include the actual characters ? or * in the pattern by using the escape character. If the escape character precedes the character ? or * in the pattern, then Data Integration Platform interprets this character literally in the pattern rather than as a special

pattern-matching character. You can also search for the escape character itself by repeating it. For example, if \ is the escape character, then you can use \\ to search for \.

- If no quotation marks are used, the pattern is uppercased and then matched. If mixed cases or lowercase is searched, it must be put into quotes. For example, for tables named BONUS, Bonus, bonus:
  - B* matches to BONUS and Bonus
  - b* matches to BONUS and Bonus
  - "B*" matches to BONUS and Bonus
  - "b*" matches to bonus

Case Sensitivity

Case sensitivity is not applied unless the pattern entered is surrounded by double-quotes, for example:

- The pattern, MyCustomer (without quotes) matches MyCustomer, Mycustomer, mycustomer, MyCUSTOMER, and so on.
- The pattern, "MyCustomer" matches MyCustomer only.
- The pattern, MyCust*omer (without quotes) matches MyCustomer, Mycust1omer, mycust1234omer, MyCUSTOMER, and so on.
- The pattern, "MyCust*omer" matches MyCustomer, MyCust1omer, MyCustAAAAomer and others but NOT Mycust1omer, mycust1234omer, MyCUSTOMER as in the previous example.

Include and Exclude Rules

Each filter rule is specified as an Include or Exclude rule.

- An "Include" rule will include all data entities with names matching the character expressions of the specified rule patterns.
- An "Exclude" rule will exclude all data entities with names matching the character expressions of the specified rule patterns.

When working with Include and Exclude rules, keep the following in mind:

- Include * is assumed if your rules list is empty.
- An Include with no wildcards is included even if it's excluded by an Exclude with a wildcard.
- An Include rule with a wild-card cannot follow an Exclude rule with a wildcard.

Rules Ordering

Explicit rules (those without any wildcard characters) are processed by Data Integration Platform Cloud in the order you specify in the Rules list. For example,

```
include s1
include s2
include s3
exclude s4
exclude s5
include s6
exclude s7
```

is executed by Data Integration Platform Cloud as follows: `(((((( {s1}  {s2}) {s3}) - {s4}) -{s5})  {s6}) - {s7}) ...` where specifies an include and - for excludes.

Explicit rules have precedence over rules without wildcards, regardless of the order you specify. For example:

Example A: If the ordered list of rules is:

1. `Include Customer`
2. `Exclude Customer*`

then the result is the table `Customer` is selected.

Example B: If the ordered list of rules is:

1. `Include Customer?`
2. `Exclude Customer*`

Then the result is no table with the name starting with `Customer` is included.

Example C: If the ordered list of rules is:

1. `Include A1`
2. `Exclude B1`
3. `Include B*`
4. `Exclude A*`

Then the result is B1 is excluded and A1 is included because the explicit rules take precedence over the rules with wildcards. Imagine if the explicit rules are processed after all the wildcard rules, then the order of rules for processing would be:

1. `Include B*`
2. `Exclude A*`
3. `Include A1`
4. `Exclude B1`

You cannot use a rule pattern more than once in your Rules list. For example, all of the following are not allowed:

```
1. Include A
2. Include A

1. Include A
2. Exclude A

1. Exclude A
2. Include A

1. Include A*
2. Exclude A*
```

Any wildcard Include rule is invalid if it follows a wildcard Exclude rule. For example, if the ordered list of rules is:

1. `Exclude Customer*`
2. `Include Customer?`

then the second rule is invalid. No table with the name starting with `Customer` is included.

If the ordered list of rules is:

1. `Exclude Customer*`
2. `Include Customer`

then the `Customer` table is included because the explicit rule takes precedence over the wildcard Exclude rule.

If the ordered list of rules is:

1. `Exclude Customer*`
2. `Include Product*`

then no table with a name starting with `Product` is included.

Synchronize Data Job Actions

When you run a Synchronize Data Task with Replication, the job is composed of the following actions, unless noted otherwise. You can review these Job Actions on the Job Details page of the Monitor Dashboard for your Synchronize Data job.

- Initialize Capture
- Initialize Delivery
- Start Capture
- Initial Load (only when Initial Load advanced option is selected)
  - Drop DBLINK
  - Create DBLINK
  - ODI_Variable
  - DBLINK_DATAPUMP
  - DBLINK_DATAPUMP_METRICS
  - DROP DBLINK
- Start Delivery

5 ODI Execution

Perform bulk data transformations using Scenarios created in ODI Studio.

Topics:

What is ODI Execution?

This task enables you to perform bulk operations and transformations.

The ODI Execution Task enables you to run Scenarios created in Oracle Data Integrator (ODI) Studio, and then monitor them centrally in Data Integration Platform Cloud.

See, Using Scenarios to learn more about Scenarios.

What's Certified for ODI Execution?

Review the supported agents, data sources, and limitations before choosing your source and target for ODI Execution.

> **Note:**
>
> - All data sources must have x86_64, the 64 bit version of x86 operating systems, with the latest upgrade.
> - Flat files located on all operating systems and versions can work for ODI Execution.
> - SQL Server is only certified on Windows.

Connection type	Version	OEL	RHEL	SLES	Windows	Source	Target
Autonomous Data Warehouse Cloud	18.1	6.x	6.x, 7.x	11, 12	no	no	yes

Connection type	Version	OEL	RHEL	SLES	Windows	Source	Target
Oracle Database Cloud Classic	12.2	6.x	no	no	no	yes	yes
Oracle Database Cloud Classic	12.1	6.x	no	no	no	yes	yes
Oracle Database Cloud Classic	11.2	6.x	no	no	no	yes	yes
Oracle Database	12.2	6.x	6.x, 7.x	no	no	yes	yes
Oracle Database	12.1	6.x	6.x, 7.x	no	no	yes	yes
Oracle Database	11.2.0.4	6.x	6.x, 7.x	no	no	yes	yes
Oracle Object Storage	n/a	n/a	n/a	n/a	n/a	yes	yes
SQL Server	2016 EE (SP2)	no	no	no	2016	yes	yes
SQL Server	2016 EE (SP1)	no	no	no	2016	yes	yes
SQL Server	2014 EE	no	no	no	2012	yes	yes
SQL Server	2012 EE	no	no	no	2012	yes	yes
MySQL	5.x	no	6.x, 7.x	no	no	yes	yes
Flat Files	n/a	yes	yes	yes	yes	yes	yes

After you verify your data source operating systems and versions, you must set up agents only for the data sources that are certified for your tasks. Ensure that you install agents on certified operating systems and that the agents can connect to your data sources. See Agent Certifications.

What are the Limitations for ODI Execution?

The following are limitations of the ODI Execution Task:

- If you run multiple tasks with the same agent at the same time, the tasks may take longer to execute.
- You can only import Patch Deployment Archives. Other deployment archive types such as Execution Deployment Archives are not supported.

Before You Create an ODI Execution Task

To perform an ODI Execution Task in Data Integration Platform Cloud, you need to download and set up your Agent, and create your Connections.

Set up a Scenario

Using ODI Studio, set up a Scenario. You may also need a deployment archive if you want to import the Scenarios into Data Integration Platform Cloud.

For more information, see Using Scenarios.

Set up a Remote Agent

- Set up an Agent
- Set up a Remote Agent for ODI

Create Connections

- Create an Oracle Database Connection
- Create a MySQL Connection
- Create a File Connection
- Create an Oracle Object Storage Connection
- Create an Autonomous Data Warehouse Cloud (ADWC) Connection

Create an ODI Execution Task

Creating an ODI execution task requires general information like name and description of the task.

To create an ODI Execution Task:

1. In the Getting Started section of the Home page, click **Create** for ODI Execution, or click **Create** in the Catalog, select **ODI Execution**.
2. In the **Edit ODI Execution Task** screen:
   a. Under the **General Information** section, enter a name and description for this task.
   b. Go to the **Connections** section:
      - **Scenario Name**: Select a scenario you want to execute. Click **Import** to select a scenario from a patch deployment archive created in ODI Studio.
      - **Logical Schema-**
        The logical schemas that are used for source and target connectivity in ODI Studio. Click **Add Connection** (the plus sign), which appears by clicking on the corresponding row, to create multiple connections.
      - Select the appropriate **Connection** and the **Schema**.
3. Click **Save** to save the task or click **Save and Run** to save and start the ODI Execution job, which brings you to the job page where you can monitor your job.

6 Replicate Data

The Replicate Data Task of Oracle Data Integration Platform Cloud captures new transactions in the source and streams them to target.

Topics:

What is Replicate Data?

The Replicate Data Task of Oracle Data Integration Platform Cloud captures changes in a data source and updates the target in real time with that change.

In the Replicate Data task you select a source and a target connection for your task. Then, from the moment that you run this task, any new transaction in the source data is captured and delivered to the target. This task doesn't perform an initial copy of the source, so you'll get all the changes from the point of time that you started your job.

Here are some examples of when you would use this task:

- **Cloud on-boarding:** Replicate on-premises data to new cloud applications in real-time using Oracle Database Cloud Classic or Autonomous Data Warehouse Cloud.
- **Real-time reporting:** Capture real-time data from multiple on-premises data sources and deliver to Oracle Database Cloud Classic or Autonomous Data Warehouse Cloud to create reports.
- **Query off-loading:** Off-load production transactions to Oracle Database Cloud Classic.
- **Real-time data warehousing:** With Oracle database (on-premises or cloud) as your source and Oracle Autonomous Warehouse Cloud as your target, replicate data to a warehouse to set up a staging environment for downstream ETL or real-time data warehousing.
- **High availability:** Create a multi-site or multi-cloud high availability strategy with synchronized data sources.
- **Streaming to Kafka topics:** Get streams of data into your Kafka topics for a real time data analysis With Kafka Connect as a target. Along with the data that's streamed to your topics, customize delivery of additional information about the

changed data such as what kind of transaction was performed on that data and when it was committed.

- **Complex event processing and data analytics:** By customizing what data should be sent to Kafka topics, you can prepare it to be further analyzed with Oracle Stream Analytics (OSA). For example, you can send real time information about clients, from your databases to Kafka topics and then with OSA, send real time message offers to clients when they enter a shopping center.

What's Certified for Replicate Data?

Review the supported agents, data sources, and limitations before choosing your source and target for Replicate Data in Oracle Data Integration Platform Cloud.

- All data sources must have x86_64, the 64 bit version of x86 operating systems, with the latest upgrade.
- Kafka Connect target can only be on-premises, because Kafka Connect is not certified on Oracle Big Data Cloud yet.
- For Kafka Connect data sources, a remote agent must be set up on the same machine as Kafka Connect, and Kafka Connect is on-premises only, so the agent must be configured on-premises.

Connection type	Version	OEL	RHEL	SLES	Windows	Source	Target
Autonomous Data Warehouse Cloud	12.2	6.x	6.x, 7.x	11, 12	no	no	yes
Oracle Database Cloud Classic	12.2	6.x	no	no	no	yes	yes
Oracle Database Cloud Classic	12.1	6.x	no	no	no	yes	yes
Oracle Database Cloud Classic	11.2	6.x	no	no	no	yes	yes
Oracle Database	12.2	6.x	6.x, 7.x	11, 12	no	yes	yes
Oracle Database	12.1	6.x	6.x, 7.x	11, 12	no	yes	yes
Oracle Database	11.2.0.4	6.x	6.x, 7.x	11, 12	no	yes	yes
Kafka Connect by Confluent	4.1.x	6.x, 7.x	6.x, 7.x	11, 12	no	no	yes

Connection type	Version	OEL	RHEL	SLES	Windows	Source	Target
Kafka Connect by Confluent	4.0.0	6.x, 7.x	6.x, 7.x	11, 12	no	no	yes
Kafka Connect by Confluent	3.2.x	6.x, 7.x	6.x, 7.x	11, 12	no	no	yes
Kafka Connect by Confluent	3.1.x	6.x, 7.x	6.x, 7.x	11, 12	no	no	yes
Kafka Connect by Confluent	3.0.x	6.x, 7.x	6.x, 7.x	11, 12	no	no	yes

After you verify your data source operating systems and versions, then you must set up agents only for data sources that are certified for your tasks. Check the operating systems that the agents are certified to run on, to ensure the agent install location is appropriate and connects to your data source. See Agent Certifications.

Replicate Data Task Limitations

Review the Replicate Data Task limitations before you perform the task.

- Replicate Data task doesn't include an initial load and captures the change in data from the moment that the job is started. If you want do an intial load from one data source to another before you synchronize them, then run a Synchronize Data Task and select the Initial Load option. See Synchronize Data.
- You can't always use the Synchronize Data task for an initial load of a data source that you want to use for Replicate Data, because certified data sources for Synchronize and Replicate Data are not the same. For example, you can stream transactions to Autonomous Data Warehouse Cloud, but you can't do an initial load to deliver data to it by using Synchronize Data.
- To replicate data you can either use one or two agents. If you use two agents, you set up one for the source and one for the target. If you use one agent, then you must provide both source and target information in the `agent.properties` file for that agent. A classpath points to a local directory and a Kafka Connect connection depends on Kafka libraries. Therefore, in a one agent option, you must **install the DIPC agent on the Kafka machine**.
- The change in the source database isn't available for processing until the Data Manipulation Language (DML) commands such as insert, update and delete have been committed. So in that respect only transactions are replicated in Replicate Data. Data Definition Language (DDL) operations such as drop or create are not part of the replication process.
- Replicate Data is one to one and uni-directional, which means that it accepts one source and one target for each task. If you want to include several sources to

deliver data to one target, you can create a Replicate Data Task for each source and have them all deliver to the same target.

Before You Replicate Data

To create a Replicate Data Task, you must first set up your agent(s) with their appropriate components and then create Connections to your source and target data sources.

Download, Register and Set up Agents

Oracle Data Integration Platform Cloud communicates with your data sources by offering agents to be installed on data sources or on machines with access to those data sources. These agents orchestrate tasks and communicate information among data sources and the Data Integration Platform Cloud server. You must download the following replication components with your agent download.

- **Oracle 12c (OGG)** for Oracle Database 12.1, 12.2 and Autonomous Data Warehouse
- **Big Data (OGG)** for Kafka Connect

Create Connections

When you create a connection, you enter the connectivity details for your data sources, so you can use them for different tasks when needed. Here's a list of connections that you can create for the Replicate Data Task. Select the one that works for you.

- Create an Oracle Database Connection
- Create a PDB Connection
- Create an Autonomous Data Warehouse Cloud (ADWC) Connection
- Create a Kafka Connect Connection

Generate Wallet and Master Key

For your data to be encrypted, when it's traveling across the network during the Replicate Data Task, you must generate a wallet with a master key.

By default, the data that the Replicate Data Task sends across the network is not encrypted. Data Integration Platform Cloud's agents can encrypt captured data from the source, before it's sent across the network, and then decrypt it in the target. You can select an Advanced Encryption Standard (AES) option when you set up your Replicate Data Task. However, you must also generate a wallet with a master key through your agent processes. The agents use the master key in the source to encrypt the data. You must copy the same master key to the target wallet directory, so that the target agent can decrypt the data with the same key.

To copy the same master key to the target wallet directory:

1. On the environment that is hosting your **running DIPC agent** for the **source**:
   - For Oracle 12c: Navigate to the gghome folder of your agent, located in `<agent_unzip_loc>/dicloud/gghome`
   - For Oracle 11g: Navigate to the gghome11g folder of your agent, located in `<agent_unzip_loc>/dicloud/gghome11g`.

2. Enter the following commands:

   ```
   [xxx:gghome] $ggsci
   GGSCI (xxx) 1 > create wallet
   Created Wallet
   Opened Wallet
   GGSCI (xxx) 2 > add masterkey
   <current time> INFO xxx created version 1 of master key
   'OGG_DEFAULT_MASTERKEY' in Oracle Wallet.

   GGSCI (xxx) 3 > info masterkey
   Masterkey Name: OGG_DEFAULT_MASTERKEY

   Version   Creation Date   Status
   1         <creation time>  Current

   GGSCI (xxx) 4 > exit
   ```

3. Ensure that the wallet, `cwallet.sso` is created in the `dirwlt` directory:

   ```
   [xxx:gghome] cd dirwlt
   [xxx:dirwlt] ls
   cwallet.sso
   ```

4. Copy the `cwallet.sso` to the the environment that is hosting your running DIPC agent for the **target** in the following location:
   - For Oracle 12c and Autonomous Data Warehouse Cloud: `<agent_unzip_loc>/dicloud/gghome/dirwlt`
   - For Oracle 11g: `<agent_unzip_loc>/dicloud/gghome11g/dirwlt`
   - For Kafka Connect: `<agent_unzip_loc>/dicloud/gghomebigdata/dirwlt`

Troubleshoot

If `ggsci` command doesn't work in the source agent environment, then add the path to the `oci` directory to your library path.

The library paths and the `ggsci` command are automatically applied after you register and start your agent, but if you haven't started your agent, then you can manually add the path to the `oci` directory.

Run the following command from the `gghome` or `gghome11g` directory first and then run your `ggsci` command.

```
[xxx:gghome] export LD_LIBRARY_PATH=<agent_unzip_loc>/dicloud/oci
[xxx:gghome] $ggsci
```

What's Auto-Match?

Auto-match is a mapping pattern in Replicate Data Task of Data Integration Platform Cloud.

Auto-match looks at the mapping rule in the source section of the Replicate Data Task and only keeps the data entities from the filtered source schema that match the target schema. This mapping pattern is applied to Oracle Database, Database Cloud Classic and Autonomous Data Warehouse Cloud targets.

Because Data Definition Language (DDL) operations such as drop or create are not part of the replication process, he Replicate Data Task doesn't create missing tables. Therefore, you must define the tables that are part of the replication, in the target schema before you run a Replicate Data Task.

Let's suppose the mapping rule is `Include SRC.*`. If `SRC` has tables `A1, A2` and `A3`, then auto-match will attempt to match each of these data entities with the target. If your target schema `TGT`, has tables `A1, A2, B1` and `B2`, then only the data changes in tables `A1` and `A2` will be updated in the target.

If there is a transaction with an insert in `SRC.A1`, then that change will be updated in `TGT.A1`. If there are changes in `SRC.A3`, then those changes will be ignored because the target doesn't have table `A3`. Tables `B1` and `B2` in the target are not affected, because there are no `B1` and `B2` in the source to replicate their change in the target.

Create a Replicate Data Task

Here’s how you can set up a Replicate Data Task:

1. From the Home page or the Create menu in the Catalog, select **Create Replicate Data**.
2. Name and describe your task in the **General Information** section. The Identifier is to identify this task through a string with no spaces. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
3. Select an encryption level for the captured data that's sent across the network. The higher the number, the harder it is for hackers to decrypt the data. Choosing **None** will send your original data, without encryption to the target.
4. If you choose an encryption level, then ensure that you have generated a wallet according to Generate Wallet and Master Key instructions, or you will generate one, before you run the task.
5. Click **Design** to select your connections and mapping pattern.
6. In the **View** section, click **Source** to display properties for Source.
7. In the **Properties** for Source, in the **Details** tab, select a **Connection** that serves as the source for this task:
8. In the **Properties** for Source, in the **Schemas** tab, select a schema from the list and then click the plus icon to add it as a mapping rule.

   For example. if you select schema `A` and then click the plus icon, the rule `Include A.*` appears in the **Rule Applied** section.

   Only **one rule** is allowed, so only one schema can be included in a mapping rule. You can further narrow down a mapping rule for a schema by including a pattern. For example, instead of `A.*`, you can edit the field to `A.EMP*` to only include tables in schema `A` that start with `EMP`.

   To reset a mapping rule, click **Reset**. The **default** rule uses the schema assigned to the **Source Connection**. You can edit the default schema for a Connection in its detail page or override the rule by replacing it with another schema by using the plus icon.

9. In the **View** section, click **Target** to display properties for target.
10. In the Properties for **Target**, in the **Details** tab, select a connection that serves as the target for this task.
11. In the Properties for **Target**, in the **Schemas/Topics** tab:
    - **For an Oracle target:**

      Select a schema in the target for an **auto-match** mapping pattern. See What's Auto-Match?
    - **For a Kafka Connect target:**

      For **Mapping Pattern** and **Key Mapping Pattern**, either select an option from the drop down menu or enter your own patterns.

      You can override both of these fields. The default value for these fields come from the Topic Mapping Template and Key Mapping Template fields in the Kafka Connect **Connection**.

      Any mapping pattern that you enter can be the name of a topic or a key, so consider the power and the responsibility of defining your own mapping patterns. For example, you can name the mapping pattern `ABC_CHANGES` and all the delivered data will go to a topic called `ABC_CHANGES`. You can partition this topic based on the key. For example `${table_name}` and that way all changes for the same table will go to the same partition. Because any value is possible, these fields are not validated. If you want your topics to be resolved based on predefined variables, then you must enter them correctly. There is no pre-check. For example, if instead of `${schemaName}_${tableName}`, you enter `schemaName_tableName` for the **Mapping Pattern**, then the topic name will not resolve to the schema and table names. Instead, you will have one topic called `schemaName_tableName` and all the changed data is saved in this topic.
12. Do one of the following:
    - Click **Save** to save your task and run it later.
    - Click **Save & Run**, to save your task and create a job from this Replicate Data Task.

Monitor a Replicate Data Task

When you run a Replicate Data Task, you create a Replicate Data job. Replicate Data jobs are listed in the Replicate Data tab of the **Monitor** page. Click a task name to see its job details.

A **Replicate Data** job is composed of the following **actions**:

- **Initialize Capture:** The Replicate Task plugin starts some applications for the capture process on the source. After the initialization is complete, status should be **Successful**.
- **Initialize OGG Data Pump:** The Replicate Task plugin starts a process on the source to send data to the target. This process only applies to Kafka targets. After the initialization is complete, status should be **Successful**.

- **Initialize Delivery:** The Replicate Task plugin starts some applications for the delivery process on the target. After the initialization is complete, status should be **Successful**.
- **Start Capture:** The capture process starts when it detects a change in the source data entity that's to be mapped. The status should be **Running**.
- **Start OGG Data Pump:** The Pump process sends the captured change data to the target. This action applies only to Kafka targets. The status should be **Running**.
- **Start delivery:** The delivery process starts after the capture process has captured the change in the source data entity that's to be mapped. The status should be **Running**.

The actions are in the Summary page of every job.

See Monitor Jobs for more information.

7 Add Data to Data Lake

A Data Lake is a massive repository that can include structured, semi-structured, unstructured, and even raw data. It enables you to store and organize large volumes of highly diverse data from a variety of sources.

Topics:

This chapter includes the following topics:

What is an Add Data to Data Lake Task?

By adding data to Data Lake, you can store vast amounts of data, enabling deep analytics, big data processing, and machine learning. You can add data from a variety of sources into the Data Lake.

Data can be ingested from a variety data sources, including relational data sources or flat files. Harvested metadata is stored in the Data Integration Platform Cloud Catalog, and the data will be transformed and secured within the target Data Lake for downstream activities.

Data Lake Builder provides the capability to:

- Add data from various sources into the Data Lake, using a simple, configurable, and flexible workflow.
- Configure a Data Lake.
- Configure file format and file options for files added to Data Lake.
- Configure a connection to sources outside of the Data Lake.

What's Certified for Add Data to Data Lake?

Review the supported agents, data sources, and limitations before choosing your source and target for the Add Data to Data Lake Task (Data Lake Builder) in Oracle Data Integration Platform Cloud.

> **Note:**
>
> - All data sources must have x86_64, the 64-bit version of x86 operating systems, with the latest upgrade.
> - The only target that you can use to build a Data Lake and then add data to it, is Oracle Object Storage Classic.

Connection Type	Data Source Version	OEL	RHEL	SLES	Windows	Source	Target
Oracle Database Cloud Classic	12.2	6.x	no	no	no	yes	no
Oracle Database Cloud Classic	12.1	6.x	no	no	no	yes	no
Oracle Database Cloud Classic	11.2	6.x	no	no	no	yes	no
Oracle Database	12.2	6.x	6.x, 7.x	11 & 12	2012, 2016	yes	no
Oracle Database	12.1	6.x	6.x, 7.x	11 & 12	2012, 2016	yes	no
Oracle Database	11.2.0.4	6.x	6.x, 7.x	11 & 12	2012	yes	no
Flat Files	n/a	yes	yes	yes	yes	yes	no
Oracle Object Storage Classic	Latest	n/a	n/a	n/a	n/a	no	yes
Autonomous Data Warehouse	18.1	6.x	6.x, 7.x	11, 12	no	no	yes
Amazon S3	Latest	n/a	n/a	n/a	n/a	yes	no

After you verify your data source, operating systems, and versions, you must set up agents only for data sources that are certified for your tasks.

See Agent Certifications.

What’s Supported for Add Data to Data Lake Task?

Before adding data to the Data Lake, you must consider what source and target data sources are supported for the Add Data to Data Lake Task.

You can use the following source and target data sources for the Add Data to Data Lake Task:

Supported Source Data Sources

- Oracle Databases
- Relational Databases
- Amazon S3
- Flat files
- Parquet
- JSON
- Delimited (csv, tsv)

Support Target Data Sources

- Parquet
- Delimited (csv, tsv)
- Autonomous Data Warehouse (ADWC)

Limitations for Add Data to Data Lake Using an Execution Environment

Consider the following limitations when you're using an Execution Environment to add data to a Data Lake.

- You can't create an external table Oracle Autonomous Data Warehouse Cloud (ADWC) if any of the following conditions are true:
  - Your target file type is Parquet. ADWC doesn't support Parquet.
  - Your target file name is alphanumerical and starts with a number or a special character, as the target file name doesn't follow ADWC naming conventions.

    See Schema Object Naming Guidelines.
  - Your target file name contains a floating decimal with suffix f, for example, 2.4f.
  - Your varchar column size is greater than 4000 and decimal column size is greater than precision 35 and a scale of 15.
- If your source or target file name contains a space, it is replace by an underscore (_).
- If the source file uses a colon (:) as a delimiter, it will fail, unless it contains a timestamp within the Text Qualifier. Use a delimiter other than a colon.
- If the data within a column is inconsistent (for example, a mix of integers, decimals, and strings) then the number of rows processed will be less than the actual number of rows in your file. This is because Data Integration Platform Cloud reads a subset of rows to determine the column type to parse and write the data. If some rows have data types different than the rows sampled, they are omitted. Cleanse your data so that the data types are consistent to avoid any loss of data.
- For Spark Execution:
  - Only Amazon S3 is supported as a source endpoint

  - Limited JSON support. Currently, only JSON files where each record is a valid JSON is supported. Multi-line JSON files or entire JSON files are not supported.
  - Output can be in a directory or in a file, based on input format.
  - Using the agent's JAVA_OPTS properties, you can configure Spark application properties (for example, *export JAVA_OPTS="-Dsparkexecutor.cores=2"*):
    - spark.executor.memory
    - spark executor.cores
    - spark.cores.max
    - spark.driver.cores
    - spark.driver.memory
    - spark,app.name
  - Only Spark on Big Data Cloud (OCI Classic) with basic authentication is supported, not BDC (OCI-C) with IDCS authentication.
  - For Spark on YARN, Spark 2.2+ with HDFS, YARN, and Spark should be preinstalled on the cluster.
  - For Spark on YARN, YARN REST port (default 8088) and HDFS port (default 8020) should be open and accessible from the DIPC Agent.
  - Only Spark on YARN with no authentication is supported.
  - DIPC uploads a jar file containing the Data Lake copy Spark application to a path in HDFS that was specified as part of the Spark Execution Environment configuration. DIPC assumes that you have the proper permissions to copy the jar file to this path in HDFS.
  - DIPC allows you to configure up to eight additional properties when configuring a Spark Execution Environment.

Before You Add Data to Data Lake

Before you perform an Add Data to Data Lake Task, you'll need to download and configure your agents and create connections to your source and target data sources.

Make sure that you’ve done the following:

- Set up an Agent
- Create an Oracle Database Connection
- Create an Oracle Object Storage Classic Connection
- Create an Autonomous Data Warehouse Cloud (ADWC) Connection
- Create an Amazon S3 Connection

You must also create a Data Lake using the Data Lake Configuration Element. Optionally, you can create an Execution Environment to execute the Add Data to Data Lake job using Spark on Big Data Cloud or YARN.

Create a Data Lake

Before you add data to a Data Lake, you must create one using the Data Lake Configuration Element.

To create a Data Lake:

1. From the Home Page Getting Started section, click **Create** in the Data Lake tile or click **Create** and select Data Lake in the **Catalog**.
2. On the Create Data Lake page, complete the General Information fields.
3. For Connection, select an existing **Oracle Object Storage Classic Connection** or create a new one.
4. For Default Data Management Settings,
   a. For **Type**, select the file format of the target file that you're adding to the Data Lake.
   b. Click **Options**, and specify the **Encoding**, **Delimiter**, **Text Qualifier**, and **Header**.
5. Click **Save**.

The Data Lake Configuration Element is stored in Data Integration Platform Cloud as a **Data Asset**. You can now use it as a target when you create an Add Data to Data Lake task.

Set Up an Execution Environment

An Execution Environment enables you to run an Add Data to Data Lake task using Spark Execution on Big Data Cloud or YARN.

To create an Execution Environment:

1. From the Home page Getting Started section, click **Create** in the Execution Environment tile or click **Create** and select **Execution Environment** in the Catalog.
2. On the Create Execution Environment Configuration page, complete the General Information fields.
3. For Environment Settings, depending on your selection of **Execution Environment Type** in the General Information section, provide the environment connection details.
4. Click **Save**.

After you save your Execution Environment configuration, you can find it in the Catalog, or use it when creating an Add Data to Data Lake task.

Add Data to the Data Lake

After your Data Lake is created, you can add data to it from a variety of data sources.

To add data to a data lake:

1. From the Getting Started section of the Data Integration Platform Cloud **Home** page, click **Create** from the Add Data to Data Lake tile or click **Create** and select **Create Data Lake** in the Catalog.

2. In the **Add Data to Datalake** page, complete the **General Information** fields.
3. To use a Spark Execution Environment, select **Spark Settings**, and then select an Execution Environment for **Spark Environment**.

   To learn more about creating Execution Environments, see Set Up an Execution Environment.
4. For **Source Configuration**, select your source Connection or create a new one.

   To learn more about creating Connections, see Create a Connection.
   1. If you select a relational database as your source Connection, click the drop-down and choose a **Schema**, and then select or enter the **Data Entity** (table) name.
   2. If you select a File Type Connection, the **Directory** populates with the default location specified when the Connection was created. For **File** click **Select** too choose the file. Click Options to select **Delimited**, **Parquet**, or **JSON.**
5. In the **Target Configuration** section,
   1. Select a **Data Lake** to copy your data to and enter a name for your new **Data Entity**.

      To learn more about creating a Data Lake, see Create a Data Lake.
   2. For **File Path**, enter a Directory/File path for your target file in the Data Lake.
   3. Select **Use Default Data Lake Settings** if you want to use the configuration settings set when you created the Data Lake, or deselect this option to overwrite the default settings.
   4. Select **Update ADWC Table** if you want to query the data in the Data Lake using Autonomous Data Warehouse.
6. Click **Save** to save the task, or click **Save & Execute** to save and start the task.

8

Prepare Data

The Data Preparation Task enables you to harvest data from a data source, perform a variety of transformations to organize and cleanse your data, and then write the resultant data to a new data source.

Topics:

- What is Data Preparation?
- What’s Certified for Data Preparation
- Before You Prepare Data
- Create a Data Preparation Task
- Transform Data

What is Data Preparation?

Data Preparation is the process of collecting, cleansing, organizing, and consolidating data from one or more data sources.

The Data Integration Platform Cloud task enables you to understand your data, refine it, reshape it, and then load it to a target data source. This cleansed data is then used to build integration projects. When performing a Data Preparation task, you harvest the data, define transformations to cleanse the data, and then load the resultant data to a target data source.

What’s Certified for Data Preparation

Review the supported data sources and limitations before choosing your source and target data source for the Data Preparation Task in.

Note:

- All data sources must have x86_64, the 64-bit version of x86 operating systems, with the latest update.
- You need a VPN to use an on-premises Oracle Database for Data Preparation.

Connection type	Version	OEL	RHEL	SLES	Windows	Source	Target
Oracle Database Cloud Classic	12.2	6.x	no	no	no	yes	yes

Connection type	Version	OEL	RHEL	SLES	Windows	Source	Target
Oracle Database Cloud Classic	12.1	6.x	no	no	no	yes	yes
Oracle Database Cloud Classic	11.2	6.x	no	no	no	yes	yes
Oracle Database	12.2	6.x	6.x, 7.x	11, 12	2012, 2016	yes	yes
Oracle Database	12.1	6.x	6.x, 7.x	11, 12	2012, 2016	yes	yes
Oracle Database	11.2.0.4	6.x	6.x, 7.x	11, 12	2012	yes	yes
Oracle Object Storage Classic	Latest	n/a	no	no	no	yes	yes
Flat Files	n/a	yes	yes	yes	yes	yes	yes

See Agent Certifications.

Before You Prepare Data

To create a Data Preparation Task, you must first set up default connections, download and configure your Agent, and create Connections to your source and target data sources.

Set up Default Connections

You must create and configure Default Connections to prepare data in. You must at least set your Oracle Staging Connection. This Connection is used to stage the data harvested from your source data sources before running the task.

For more information about configuring these Connections, see Set up Default Connections.

Download and configure Agents

Agents enable data exchanges between Data Integration Platform Cloud and your data sources.

For more information about downloading and configuring agents, see Set up an Agent.

Create Connections

Connections enable you to enter the connectivity details for your source or target data sources, so you can use them for different tasks when needed.

For more information about creating Connections, see Create a Connection.

Create a Data Preparation Task

Create a Data Preparation Task to harvest data from a data source, organize and cleanse the data, and then write the resultant data to a new data source.

To create a Data Preparation Task:

1. On the Catalog page, select **Data Preparation** from the **Create** menu.
2. In the General Information section, enter a name and description.

   > **Note:**
   >
   > You'll use the value you provide in the **Name** field to identify it in the Catalog. The **Identifier** is auto-generated based on what you enter into the **Name** field and is used when searching for this Task in menus and other collections. You can edit the Identifier later, but it can only include capital letters, numbers, and underscores (_).

3. In the Source Configuration section, select a source Connection:
   - If your Connection is a database, select the appropriate **Schema** and **Data Entity** for this Task
   - If your Connection is a file, the default **Directory** selected when you created the Connection is displayed. Click **Select** to choose a subdirectory of the default directory.

     For File, click **Select** to choose the appropriate file for this Task. Click **Options** to specify how Data Integration Platform Cloud reads your data.
4. In the Target Configuration section, select a target Connection:
   - If your Connection is a database, select the appropriate target **Schema** for this Task, and then enter a name for your new table in the **Data Entity** field.
   - If your Connection is a file, the default **Directory** selected when you created the Connection is selected. Click **Select** to choose a subdirectory of the default directory.

     For File, click **Select** to choose the appropriate file for this Task. Click **Options** to specify how Data Integration Platform Cloud writes your data.
5. Click **Save** to save your Task.
6. Click **Transform** to save the configuration and then use the Transformation Creation Tool to select the transforms to perform on your data.

It may take a few minutes for Data Integration Platform Cloud to harvest your data before it's ready for you to transform. You can monitor the progress on the **Monitor** page.

Transform Data

After you create your Data Preparation Task, the next step is to create your transforms. It may take a few minutes for your data to become available.

Let's review the different areas of the Transformation Creation Tool:

- **Configuration**: Displays the source and target Connection details. Click **Edit** to return to the Data Preparation Task page and modify your Connection details.
- **Transforms**: Lists all transform activity for this Transformation session. Click **Remove** to undo specific transforms.
- **Interactive transforms area:**
  - **Metadata**: Displays the columns, column type, and sample data values harvested from the source data entity. From the **Transform Actions** menu, you can perform transforms such as **Delete**, **Rename**, and **Replace**.
  - **Data**: Displays all the data values harvested from the source data entity.
  - **Reorder Columns**: Click this icon to reorder the columns displayed in the interactive transforms area.
  - **Expand/Collapse Data Profile**: Click this icon to collapse the Data Profile pane to give yourself more room to work in the Interactive transforms area. Click the icon once more to expand the Data Profile pane to view summary information about your data.
- **Data Profile**: Displays a summary of your data, such as total number of rows and columns.
- **Column Profile**: Displays a more detailed summary of your data.

After your data becomes available, you can perform the following transforms from the **Transform Actions** menu of a selected columns.

- **Delete**: Removes the column and its data values from the source data entity.
- **Rename**: Renames the column to a name of your choice.
- **Replace**: Enables you to search and replace data values in the column with other specified values.
- **Replace with Regex**: Enables you to search and replace data values using regex patterns.
- **Extract**: Searches and extracts data values into a new column.
- **Extract with Regex**: Searches and extracts data values into a new column using regex patterns.
- **Change Case**: Modifies column values to either Lower, Upper, or Proper.
- **Null Check**: Specifies the threshold for null values within your data.
- **Null Fillup**: Replaces all null values with a specified string.
- **Force Type**: Changes the column from one type to another.
- **Merge Columns**: Combines two columns into a new column.

The Transformation Creation Tool uses the same regex pattern syntax used with Oracle Database.

See Using Regular Expressions in Database Applications.

After completing your Transformation session, click **Save** to save your changes or click **Run** to save and execute the Job.

9

Monitor Jobs

The Monitor page displays information about all the Jobs ever started in Data Integration Platform, whether they're running or stopped, already finished successfully or failed.

The Monitor page is a read-only page that gives you an aggregate overview of Jobs through its four pages: Summary for all the jobs, and then pages for these three specific types of jobs created from these tasks: Synchronize Data, Replicate Data and Data Lake Management. Each of these pages has three tiles and a table:

Agent Health

This tile reports whether capture and delivery plugins related to Oracle Data Integration Platform Cloud tasks are running on the data sources. This status is different from the status of the agents displayed on the Agents page.

Oracle Data Integration Platform Cloud has plugins on sources and targets for tasks with capture and delivery. These plugins instantiate the capture and delivery processes on the data sources, allocate port numbers, perform management and create reports. The Agent Health tile gets reports from Data Integration Platform Cloud's agents whether these plugin processes are running or not. If the plugins stop working, then jobs with capture and delivery can't be performed. This tile displays a two-color ring chart: green for Jobs with running plugins and red for stopped ones. The percentage of agents reporting running plugins is displayed at the center of the ring chart. For example, if there are four running plugins and one stopped one, then the center of the ring displays 80%. The number of running and stopped plugins are each displayed adjacent to their relative section of the chart. This chart is not clickable.

The Agent page displays the status of agents, but the Agent Health tile displays if the capture and delivery plugins are stopped or running. They don’t display the same information.

Currently the Agent Health displays information sent from plugins for the Synchronize Data and Replicate Data jobs. All other jobs don't use a plugin. Therefore, the Agent Health doesn't display a correct status for them. For now, use this tile only to review Synchronize Data and Replicate Data jobs.

Currently this tile doesn't distinguish the Replicate Data and Synchronize Data jobs and shows the sum of all information sent for both of these tasks.

The number on the Agent Health tile may be different than the total number of jobs that you have. The number displayed on this tile is equivalent to the number of plugins that you have. These plugins start on your sources and targets after you set up your agents and set up your GG_HOME directories for the Synchronize Data and Replicate Data tasks. For example, if you have two GG_HOME directories, one for source and one for target of a Synchronize Data job, then you have two plugins. So you have one job, but two plugins, and if both are running, you'll see the number 2 in the Agent Health tile with a status of Running.

Job Health

The Job Health tile located in the Monitor page of Oracle Data Integration Platform Cloud displays information for all the Jobs ever run on the server.

The Job Health tile displays a tri-color ring chart: green for running, red for failed and yellow for stopped Jobs. The percentage of running Jobs is displayed at the center of the ring chart. For example, if there are three running Jobs, one stopped and one failed Job, then the center of the ring displays 60%. The number of running, failed and stopped Jobs are each displayed adjacent to their relative section of the chart. This chart is not clickable.

For visual simplicity, Jobs with status of *Prepared, Waiting, Running* or *Successful* are all counted in the Running section of the chart, because they're neither *Stopped* nor *Failed*.

Top Duration Report

The Top Duration tile in the Monitor page of Oracle Data Integration Platform Cloud displays the top three jobs with the longest duration.

- For jobs that are currently running, duration is current time minus the job's start time.
- For jobs that ran successfully and ended without being stopped, it's the job's end time minus its start time.

Failed jobs don't display the correct duration.

- For a failed job, currently the duration is calculated as current time minus the job's start time, instead of the time that job stopped due to failure minus the job’s start time.
- For a job that ran successfully, but stopped for a duration in between, the duration is still calculated as the job's end time minus its start time.

For example:

- Job A runs for an hour (started at 2:00PM)
- Job A is stopped for 3 hours (between 3:00 and 6:00PM)
- Job A is restarted (restarted at 6:00PM)
- Job A finishes successfully after an additional hour (stops successfully at 7:00PM) The duration in this example will display five hours and you won't know that the job ran for two hours and was stopped for three hours.

The duration in this example will display five hours and you won't know that the job ran for two hours and was stopped for three hours.

Top 3 Data Report

The Top 3 Data Report tile in Oracle Data Integration Platform Cloud displays the top three Jobs with highest data volume in the past three days. This tile is available in the Synchronize Data, Replicate Data and Data Lake Management pages of the Monitor page.

Data volume is the number of operations replicated in the past three days in Mega Bytes per second.

Go to the Admin page and then click **Data Usage** to narrow down your durations and get snapshots for:

- Total Data Processed (GB)
- GB of Data Processed per Hour

Top 3 Lag Report

The Top 3 Lag Report tile in Oracle Data Integration Platform Cloud displays the top three running Jobs with the most lag. This tile is only available in the Synchronize Data and Replicate Data pages of the Monitor page.

The bar chart on this tile displays a color for each bar, with the color legend displaying the name of the corresponding Job.

Here, the lag is only for the **Start Delivery** action in the job detail page of the Synchronize Data and Replicate Data pages.

The Lag displayed in Top 3 Lag report is not the **Lag** displayed in the job detail page which represents the total log of a job. It is the **Start Delivery lag** which is an action only in Synchronize Data and Replicate Data jobs.

Job Details

The **Job Details** section of the Monitor page in Oracle Data Integration Platform Cloud displays all the Jobs ever started.

The Summary section of the Monitor page displays details for all the jobs ever started. You can go to Synchronize Data, Replicate Data and Data Lake Management pages to get a list jobs for these specific tasks.

The Jobs in each of these pages are listed in a table. Click each job to see its full details.

The **Type** of a job in the Summary page includes jobs created from all tasks including ODI Execution and Data Preparation.

The **Status** of a job can be prepared, waiting, running, stopped, success or failed.

Job Statuses

Every time you run a Oracle Data Integration Platform Cloud task, a new Job is created.

Job Status Life Cycle

When you click Run in the action menu of a task in the Catalog, you create a Job. A Job goes through many phases before it actually runs. Here is a list of all the states that a Job, or the actions it contains, may be in.

Job status	Description
Prepared	The **Prepared** state transforms the actions in a Job into a correct format, so they can be ready to run.
Waiting	The orchestration service has received a message to run a certain action, but the action is in the **Waiting** state and has not run yet. It could be, for example, that the runtime engine is overloaded at the time and can't run the action.
Running	The Job or action is running.
Successful	The Job or action has run and finished successfully.
Stopped	The Job has been manually stopped.
Failed	The execution of a Job or one or more of its actions failed.

Heartbeat Lag

The Heartbeat Lag column appears only for Replicate Data jobs. It also requires a few configuration steps to display the right information.

In a Replicate Data Task, the source agent sends periodic signals, called heartbeats, from the source to the target to monitor the lag. The source agent creates these heartbeats by sending an artificial transaction to the target every 10 seconds. These transactions contain timing information that is used for the heartbeats. The agent also creates a heartbeat table in the source and target table to keep track of the heartbeat history.

Heartbeat lag is the difference between the time the source agent sends a signal to and is captured by the target. You can find the heartbeat lag for your job on its Job Details page. By default, this column is not visible. You need to click **Customize the Table View** (the gear icon), and then select Heartbeat Lag for display.

Configure Hearbeat Lag

For the Heartbeat Lag to work, you must create a Connection with a dba user named `ggadmin`. This user must have dba privileges on both the source and target database. When you create the Replicate Data Task, use the `ggadmin` connection with your desired schema.

You can also adjust the Heartbeat interval. The default value is 10 seconds. You can change this value in the `agent.properties` file. See Set Your Agent Properties.

Example 9-1 agentHeartBeatInterval

```
# agentHeartBeatInterval
# : This is the interval in seconds at which heart-beat is sent to the
server
# Default value : 10
```

```
#agentHeartBeatInterval=10
agentHeartBeatInterval=20
```

10
Create Policies

Create Policies to schedule jobs and stay informed of job events in Data Integration Platform Cloud.

Topics:

What is a Policy?

You can use **Policies** to specify when and how often a job should run and when you should be notified when a certain condition is met.

Policies are a set of conditions for job activities. If the conditions are met, a job will execute or you'll receive a notification. You're alerted to new notifications within the Data Integration Platform Cloud console. The alert icon at the top of the page displays the number of new notifications you have.

For more information, see What is a Notification?

You create policies to schedule jobs and to stay informed about your job activities. You can receive notifications for both desired and undesired situations. For example, you can be notified when a service stops, a conflict is detected, or a certain number of inserts are applied.

What is a Notification?

A Notification is an alert that you receive in Data Integration Platform Cloud when Notification Policy conditions are met.

To view your notifications, click the **Notifications** icon, displayed as a bell, accessible from any page in your Data Integration Platform Cloud console.

To learn more about creating Notification Policies, see Create a Notification Policy.

On the Notifications page, you can:

- Sort your notifications by Severity, Policy Metric, Policy Name, or Created Date.
- Get a glimpse of what policy metric was triggered through the Policy Metric column.
- From the Action menu of a notification, select **Open Job Details** to learn why you got the notification.
- After you're done with a notification, mark it as Read or Delete it.

- From the Action menu, open the policy that caused the notification, then review or edit the policy for future notifications.
- Mark Read notifications as Unread to review them later.

Create a Notification Policy

A Notification Policy sets the conditions for which you're notified of a job event in Data Integration Platform Cloud.

To create a Notification Policy:

1. On the Policies page, click **Create**, and then select **Notification Policy**.
2. On the Create Notification Policy page, complete the General Information fields.

   The Identifier identifies this connection through a string with no spaces. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
3. For Severity, select the importance of your notification: **Low**, **Medium**, or **High**.
4. Click **Enable this policy** to activate it.
5. Under Policy Metrics, select when you should be notified, either when **All conditions are satisfied** or when **Any one condition is satisfied**, and then set the conditions:
   a. Select a metric.
   b. Select an operator.
   c. Select or specify a value.
6. Click **Save**.

After you save your policy, it's listed on the Policies page where you can edit or delete it from its **Show Context menu**.

Create a Scheduling Policy

A Scheduling Policy sets the conditions for when and how often a job should run in Data Integration Platform Cloud.

To create a Scheduling Policy:

1. On the Policies page, click **Create**, and then select **Scheduling Policy**.
2. On the Create Scheduling Policy page, complete the General Information fields.

   The Identifier identifies this connection through a string with no spaces. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
3. Click **Enable this policy** to activate it.
4. Under Scheduling Information:
   a. Select how often you want the job to run from the **Frequency** menu.
   b. Set the **Start Date** and **End Date** values.
5. Click **Save**.

After you save your policy, it's listed on the Policies page where you can edit or delete it from its **Show Context menu**.

11

Maintain Oracle Data Integration Platform Cloud

Learn maintenance techniques, troubleshooting, and frequently asked questions about Oracle Data Integration Platform Cloud.

Topics

- Maintaining Your Oracle Data Integration Platform Cloud Service Instances
- Troubleshooting Oracle Data Integration Platform Cloud

Maintain Your Oracle Data Integration Platform Cloud Services

Here, you'll learn how to find information, backup and restore, configure access rules, and delete your Data Integration Platform Cloud service instances.

- Get Information About Your Service and Service Instances
- Configure Backup and Recovery
- Delete an Oracle Data Integration Platform Cloud Service Instance

Get Information About Your Service and Service Instances

View details of all your service instances.

To view information about your Oracle Data Integration Platform Cloud service in a given identity domain:

1. Log in to Oracle Cloud.
2. Open the navigation menu in ther upper left corner, select **Platform Services**, and then **Data Integration Platform** or **Data Integration Platform Classic**.
3. In the service console for **Data Integration Platform**, you can select an instance to view detailed information.

Configure Backup and Recovery

If you decide to use backup and recovery, then you must create a Storage Container in Oracle Storage Classic.

You can perform instance backup and recovery from your Data Integration Platform Cloud service console. To perform a backup:

1. Select the service instance to perform a backup on, and then click **Administration.**
2. From the **Manage backups on this service** menu, select **Backup Now.**

> **Note:**
>
> If you restore a Data Integration Platform Cloud 17.4.3 backup on a service instance that has been patched to 17.4.5, then you will not see the Data Integration Platform Cloud console in your service menu options. This is because domain data is also included in the backup process.

Delete an Oracle Data Integration Platform Cloud Service Instance

You may find yourself needing to remove a Data Integration Platform Cloud service instance. For example, you can delete a service instance when you no longer need it. Here's how you can delete a service instance.

Only a Data Integration Platform Cloud service administrator can delete a service instance.

When you delete an Oracle Data Integration Platform Cloud instance,

- Compute resources such as block storage volumes and IP addresses are removed
- The Oracle Database Cloud Service instance isn't deleted when you delete the service instance, only the database repository and schemas are deleted. Your account remains charged for the database service instance. You probably want to retain this database service instance for use with other service instances.

  For more information, see Deleting an Oracle Database Cloud Service Instance in *Using Oracle Database Cloud Service*.
- The Oracle Storage Cloud Service container isn't deleted. Your account remains charged for the storage container. You probably want to retain this storage container for use with other service instances.

  For more information, see Deleting Containers in *Using Oracle Storage Cloud Service*.

To delete an Oracle Data Integration Platform Cloud service instance:

1. In the Oracle Data Integration Platform Cloud service console, locate the instance you want to delete.
2. From the instance's **Manage this service** menu, select **Delete**.
3. In the **Delete Instance** dialog box:
   a. Select **Force service deletion** if you want the service to force the deletion and ignore any backend script failures.
   b. Select **Force Delete without Validation** if you want to delete the instance without credential validation.
   c. Enter your administrative user name and password in their respective fields.
   d. Enter `true` or `false` in the **should backup be skipped** field.
4. Click **Delete**.

The Data Integration Platform Cloud service instance is removed from the service list.

Review Your Data Usage

Get information about the volume of data processed by your Data Integration Platform Cloud instance.

You can find the Data Usage page under the Admin area of Data Integration Platform Cloud. On this page, you select a period of time and you'll get information about the total data processed daily or how many gigabytes were processed per hour. This page is especially useful for getting information about your volume of processed data, to decide what to choose for the **Data processing per hour** field when you create an instance.

This read-only page gives you an aggregate overview of the volume of data processed through two charts and one table:

- Total Data Processed
- Daily Usage
- Details

Total Data Processed

This section displays one number which is either the total data processed in gigabytes (GBs) or the calculated average of total data processed per hour. There is a drop-down menu on this page that allows you to toggle between these two options.

Daily Usage

This section displays a bar chart with the dates on the vertical axis and the total data processed in GBs represented as a bar for each of the days listed on the horizontal axis. The chart displays the data processed in the period of time you choose.

Details

In this section, you get a calendar option to choose a date range with the fields From and To. Each row of this table displays the following information:

- **Date,** one day of the selected range, displayed in chronological order.
- Total Data Processed (Gb) for that day
- **Gb of Data Processed per Hour,** the Total Data Processed (Gb) for that day divided by 24

Troubleshoot Oracle Data Integration Platform Cloud Service

Resolve common problems and issues that may occur when you use Oracle Data Integration Platform Cloud Service.

- Troubleshoot Failed Jobs
- Perform a Manual Clean Up of Tasks

Troubleshoot Failed Jobs

The Data Integration Platform Home page displays the number of failed jobs in the Jobs section. Clicking the number enables you to view the list of failed jobs in the Jobs page, and troubleshoot it.

To troubleshoot a failed job execution:

1. In the Data Integration Platform **Home** page, go to the **Jobs** section.

   The Jobs section displays the number of failed Jobs.

2. Click the number.

   The Jobs page opens, displaying a list of the failed Jobs.

3. Click the failed Job that you want to troubleshoot.

   The Job Details page opens, showing the list of actions.

   For **“Initial Load using Datapump Action failed”**, do the following:

   a. Click the corresponding failed status icon, to review the error message.

      The error message appears as:

      “ORA-31631: privileges are required ORA-39149: cannot link privileged user to non-privileged user”

      > **Note:**
      >
      > This error occurs, when the user is not having privileges for running the datapump action.

   b. From the left navigation bar, click **Catalog**.

   c. Locate the **Task** linked to that **Failed Job** name, and open it.

   d. From the **Task Details** page, click the **Task Editor** to review the Task settings.

   e. In the **Task Editor**, identify the database user by the **Task/Job** name.

   f. Go to the **SQL Developer**, and set the appropriate database user privileges.

   g. Go back to the **Data Integration Platform**, and from the left navigation bar, click **Catalog**.

   h. Go to the **Failed Job**, and click **Run** to start a new Job execution.

      Job execution starts with a message, “Job Synchronize Data started” displayed at the top of the screen.

      Clicking the message redirects you to the Jobs page.

   i. In the **Jobs** page, click **Auto Refresh**.

Perform a Manual Clean Up of Tasks

You may need to perform the following steps to manually clean up a replication task.

1. From gghome, run ./ggsci, and then enter the following commands:

    ```
    info all
    stop replicat <replicat_name>
    stop extract <extract_name>
    dblogin useridalias <to source alias>
    unregister extract <extract_name> database
    delete extract <extract_name>
    dblogin useridalias <to target alias>
    delete replicat <replicat_name>
    ```

2. Check for the diroby directory. If files still appear in this directory, delete all the files.

    You may find that the files are cleaned up after successful delete of EXTRACT and REPLICAT, so this step may not always be necessary.

Part III

Perform Tasks on Hosted VMs

You can extend the capabilities of Data Integration Platform Cloud by accessing the

This topic only applies to Data Integration Platform Cloud Classic.

Topics:

- Replicate Data to Oracle Autonomous Data Warehouse Cloud
- Analyzing Data Streams
- Integrating Data
- Validating Data Quality

12
Replicate Data to Oracle Autonomous Data Warehouse Cloud

This topic only applies to Data Integration Platform Cloud Classic.

You can replicate data to Oracle Autonomous Data Warehouse Cloud using Oracle Data Integration Platform Cloud .

> **Note:**
>
> In order to replicate data to Oracle Autonomous Data Warehouse Cloud using Data Integration Platform Cloud (previously known as Oracle-managed or Autonomous DIPC), you must install a DIPC agent on either a Cloud Linux VM or an on-premises Linux machine.

Topics:

- About Replicating Data to Oracle Autonomous Data Warehouse Cloud
- Understand What is Supported While Replicating to Autonomous Data Warehouse Cloud
- How Do I Replicate Data to Oracle Autonomous Data Warehouse Cloud?

About Replicating Data to Oracle Autonomous Data Warehouse Cloud

You can configure an Oracle Autonomous Data Warehouse Cloud instance as a **target database** for Oracle Data Integration Platform Cloud.

Autonomous Data Warehouse Cloud is a fully-managed data warehouse designed to support all standard SQL and business intelligence tools and deliver scalable analytic query performance. Autonomous Data Warehouse Cloud provides all of the performance of the market-leading Oracle Database in a fully-managed environment that is tuned and optimized for data warehouse workloads. Some data types, SQL commands, and database features are not available in Autonomous Data Warehouse Cloud.

To setup Oracle Autonomous Data Warehouse Cloud as a target database, you must use **Oracle GoldenGate 12.3** in an Oracle Data Integration Platform Cloud instance that includes Oracle GoldenGate 12c (12.3.0.1.2). The source for the replication to Oracle Autonomous Data Warehouse Cloud can be:

- Oracle GoldenGate On Premises releases 12.3.0.1.2 and later are certified with Oracle Autonomous Data Warehouse Cloud for remote delivery using the non-integrated Replicat *only*. However, any supported release of Oracle GoldenGate for any supported database and operating system combination that can send trail data to Oracle GoldenGate for Oracle Database release 12.3.0.1.2 and later, can be used as a source system.

- Oracle Database Cloud Service on Oracle Cloud and Oracle Cloud at Customer
- Oracle Exadata Cloud Service on Oracle Cloud and Oracle Cloud at Customer

> **Note:**
>
> You cannot setup Oracle Autonomous Data Warehouse Cloud database as a source database for Oracle Data Integration Platform Cloud.

Use Case for Replicating to Oracle Autonomous Data Warehouse Cloud with Oracle Data Integration Platform Cloud

Use Oracle Data Integration Platform Cloud to replicate data to Oracle Autonomous Data Warehouse Cloud for:

- **Real-time data warehousing**: Replicate on-premises data to Oracle Autonomous Data Warehouse Cloud to set up a staging environment for downstream ETL or real-time data warehousing.
- **Operational reporting**: Replicate real-time data from multiple on-premises data sources and deliver to Oracle Autonomous Data Warehouse Cloud for creating reports.

Understand What is Supported While Replicating to Autonomous Data Warehouse Cloud

Review the supported data types and limitations before replicating data to Autonomous Data Warehouse Cloud.

Understanding Limitations

For a complete list of database initialization parameter restrictions, database features restrictions, SQL commands restrictions, and data types restrictions, see Autonomous Data Warehouse Cloud for Experienced Oracle Database Users.

The Oracle Database data types that are supported by Oracle GoldenGate can be replicated to Autonomous Data Warehouse Cloud. For a complete list of supported data types, see Details of Support for Oracle Data Types in *Using Oracle GoldenGate for Oracle Database*. The support limitations mentioned for replicating data to Oracle Database using Oracle GoldenGate also apply to replicating data to Autonomous Data Warehouse Cloud. There are additional limitations when replicating data into Autonomous Data Warehouse Cloud as listed in the following section.

Oracle GoldenGate Replicat Limitations for Autonomous Data Warehouse Cloud

Currently, only non-integrated Replicats are supported with Autonomous Data Warehouse Cloud. Integrated and parallel Replicats are not supported.

For the best compression ratio in your target tables in Autonomous Data Warehouse Cloud, Oracle recommends replicating changes (including updates and deletes) from your source systems as inserts into staging tables and using in-database batch operations to merge the changes into your target tables.

Data Type Limitations for DDL and DML Replications

The following data types are not supported while replicating data to Autonomous Data Warehouse Cloud:

- `LONG`
- `LONG RAW`
- `XMLTYPE STORE AS OBJECT RELATIONAL`
- `XMLTYPE STORE AS BINARY`
- `BFILE`
- `MEDIA`
- `SPATIAL`

The following data types are supported only when the trail file is generated through an integrated Extract.

- `ABSTRACT/USER DEFINED TYPE`
- `UROWID`
- `ANYDATA`

How Do I Replicate Data to Oracle Autonomous Data Warehouse Cloud?

You need to configure **non-integrated Replicats** to deliver data to Oracle Autonomous Data Warehouse Cloud.

Prerequisites:

You should have the following details available with you:

- Oracle Data Integration Platform Cloud instance details.
- Oracle Autonomous Data Warehouse Cloud database account details.
- Your source database with Oracle GoldenGate Extract processes configured.

Replication Procedure Overview

To deliver data to Oracle Autonomous Data Warehouse Cloud using Oracle Data Integration Platform Cloud, perform the following tasks:

- Configure Oracle Autonomous Data Warehouse Cloud for Replication.
  - Unlock the pre-created Oracle GoldenGate database user `ggadmin` in Oracle Autonomous Data Warehouse Cloud.
  - Create new application user `user_target`.
  - Create target database tables for replication.
- Obtain Oracle Autonomous Data Warehouse Cloud client credentials file.
- Configure Oracle Data Integration Platform Cloud for replication.

  - Transfer the client credentials file that you downloaded from Oracle Autonomous Data Warehouse Cloud.
  - Configure the `sqlnet.ora` file.
  - Configure the `tnsnames.ora` file.
  - Create a `useridalias` for the `ggadmin` Oracle Autonomous Data Warehouse Cloud user.
- Configure Oracle GoldenGate Manager and non-integrated Replicats to deliver data to Oracle Autonomous Data Warehouse Cloud.

Configure Oracle Autonomous Data Warehouse Cloud for Replication

In the Oracle Autonomous Data Warehouse Cloud database, complete the following tasks:

1. Oracle Autonomous Data Warehouse Cloud has a pre-existing database user created for Oracle GoldenGate called `ggadmin`. The `ggadmin` user has been granted the right set of privileges for Oracle GoldenGate Replicat to work. By default, this user is locked. To unlock the `ggadmin` user, connect to your Oracle Autonomous Data Warehouse Cloud database as the `ADMIN` user using any SQL client tool. See About Connecting to Autonomous Data Warehouse Cloud.
2. Run the `alter user` command to unlock the `ggadmin` user and set the password for it. See Creating Users with Autonomous Data Warehouse Cloud.

   ```
   alter user ggadmin identified by password account unlock;
   ```

3. Create the new application user `user_target`.

   ```
   create user user_target identified by password;
   grant create session, resource, create view, create table to
   user_target;
   ```

4. Connect to Oracle Autonomous Data Warehouse Cloud database as the `user_target` user and create schema and target tables for which DDL replication is not enabled.

Obtain Oracle Autonomous Data Warehouse Cloud Client Credentials File

To establish connection to your Oracle Autonomous Data Warehouse Cloud database, you download the client credentials file from the Oracle Autonomous Data Warehouse Cloud service console.

> **Note:**
>
> If you do not have administrator access to Oracle Autonomous Data Warehouse Cloud, you should ask your service administrator to download and provide the client credentials file to you.

1. Log into your Oracle Autonomous Data Warehouse Cloud account.
2. From the **Instance** page, click the menu option for the Oracle Autonomous Data Warehouse Cloud instance and select **Service Console**.

3. Log into the service console using the `admin` username and its associated password.
4. In the service console, click the **Administration** tab.
5. Click **Download Client Credentials**.
6. Enter a password to secure your client credentials file and click **Download**.
7. Save the client credentials file to your local system.

The client credentials file contains the following files:

- `cwallet.sso`
- `ewallet.p12`
- `keystore.jks`
- `ojdbc.properties`
- `sqlnet.ora`
- `tnsnames.ora`
- `truststore.jks`

You refer to the `sqlnet.ora` and `tnsnames.ora` files while configuring Oracle Data Integration Platform Cloud to work with Oracle Autonomous Data Warehouse Cloud.

Configure Oracle Data Integration Platform Cloud for Replication

In the Oracle Data Integration Platform Cloud instance, complete the following tasks:

1. Connect to your Oracle Data Integration Platform Cloud instance.

   ```
   ssh -i private_key opc@IP_address_of_your_instance
   ```

2. After you connect to your Oracle Data Integration Platform Cloud instance, change user to `oracle`.

   ```
   sudo su - oracle
   ```

3. Transfer the client credentials file that you downloaded from Oracle Autonomous Data Warehouse Cloud to your Oracle Data Integration Platform Cloud instance.
4. In the Oracle Data Integration Platform Cloud instance, unzip the client credentials file into a new directory. For example, `/u01/data/adwc_credentials`. This will be your key directory.
5. To configure the connection details, open your `tnsnames.ora` file from the oracle client location in the Oracle Data Integration Platform Cloud instance.

> **Note:**
>
> Because your Oracle Data Integration Platform Cloud instance is associated with an Oracle Database Cloud instance, the `tnsnames.ora` file is located in the`network/admin` directory under `/u01/app/oracle/suite/oci`.

```
cd /u01/app/oracle/suite/oci/network/admin
ls
tnsnames.ora
```

6. Edit the `tnsnames.ora` file in the Oracle Data Integration Platform Cloud instance to include the connection details that is available in the `tnsnames.ora` file in your key directory (the directory where you unzipped the client credentials file downloaded from Oracle Autonomous Data Warehouse Cloud).

```
Sample Connection String
connection_name = (description= (address=(protocol=tcps)(port=TNS
Service port)(host=ADWC_IP))
(connect_data=(service_name=ADWC_Database_Name
(security=(ssl_server_cert_dn="ADWC SSL certification")))
```

> **Note:**
>
> The `tnsnames.ora` file provided with the client credentials file contains three database service names identifiable as:
>
> ```
> ADWC_Database_Name_low
> ADWC_Database_Name_medium
> ADWC_Database_Name_high
> ```
>
> For Oracle GoldenGate replication, use *`ADWC_Database_Name`*`_low`. See Predefined Database Service Names for Autonomous Data Warehouse Cloud

7. To configure the wallet, create a `sqlnet.ora` file in the oracle client location in the Oracle Data Integration Platform Cloud instance.

```
cd /u01/app/oracle/suite/oci/network/admin
ls
sqlnet.ora tnsnames.ora
```

8. Edit this `sqlnet.ora` file to include your key directory.

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
(DIRECTORY="/u01/data/adwc_credentials")))
SSL_SERVER_DN_MATCH=yes
```

9. To create a `useridalias`, start `GGSCI`.

   ```
   cd $GGHOME
   ./ggsci
   ```

10. Create the Oracle GoldenGate wallet and add the `useridalias` to the credential store.

    ```
    GGSCI add credentialstore
    GGSCI Alter credentialstore ADD USER ggadmin@connection_name PASSWORD
    password alias ggadmin_alias
    GGSCI DBLOGIN USERIDALIAS ggadmin_alias
    ```

Configure Oracle GoldenGate Manager and Classic Replicat to deliver to Oracle Autonomous Data Warehouse Cloud

1. If you are not already connected to your Oracle Data Integration Platform Cloud instance, connect using the `ssh` command.

   ```
   ssh -i private_key opc@IP_address_of_your_host
   ```

2. After you connect to your Oracle Data Integration Platform Cloud instance, change user to `oracle`.

   ```
   sudo su - oracle
   ```

3. From your Oracle Data Integration Platform Cloud instance, test the connection to your Oracle Autonomous Data Warehouse Cloud instance using `sqlplus`.

   ```
   sqlplus ggadmin/password@connection_name
   ```

4. After you are connected to the Oracle Autonomous Data Warehouse Cloud database in `sqlplus` from your Oracle Data Integration Platform Cloud instance, ensure that the target replication tables are available.
5. Before you run `GGSCI`, you must source the setup file:

   ```
   $ source ~/.ggsetup
   ```

6. Find out the full path to `GGSCI`:

   ```
   $ which ggsci
     alias ggsci='/u01/app/oracle/suite/gghome/ggsci'
                  /u01/app/oracle/suite/gghome/ggsci
   ```

7. Connect to `GGSCI`.

   ```
   cd $GGHOME
   ./ggsci
   ```

8. To configure, Oracle GoldenGate manager, open the `mgr` parameter file to edit it.

   ```
   GGSCI edit param mgr
   ```

9. Ensure that the manager parameter file has the following information:

```
PORT port_number
Dynamicportlist port_1-port_n
ACCESSRULE, PROG COLLECTOR, IPADDR IP_Address_GGCS_host, ALLOW
PURGEOLDEXTRACTS path_to_the_trail_file, USECHECKPOINTS, MINKEEPHOURS n
hours MINKEEPFILES n files
AUTORESTART ER *, RETRIES n, WAITMINUTES n, RESETMINUTES n
```

10. Add `GGSCHEMA ggadmin` to your `GLOBALS` file.

11. Stop and start the manager and confirm that it started.

```
GGSCI stop mgr
GGSCI start mgr
GGSCI info mgr
```

12. Configure your Replicat file. A sample Replicat file is listed here. For a complete list of Replicat file parameters, see Oracle GoldenGate Parameters in *Oracle GoldenGate Reference*.

```
replicat rep_name
useridalias ggadmin_alias
discardfile ./dirrpt/rep_name.dsc,purge
REPORTCOUNT EVERY 5000 RECORDS
map SourceSchema.SourceTable, target TargetSchema.TargetTable [filter
clauses];
```

13. In `GGSCI`, login to the database using `useridalias` and add the Replicat.

```
add replicat rep_name,exttrail ./dirdat/trail_file_name
```

> **Note:**
>
> Oracle Autonomous Data Warehouse Cloud times out and disconnects Replicat if it is idle for more than 60 minutes. When Replicat tries to apply changes (when it gets new changes) after being idle, it encounters a database error and abends. You must restart the Replicat unless you have configured `AUTORESTART`.

You have successfully configured a classic Replicat. You can now start the Replicat process and replicate data to Oracle Autonomous Data Warehouse Cloud.

13 Analyzing Data Streams

This topic only applies to Data Integration Platform Cloud Classic.

With Stream Analytics, you can analyze complex event data streams that Data Integration Platform Cloud consumes using sophisticated correlation patterns, enrichment, and machine learning to provide insights and real-time business decisions.

Stream Analytics is available only for user-managed Data Integration Platform Cloud instances.

Topics:

- Planning Your Stream Analytics Installation
- Provisioning Oracle Event Hub and Big Data Cloud Instances
- Installing Oracle Stream Analytics
- Configuring Stream Analytics
- Upgrading Oracle Stream Analytics
- Administering Stream Analytics
- Working with Stream Analytics
- Working with Patterns
- Understanding Expression Builder Functions

Planning Your Stream Analytics Installation

This topic applies only to Oracle user-managed services.

Before you begin with Stream Analytics, make sure you meet these requirements.

Download the following required software to your Data Integration Platform Cloud VM:

- JDK 8 Update 131+
- Spark version 2.2.1 with Hadoop 2.7 or higher
- OSA-18.1.0.0.1.zip

Ensure that you have Google Chrome version 60+ downloaded on your local machine.

It’s assumed that you have an Oracle Cloud account with Database Cloud Service, Data Integration Platform Cloud, and Storage instances provisioned. If not, see Create Instances for Data Integration Platform Cloud

Provisioning Oracle Event Hub and Big Data Cloud Instances

This topic applies only to Oracle user-managed services.

Using Oracle Stream Analytics with Oracle Data Integration Platform Cloud requires Oracle Event Hub and Oracle Big Data Cloud instances.

Complete the steps in the following sections to provision the necessary service instances needed to use Oracle Stream Analytics with Data Integration Platform Cloud.

Topics:

- Provisioning an Oracle Event Hub — Dedicated Instance
- Provisioning an Oracle Event Hub Cloud Instance
- Provisioning an Oracle Big Data Cloud Instance

Provisioning an Oracle Event Hub — Dedicated Instance

This topic applies only to Oracle user-managed services.

Complete these steps to provision an Oracle Event Hub — Dedicated cloud instance.

To provision an Oracle Event Hub — Dedicated cloud instance:

1. Log in to Oracle Cloud My Services Dashboard.
2. Click **Create Instance**, then click **All Services** and select **Event Hub — Dedicated**.
3. On the Instance page of the Create New Instance wizard, enter an **Instance Name**, **Description**, and **Notification Email**, and then click **Next**.
4. On the Details page, complete the following fields, and then click **Next**:
   a. Select **Basic** or **Recommended** for **Deployment Type**.
   b. Click **Edit** for **SSH Public Key** and download new keys to your local machine.
   c. Select **Enable REST Access**, and then enter the Username and Password credentials for REST Proxy Acesss.
5. Confirm your instance details and click **Finish**.
6. After your Event Hub — Dedicated instance has been provisioned successfully, click **Manage this Service** and select **Access Rules**.
7. Go to **Actions** and select **Enable all rules**.

Provisioning an Oracle Event Hub Cloud Instance

This topic applies only to Oracle user-managed services.

Complete these steps to provision an Oracle Event Hub Cloud instance, also referred to as your Kafka topic.

To provision an Oracle Event Hub Cloud instance:

1. Log in to Oracle Cloud My Services Dashboard.
2. Click **Create Instance**, then click **All Services** and select **Event Hub**.
3. On the Instance page of the Create New Instance wizard, enter an **Instance Name**, **Description**, and **Notification Email**.
4. Select your Oracle Event Hub — Dedicated instance from the **Hosted On** menu.
5. Enter the **Number of Partitions** and **Retention Period** in hours, and then click **Next**.
6. Confirm your instance details and click **Create**.

Provisioning an Oracle Big Data Cloud Instance

This topic applies only to Oracle user-managed services.

Complete these steps to provision an Oracle Big Data Cloud instance.

To provision an Oracle Big Data Cloud instance:

1. Log in to Oracle Cloud My Services Dashboard.
2. Click **Create Instance**, then click **All Services** and select **Big Data Cloud**.
3. On the Instance page of the Create New Instance wizard, enter an **Instance Name**, **Description**, and **Notification Email**, and then click **Next**.
4. On the Details page, complete the following fields, and then click **Next**:
   a. Select **Basic** or **Full** for Deployment Profile.
   b. Enter 3 or greater for **Number of Nodes**.
   c. Select **OC2m** for **Compute Shape**.
   d. Select **2.1** for **Spark Version**.
   e. Click **Edit** for **SSH Public Key** and download the keys to your local machine.
   f. Enter the **Administrative User** and **Password** details.
   g. For Event Hub Cloud Service, select the instance you created previously.
   h. Enter the **Cloud Storage Container**, **Username**, and **Password**.
5. Confirm your instance details and click **Create**.
6. After your Big Data Cloud instance has been provisioned successfully, click **Manage this service**, and then select **Access Rules**.
7. Click **Actions**, and select **Enable all rules**.
8. Click **Create Rule** and create each of the following rules:

Rule Name	Source	Destination	Destination Port	Protocol
Yarn_Web_UI	PUBLIC-INTERNET	bdcsce_Master	8088	TCP
Hadoop_Web_UI	PUBLIC-INTERNET	bdcsce_Master	50070	TCP
Access_to_HDFS	PUBLIC-INTERNET	bdcsce_Master	50075	TCP
Access_to_HDFS_CopyFile	PUBLIC-INTERNET	bdcsce_Master	8020	TCP
Yarn_RM_Rule	PUBLIC-INTERNET	bdcsce_Master	8032	TCP
Access_to_HDFS_DataNode	PUBLIC-INTERNET	bdcsce_Master	50010	TCP
Access_to_Zookeeper	PUBLIC-INTERNET	bdcsce_Master	2181	TCP
Access_to_Kafka_Broker	PUBLIC-INTERNET	bdcsce_Master	9092	TCP
Access_to_Kafka_Server	PUBLIC-INTERNET	bdcsce_Master	6667	TCP

Installing Oracle Stream Analytics

This topic applies only to Oracle user-managed services.

After you ensure that you have procured the required software and hardware and completed the procedures to set up the required environment for installation, use the following steps to install Oracle Stream Analytics 18.1.0.0.1:

1. Create a directory, for example, `spark-downloads`, and download Apache Spark into the newly created folder as specified below:
   - Spark release: `2.2.1`
   - Package type: `Pre-built for Apache Hadoop 2.7 and later`
   - Download Spark: `spark-2.2.1-bin-hadoop2.7.tgz`
2. Extract the Spark archive to a local directory.

   You can see a subfolder, `spark-2.2.1-bin-hadoop2.7`.
3. Create a new directory, for example, `OSA-downloads` and download `OSA-18.1.0.0.1.zip` from Oracle eDelivery and extract it into the newly created folder.

   You can find the `OSA-18.1.0.0.1-README.txt` file in the `OSA-18.1.0.0.1` zip file.
4. Review the file `OSA-18.1.0.0.1-README.txt`.
5. Set the environment variables:
   a. Set the `SPARK_HOME` environment variable in the `OSA-18.1.0.0.0/osa-base/etc/osa-env.sh` file to point to the directory where you have extracted the Spark archive. For example:

```
SPARK_HOME=/products/spark-downloads/spark-2.2.1-bin-hadoop2.7
```

a. Set the `JAVA_HOME` environment variable in the `OSA-18.1.0.0.0/osa-base/etc/osa-env.sh` file to point to the directory where you have extracted the JDK archive. For example:

```
JAVA_HOME=/products/java-downloads/jdk1.8.0_131
```

6. Configure your data source in `OSA-18.1.0.0.1/osa-base/etc/jetty-osa-datasource.xml` as per instructions in Initializing Metadata Store.

   a. Uncomment and edit one of the two Data source configurations, either for Oracle Database or MySQL depending on the database you want to use as metadata store. The uncommented fragment for Oracle database is shown below:

```
<New id="osads" class="org.eclipse.jetty.plus.jndi.Resource">
     <Arg>
       <Ref refid="wac"/>
     </Arg>
     <Arg>jdbc/OSADataSource</Arg>
     <Arg>
       <New class="oracle.jdbc.pool.OracleDataSource">
         <Set name="URL">jdbc:oracle:thin:@myhost.example.com:
1521:OSADB</Set>
         <Set name="User">{OSA_USER}</Set>
         <Set name="Password">{OSA_USER_PWD}</Set>
         <Set name="connectionCachingEnabled">true</Set>
         <Set name="connectionCacheProperties">
           <New class="java.util.Properties">
             <Call name="setProperty"><Arg>MinLimit</Arg><Arg>1</
Arg></Call>
             <Call name="setProperty"><Arg>MaxLimit</Arg><Arg>15</
Arg></Call>
             <Call name="setProperty"><Arg><InitialLimit</
Arg><Arg>1</Arg></Call>
           </New>
         </Set>
       </New>
     </Arg>
</New>
```

   b. Change the database host, port, SID, Oracle Stream Analytics schema user name and password fields marked in bold in the code in step 6a.

Configuring Stream Analytics

This topic applies only to Oracle user-managed services.

Now that you have Stream Analytics installed, there are a few post-installation tasks that you need to complete before you use Stream Analytics with Data Integration Platform Cloud.

Topics:

- Initializing Metadata Store
- Changing the Admin Password
- Setting up Runtime for Oracle Stream Analytics Server

Initializing Metadata Store

This topic applies only to Oracle user-managed services.

After installing Oracle Stream Analytics, you need to configure the metadata store with the database admin credential details and the version of Oracle Stream Analytics as required.

> **Note:**
>
> If you do not have the database admin credentials, ask your database administrator to create a Oracle Stream Analytics database user and initialize the content under the user by using the SQL scripts available in the `OSA-18.1.0.0.1/osa-base/sql` folder. The Oracle Stream Analytics database username must match the one configured in `jetty-osa-datasource.xml`.

To initialize the metadata store, you need database admin credentials with sysdba privileges:

1. Change directory to `OSA-18.1.0.0.1/osa-base/bin`.
2. Execute the following command:

   ```
   ./start-osa.sh dbroot=<root user> dbroot_password=<db root password>
   ```

   > **Note:**
   >
   > Ensure that you replace the database root username and password as required.

   The following console messages indicates that the Oracle Stream Analytics schema is created and the metadata store is successfully initialized:

   ```
   OSA DB user created: <User>
   OSA schema version: 18.1.0.0.1
   ```

> **Note:**
>
> If you don't see the above messages, check the `OSA-18.1.0.0.1/osa-base/logs` folder to identify the cause and potential solution.

3. Press enter on the console and run the `OSA-18.1.0.0.1/osa-base/bin/stop-osa.sh` file to complete the metadata store initialization.

Changing the Admin Password

This topic applies only to Oracle user-managed services.

Oracle Stream Analytics comes with a default user, `osaadmin`. You must create your own password to login by using that user ID.

Use the following steps to change the password:

1. Change directory to top-level folder `OSA-18.1.0.0.1` and execute the following command:

   ```
   java -cp ./lib/jetty-util-9.4.7.v20170914.jar
   org.eclipse.jetty.util.security.Password osaadmin <your new password>
   ```

   You can see the following codes in the console:

   ```
   2018-03-01 15:37:55.800:INFO::main: Logging initialized @117ms to
   org.eclipse.jetty.util.log.StdErrLog
   alphago
   OBF:1u9d1uv81uha1u9r1ugg1uuy1ua5
   MD5:34d0a556209df571d311b3f41c8200f3
   CRYPT:osX/8jafUvLwA
   ```

2. Make a note of the obfuscated password string, marked in bold in the code in step 1, and copy it to a notepad.
3. Connect to the Oracle Stream Analytics metadata store database by using your SQL tool such as SQL Plus or SQL Developer.

   The credentials must match the ones provided in the `OSA-18.1.0.0.1/osa-base/etc/jetty-osa-datasource.xml` file.
4. Update the `osa_users` table by using the command below and replace the obfuscated password with the string that you copied in step 2:

   ```
   update osa_users set pwd='<CopiedObfuscatedPasswordString>' where
   username='osaadmin'
   ```

> **Note:**
>
> This on-premise version of Oracle Stream Analytics doesn't support role based access. All users have admin privileges. To create additional users with obfuscated passwords, see Adding Users.

Setting up Runtime for Oracle Stream Analytics Server

This topic applies only to Oracle user-managed services.

Before you start using Stream Analytics, you need to specify the runtime server, environment, and node details. You must do this procedure right after you launch Stream Analytics for the first time.

1. Change directories to OSA-18.1.0.0.0/osa-base/bin and execute the following command to start the Stream Analytics server: `./start-osa.sh`

   You'll see the following message:

   ```
   OSA schema version 18.1.0.0.0
   The schema is preconfigured and current. No changes or updates are
   required.
   ```

   > **Note:**
   >
   > If you don't see this message, check the log file in `OSA-18.1.0.0.0/osa-base/logs`.

2. Set up access rules on your Data Integration Platform Cloud instance. Log in to Oracle Cloud, go to the Data Integration Platform Cloud Service Console, and then locate your Data Integration Platform Cloud instance under the **Instances** list.
3. Click **Manage this service** and select **Access Rules** for your Data Integration Platform Cloud instance.
4. Click Create Rule and create the following access rules:

Rule Name	Source	Destination	Destination Port	Protocol
Access_to_OSA_UI_1	PUBLIC-INTERNET	WLS_MS	9080	TCP
Access_to_OSA_UI_2	PUBLIC-INTERNET	WLS_ADMIN	9080	TCP
Access_to_OSA_UI_3	PUBLIC-INTERNET	WLS_ADMIN_HOST	9080	TCP

5. In your Chrome browser's address bar, enter `<Public IP of your Data Integration Platform Cloud instance>:9080/osa` to access OSA's login page.

   > **Note:**
   >
   > You can find your Data Integration Platform Cloud instance's public IP on its service instance detail page.

6. Enter the username and password credentials for the `osaadmin` user.

> **Note:**
>
> The password is a plain-text password.

7. After you're logged in, click **System Settings** and complete the fields as follows, and then click **Save**:
   a. Enter the host and port where Kafka ZooKeeper is running in the **Kafka ZooKeeper Connection** field.

      This is required to see live results while you design your pipeline.
   b. Enter the Hadoop cluster public IP where YARN resource manager is running in the **YARN Resource Manager URL field**.

      (Check `<Resource Manager public IP>:8088/cluster`.)
   c. Select **HDFS** for **Storage**.
   d. Enter the `<HDFS cluster public IP where Namenode is running>`, plus a root folder.

      (Check `<HDFS Hadoop cluster public IP>:50070`. If the root folder does not exist, it will automatically be created but the user specified in the Hadoop authentication below must have write permissions.)
   e. Select **Simple** for Hadoop Authentication.
   f. Enter the username of a user who has write permissions for the folder specified in `webhdfs://URL` in the **Username** field.

Upgrading Oracle Stream Analytics

This topic applies only to Oracle user-managed services.

If you have an existing Oracle Stream Analytics 18.1.0.0.0 installation, use the following steps to upgrade to Oracle Stream Analytics 18.1.0.0.1:

Metadata Upgrade

Use these steps if there is no additional configuration apart from the data source configuration and environment variables:

1. Backup the existing Oracle Stream Analytics 18.1.0.0.0 metadata database, for example, use Oracle Database Backup tool.
2. Stop Oracle Stream Analytics 18.1.0.0.0.
3. Unzip Oracle Stream Analytics 18.1.0.0.1 installer.
4. Copy the data source configuration from existing installation to the new installation:

   ```
   cp OSA-18.1.0.0.0/osa-base/etc/jetty-osa-datasource.xml OSA-18.1.0.0.1/
   osa-base/etc/jetty-osa-datasource.xml
   ```

5. Copy the environment variable `osa-env.sh` from existing installation to the new installation:

   ```
   cp OSA-18.1.0.0.0/osa-base/etc/osa-env.sh OSA-18.1.0.0.1/osa-base/etc/
   osa-env.sh
   ```

Upgrade Using the Existing Jetty Configurations

Use these steps if you want to upgrade using the existing Jetty configurations:

1. Backup the existing Oracle Stream Analytics 18.1.0.0.0 metadata database, for example, use Oracle Database Backup tool.
2. Stop Oracle Stream Analytics 18.1.0.0.0.
3. Unzip Oracle Stream Analytics 18.1.0.0.1 installer.
4. Use the following set of commands:

   ```
   #!/bin/bash
   OSA_V18.1.0.0.0=<refers to the existing installation that you would
   like to update>
   OSA_V18.1.0.0.1=<refers to the unzipped OSA 18.1.0.0.1 installer>

   cp ${OSA_V18.1.0.0.1}/osa-base/bin/osa.installer.jar ${OSA_V18.1.0.0.0}/
   osa-base/bin/osa.installer.jar
   cp ${OSA_V18.1.0.0.1}/osa-base/extensibility-api/osa.spark-
   cql.extensibility.api.jar ${OSA_V18.1.0.0.0}/osa-base/extensibility-api/
   osa.spark-cql.extensibility.api.jar
   cp ${OSA_V18.1.0.0.1}/osa-base/lib/ext/osa.web.jetty-session.jar $
   {OSA_V18.1.0.0.0}/osa-base/lib/ext/osa.web.jetty-session.jar

   cp ${OSA_V18.1.0.0.1}/osa-base/resources/libs/wlthint3client.jar $
   {OSA_V18.1.0.0.0}/osa-base/resources/libs/wlthint3client.jar
   cp ${OSA_V18.1.0.0.1}/osa-base/resources/libs/spark-archive.jar $
   {OSA_V18.1.0.0.0}/osa-base/resources/libs/ #spark-acrhive.jar is a new
   file added in Oracle Stream Analytics 18.1.0.0.1 installer

   #comment below line if you want to keep existing logging configuration
   cp ${OSA_V18.1.0.0.1}/osa-base/resources/log4j2.xml ${OSA_V18.1.0.0.0}/
   osa-base/resources/log4j2.xml

   cp ${OSA_V18.1.0.0.1}/osa-base/resources/modules/spark-osa.jar $
   {OSA_V18.1.0.0.0}/osa-base/resources/modules/spark-osa.jar
   cp ${OSA_V18.1.0.0.1}/osa-base/sql/*.sql ${OSA_V18.1.0.0.0}/osa-base/
   sql/

   rm -f ${OSA_V18.1.0.0.0}/osa-base/start.d/logging-log4j2.ini

   cp ${OSA_V18.1.0.0.1}/osa-base/webapps/osa.web.*.war ${OSA_V18.1.0.0.0}/
   osa-base/webapps/
   cp ${OSA_V18.1.0.0.1}/osa-base/webapps/osa.web.admin.xml $
   {OSA_V18.1.0.0.0}/osa-base/webapps/
   #osa.web.admin.xml is a new file added in Oracle Stream Analytics
   ```

```
18.1.0.0.1 installer

cp ${OSA_V18.1.0.0.1}/osa-base/version.txt ${OSA_V18.1.0.0.0}/osa-base/
version.txt
```

Administering Stream Analytics

This topic applies only to Oracle user-managed services.

Administering Stream Analytics is essential to get the required results.

Topics:

- Managing Users in Stream Analytics
- Configuring Stream Analytics System Settings
- Configuring User Preferences

Managing Users in Stream Analytics

This topic applies only to Oracle user-managed services.

After installing Oracle Stream Analytics, the next important step is to set up and manage users to use the application.

In this release of Oracle Stream Analytics, user details are stored in a database. When you create Oracle Stream Analytics schema at the time of installation, the following database tables are populated with one record in each table:

- `osa_users` — table containing the users
- `osa_user_roles` — table containing the user names and their associated roles

When you execute a query to pull in all the data from the `osa_users` table, you can see the following:

```
select * from osa_users;


+----+----------+--------------------------------------+
| id | username | pwd                                  |
+----+----------+--------------------------------------+
|  1 | osaadmin | MD5:201f00b5ca5d65a1c118e5e32431514c |
+----+----------+--------------------------------------+
```

where `osaadmin` is the pre-configured user along with the encrypted password.

When you execute a query to pull in all the data from the `osa_user_roles` table, you can see the following:

```
select * from osa_user_roles;


+---------+---------+
| user_id | role_id |
+---------+---------+
|       1 |       1 |
+---------+---------+
```

where `role_id` of value `1` indicates that the user is an administrator.

Configuring Stream Analytics System Settings

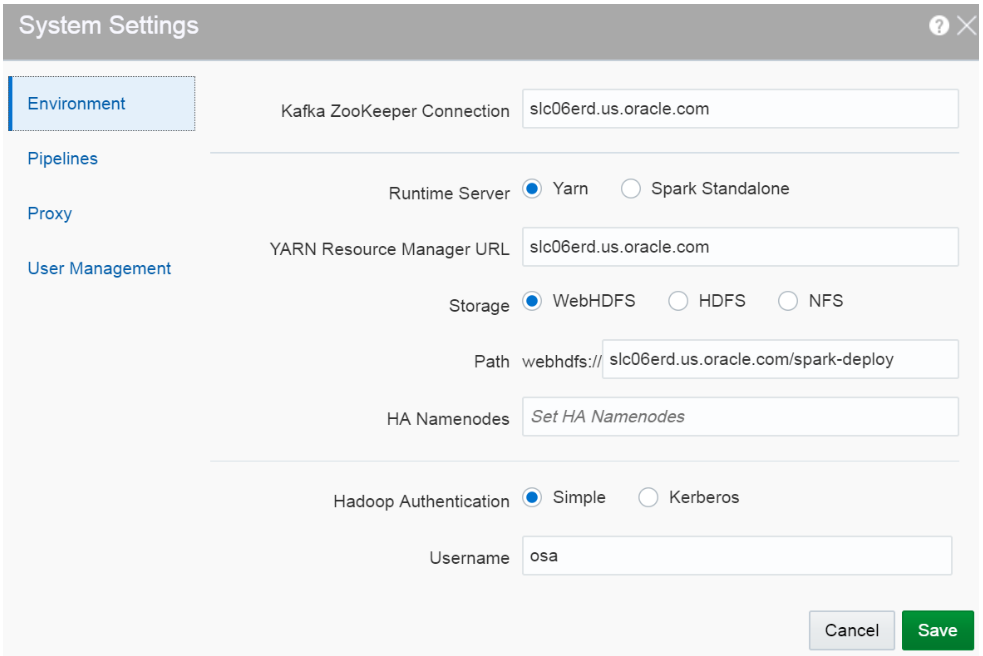
This topic applies only to Oracle user-managed services.

Only users with the *Administrator* role can set the system settings in Stream Analytics.

To set/update system settings:

1. Click the user name in the top right corner of the screen.
2. Click **System Settings**.

   The System Settings page opens.
3. Click **Environment**.
4. Specify the server names and URLs where the Kafka Zookeeper, Yarn Resource Manager, or Spark Standalone, Path, HA Namenodes are deployed and running. Hadoop authentication is an optional setting.



System Settings

Environment
Pipelines
Proxy
User Management

Kafka ZooKeeper Connection: slc06erd.us.oracle.com

Runtime Server: ● Yarn ○ Spark Standalone

YARN Resource Manager URL: slc06erd.us.oracle.com

Storage: ● WebHDFS ○ HDFS ○ NFS

Path: webhdfs:// slc06erd.us.oracle.com/spark-deploy

HA Namenodes: *Set HA Namenodes*

Hadoop Authentication: ● Simple ○ Kerberos

Username: osa

- **Kafka Zookeeper Connection** — the URL where the Zookeeper server or servers are configured, separated by comma. Kakfa is used as internal transport.
- **Runtime Server** — the runtime server you want your Stream Analytics instance to run on
- **YARN Resource Manager URL** — the URL where the YARN Resource Manager is configured, if the runtime server is Yarn
- **Spark REST URL** — Spark standalone REST URL, if the runtime server is Spark standalone. To submit an Stream Analytics pipeline to Spark, the pipeline needs to be copied to a storage location that is accessible by all Spark nodes.
- **Storage** — the type of storage for pipelines
- **Path** — the path where the storage exists. The user who accesses this folder must have the write permission to it. The folder at this path is either created by the administrator, or Stream Analytics creates the folder at runtime.
- **HA Namenodes** — If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here in the format `hostname1<port>`, `hostname2<port>` etc.
- **Hadoop Authentication** — the type of authentication configured on the Hadoop cluster. Stream Analytics supports only simple authentication.

5. Click **Pipelines**. Specify the various settings for the pipeline within Stream Analytics.
   - **Batch Duration** — the default duration of the batch for each pipeline
   - **Executor Count** — the default number of executors per pipeline
   - **Cores per Executor** — the default number of cores. A minimum value of 2 is required.
   - **Executor Memory** — the default allocated memory for each executor instance in megabytes
   - **Cores per Driver** — the default number of cores
   - **Driver Memory** — the default allocated memory per driver instance in megabytes
   - **High Availability** — toggle the default HA value as on/off for each pipeline
6. Click **Proxy**. If you set proper proxy, the back-end system will use these settings to reach the REST target. The proxy settings are also required for geo code related spatial patterns.
7. Click **User Management**. You see the list of available users here. You can add/ delete users from this screen.
8. Click **Save**.

> **Note:**
>
> On the Yarn Cluster, make sure that you do not set a high value for `yarn.nm.liveness-monitor.expiry-interval-ms` (for example, 3000 ms instead of 10 minutes). This property determines the default value for how long to wait until a node manager is considered dead. Using a high value for this property does not process the events and the events are lost.

Configuring User Preferences

This topic applies only to Oracle user-managed services.

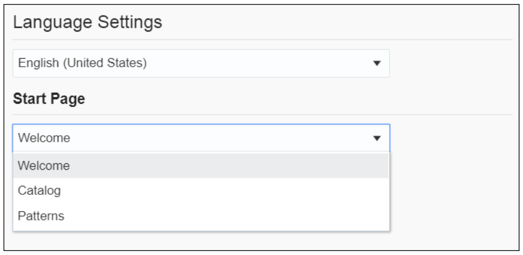
Based on the preferences the users set in this page, the characteristics of Stream Analytics vary.

To set/update user preferences:

1. Click the user name in the top right corner of the screen.
2. Click **Preferences**. The Preferences page opens.

General

Provides a set of general preferences that you can view and set according to your requirements.



Language Settings

English (United States)

Start Page

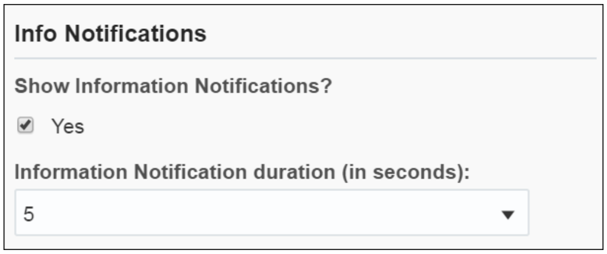
Welcome

Start Page

Select if you want the Home page, the **Catalog** page, or the **Patterns** page to appear as the **Start** Page.

Notifications

Provides a set of notifications preferences that you can view and set according to your requirements.



Info Notifications

Show Information Notifications?

☑ Yes

Information Notification duration (in seconds):

5

Show Information Notifications

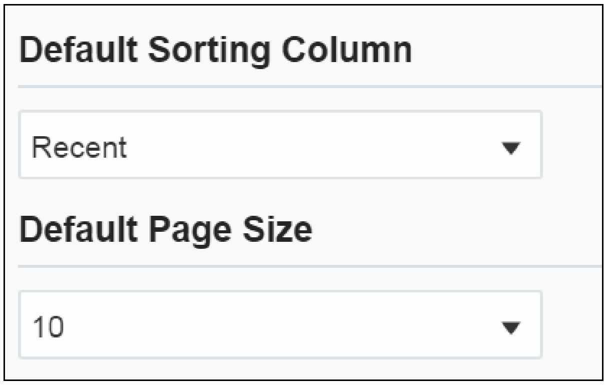
Select this option if you want the information notifications to appear in the pipeline. This option is selected by default.

Information Notification duration (in seconds)

Choose the number of seconds for which the notifications appear. The default value is 5.

Catalog

Provides a set of catalog preferences that you can view and set according to your requirements.



Default Sorting Column

Recent ▾

Default Page Size

10 ▾

Default Sorting Column

Select the column by which you want the columns to be sorted. This value will be used as the default for all columns until you change the value again.

Default Sorting Order

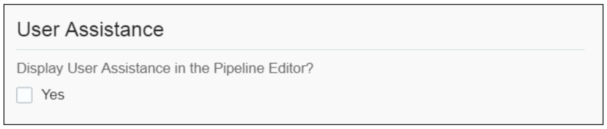
Select the order by which you want the columns to be sorted. This value will be used as the default value for all columns until you change the value again.

Default Page Size

Select the value to be used as the default page size. Based on the value selected, the number of records that appear on a page vary. This value will be used as the default for all pages until you change the value again.

Pipeline

Provides a set of pipeline preferences that you can view and set according to your requirements.



User Assistance

Display User Assistance in the Pipeline Editor?

□ Yes

Select **Yes** if you want to display the User Assistance text for the pipelines in the Pipeline Editor.

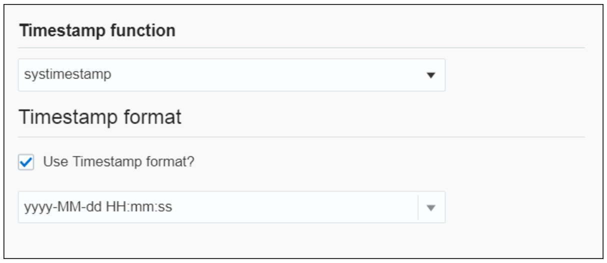
Live Output Stream

Provides a set of pipeline live output stream preferences that you can view and set according to your requirements.

Select a value that you want to be applied as the default table size for the data in Live Output Stream of a pipeline.

Timestamp

Provides a set of pipeline timestamp preferences that you can view and set according to your requirements.



Timestamp function

systimestamp

Timestamp format

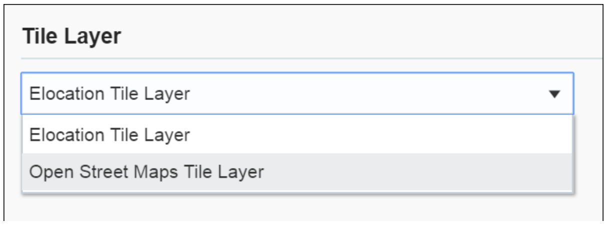
☑ Use Timestamp format?

yyyy-MM-dd HH:mm:ss

Map

Provides a set of map preferences that you can view and set according to your requirements.

Select a value that you want to be used as the default tile layer preference in the geo fences.



Tile Layer

Elocation Tile Layer

Working with Stream Analytics

This topic applies only to Oracle user-managed services.

Stream Analytics has various artifacts like connections, references, streams, targets, cubes, dashboards, predictive models, custom jars, and many more. The artifacts are the important resources using which you can create pipelines.

Topics:

- Home Page
- About the Catalog
- Typical Workflow for Administering Stream Analytics
- Creating a Connection
- Cache Configuration for Coherence

- Creating a Stream
- Creating a Reference
- Creating a Dashboard
- Creating a Cube
- Exploring a Cube
- Creating a Target
- Creating a Geo Fence
- Creating a Predictive Model
- Creating a Custom Jar
- Creating a Pipeline
- Configuring a Pipeline
- Exporting and Importing a Pipeline and Its Dependent Artifacts
- Publishing a Pipeline
- Using the Topology Viewer

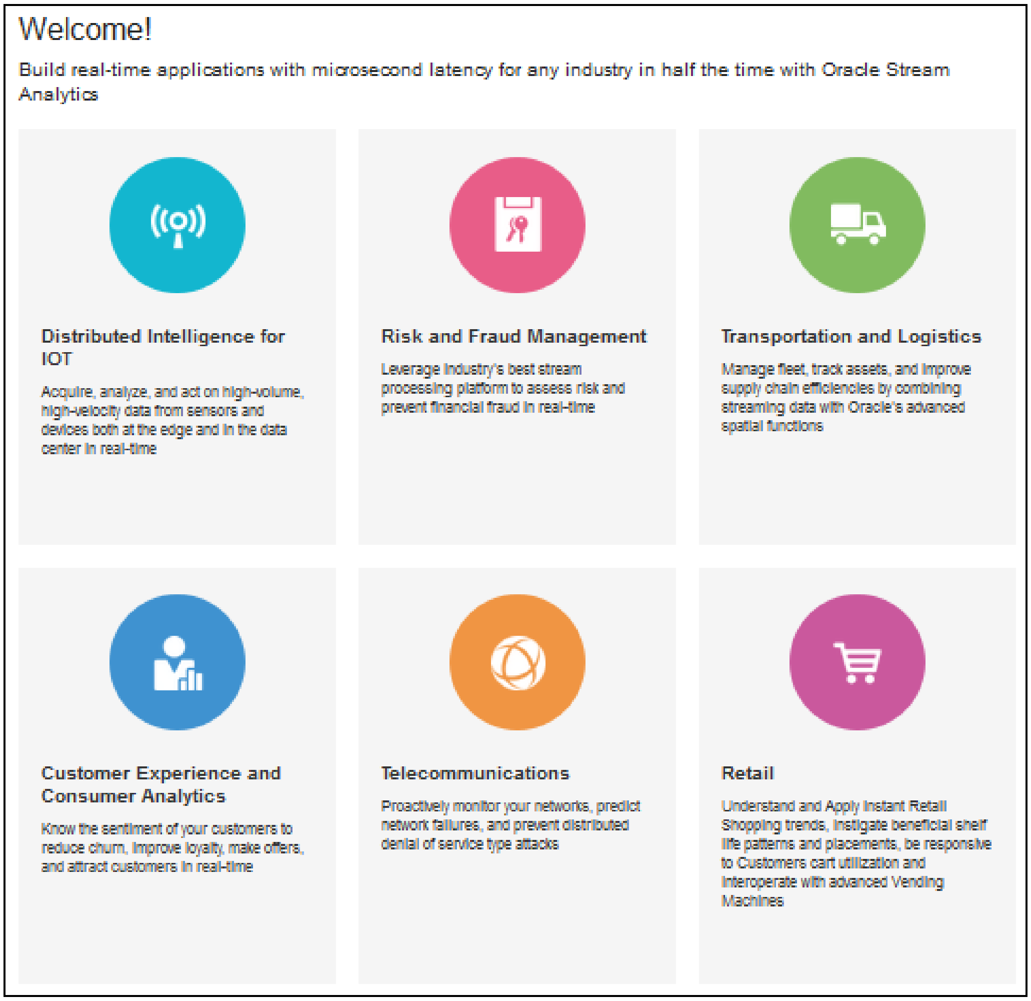
Home Page

The Home page is the first page that you see when you login to Stream Analytics. This page lists the industry verticals that Stream Analytics supports.

Each industry vertical has a tag associated with it and the tags are case-sensitive.

- *Distributed Intelligence for IOT* - Acquire, analyze, and act on high-volume, high-velocity data from sensors and devices both at the edge and in the data center in real-time. Tag for this vertical is *IOT*.
- *Risk and Fraud Management* - Leverage industry's best stream processing platform to assess risk and prevent financial fraud in real-time. Tag for this vertical is *risk*.
- *Transportation and Logistics* - Manage fleet, track assets, and improve supply chain efficiencies by combining streaming data with Oracle's advanced spatial functions. Tag for this vertical is *transportation*.
- *Customer Experience and Consumer Analytics* - Know the sentiment of your customers to reduce churn, improve loyalty, make offers, and attract customers in real-time. Tag for this vertical is *customer*.
- *Telecommunications* - Pro actively monitor your networks, predict network failures, and prevent distributed denial of service type attacks. Tag for this vertical is *telecom*.
- *Retail* — Understand and Apply instant Retail Shopping trends, instigate beneficial shelf life patterns and placements, be responsive to Customers cart utilization and interoperate with advanced Vending Machines. Tag for this vertical is *retail*.

The Home page is as shown below:



Welcome!

Build real-time applications with microsecond latency for any industry in half the time with Oracle Stream Analytics

Distributed Intelligence for IOT

Acquire, analyze, and act on high-volume, high-velocity data from sensors and devices both at the edge and in the data center in real-time

Risk and Fraud Management

Leverage industry's best stream processing platform to assess risk and prevent financial fraud in real-time

Transportation and Logistics

Manage fleet, track assets, and improve supply chain efficiencies by combining streaming data with Oracle's advanced spatial functions

Customer Experience and Consumer Analytics

Know the sentiment of your customers to reduce churn, improve loyalty, make offers, and attract customers in real-time

Telecommunications

Proactively monitor your networks, predict network failures, and prevent distributed denial of service type attacks

Retail

Understand and Apply instant Retail Shopping trends, instigate beneficial shelf life patterns and placements, be responsive to Customers cart utilization and interoperate with advanced Vending Machines

You can navigate to the Catalog or the Patterns page from the home page to get started with Stream Analytics.

About the Catalog

The Catalog page is the location where resources including pipelines, streams, references, maps, connections, targets, dashboards, predictive models, custom jars, visualizations, and cubes are listed. This is the go-to place for you to perform any tasks in Stream Analytics.

You can mark a resource as a favorite in the Catalog by clicking on the Star icon. Click the icon again to remove it from your favorites. You can also delete a resource or view its topology using the menu icon to the right of the favorite icon.

The tags applied to items in the Catalog are also listed on the screen below the left navigation pane. You can click any of these tags to display only the items with that tag in the Catalog. The tag appears at the top of the screen. Click **Clear All** at the top of the screen to clear the Catalog and display all the items.

You can include or exclude pipelines, streams, references, predictive models, geo fences, connections, targets, custom jars, visualizations, dashboards, and cubes using the **View All** link in the left panel under **Show Me**. When you click **View All**, a check mark appears beside it and all the components are displayed in the Catalog.

When you want to display or view only a few or selective items in the Catalog, deselect **View All** and select the individual components. Only the selected components will appear in the Catalog.

Typical Workflow for Administering Stream Analytics

The typical workflow lists the artifacts required to create a pipeline in Stream AnalyticsStream Analytics.

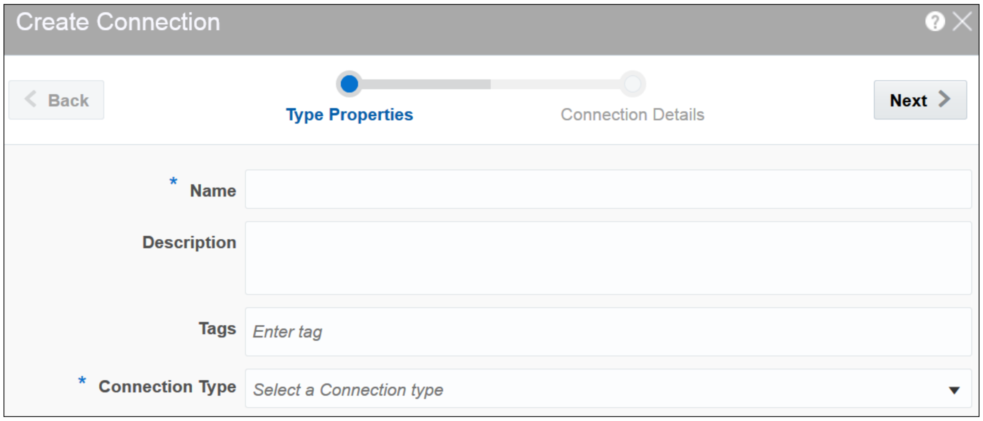
The prerequisites for a pipeline are:

- A connection is required to create a stream, except for a file stream.
- A stream is required to create a pipeline.

Creating a Connection

To create a connection:

1. Click **Catalog** in the left pane.
2. From the **Create New Item** menu, select **Connection**.
3. Provide details for the following fields on the **Type Properties** page and click **Next**:
   - **Name** — name of the connection
   - **Description** — description of the connection
   - **Tags** — tags you want to use for the connection
   - **Connection Type** — type of connection: Coherence, Database, Druid, JNDI, or Kafka



4. Enter **Connection Details** on the next screen and click **Save**.

   When the connection type is **Coherence**:

   - **Host name** — the Coherence Extend Proxy Services TCP/IP Server Socket host
   - **Port** — the Coherence Extend Proxy Services TCP/IP Server Socket port

   When the connection type is **Database**:

- **Connect using** — select the way you want to identify the database; `SID` or `Service name`
- **Service name/SID** — the details of the service name or SID
- **Host name** — the host name on which the database is running
- **Port** — the port on which the database is running. Usually it is `1521`
- **Username** — the user name with which you connect to the database
- **Password** — the password you use to login to the database

When the connection type is **Druid**, provide Zookeeper URL.

When the connection type is **JNDI**:

- **JNDI Provider** — select the JNDI service provider
- **Server Url(s)** — the server url(s) for the JNDI connection; for example: host1:port1, host2:port2
- **Username** — the user name for authenticating the JNDI connection
- **Password** — the password for the JNDI connection

When the connection type is **Kafka**, provide Zookeeper URL.

A connection with the specified details is created.

Cache Configuration for Coherence

Stream Analytics requires a special coherence cache configuration and the proxy schema, so that it can connect to the coherence.

To enrich stream data with external coherence cluster reference data, you must access external coherence cluster using extend client APIs. To access external cluster as client, you need to configure `cache-config` with `ExtendTcpCacheService` and `ExtendTcpInvocationService`.

Configure the Coherence Cluster

Make sure that you have Coherence for Java is installed.

To configure the external cluster as client:

1. Create an XML file named `cache-config.xml`.
2. Copy the following XML to the file:

```
<?xml version="1.0"?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-
cache-config
   coherence-cache-config.xsd">
        <caching-scheme-mapping>
            <cache-mapping>
                <cache-name>
                    externalcache*
                </cache-name>
                <schema-name>
```

```
                    remote
                </schema-name>
            </cahce-mapping>
        </caching-scheme-mapping>

        <caching-schemes>
            <remote-cache-scheme>
                <scheme-name>
                    remote
                </scheme-name>
                <service-name>
                    ExtendTcpCacheService
                </service-name>
                <initiator-config>
                    <tcp-initiator>
                        <remote-addresses>
                            <socket-address>
                                <address>localhost    </address>
                                <port>9099</port>
                            </socket-address>
                        </remote-addresses>
                    </tcp-initiator>
                    <outgoing-message-handler>
                        <request-timeout>5s</request-timeout>
                    </outgoing-message-handler>
                </initiator-config>
            </remote-cache-scheme>

            <remote-invocation-scheme>
                <scheme-name>extend-invocation</scheme-name>
                <service-name>ExtendTcpInvocationService</service-name>
                <initiator-config>
                    <tcp-initiator>
                        <remote-addresses>
                            <socket-address>
                                <address>localhost</address>
                                <port>9099</port>
                            </socket-address>
                        </remote-addresses>
                    </tcp-initiator>
                    <outgoing-message-handler>
                        <request-timeout>5s</request-timeout>
                    </outgoing-message-handler>
                </initiator-config>
            </remote-invocation-scheme>
        </caching-schemes>
</cache-config>
```

3. Save and close the file.
4. Test the connection to the cluster.

```
InvocationService service = 
(InvocationService) CacheFactory.getConfigurableCacheFactory().ensureSer
vice("ExtendTcpInvocationService");
```

`ensureService()` will throw exception if there is no coherence cluster available with the given host and port.

5. Create a coherence reference using a coherence connection.
6. Register the coherence as reference.

The following is the sample code to register the coherence as reference:

```
override def initialize():Unit = {
    repartition = true
    val externalEvent = EventType("externalorders",IntAttr("orderId"),
VarCharAttr("orderDesc", 20))
    val sExtSrcProps = Map(EXT_URL -> "",EXT_ENTITY_NAME ->
"externalcache")
    val jExtSrcProps = new java.util.HashMap[String,String](sExtSrcProps)
    val converter = ConverterFactory(ConverterType.COHERENCE,externalEvent)
    cc.registerEventType(externalEvent)

cc.registerRelation(externalEvent).onExternal(jExtSrcProps,ExtSourceType.CO
HERENCE,converter)
 }

def main(args: Array[String]) {
    cql = "istream(select R.orderId as orderId, R.orderStatus as
orderStatus, Ext.orderDesc as orderDesc from orders[now] as R,
externalorders as Ext where R.orderId = Ext.orderId)"
    name = "CoherenceCorrelation"
    processOrders(args)
    }
}
// EXT_URL is not used for coherence as reference , currently used for
webservice & database, so this will be set to EMPTY
//EXT_ENTITY_NAME is the cache name of the external coherence cluster
```

For the above example, coherence cache must have key as `orderId <Integer>` and value as `Map of values for orderId and orderDesc`. A sample cache similar to the following will populate:

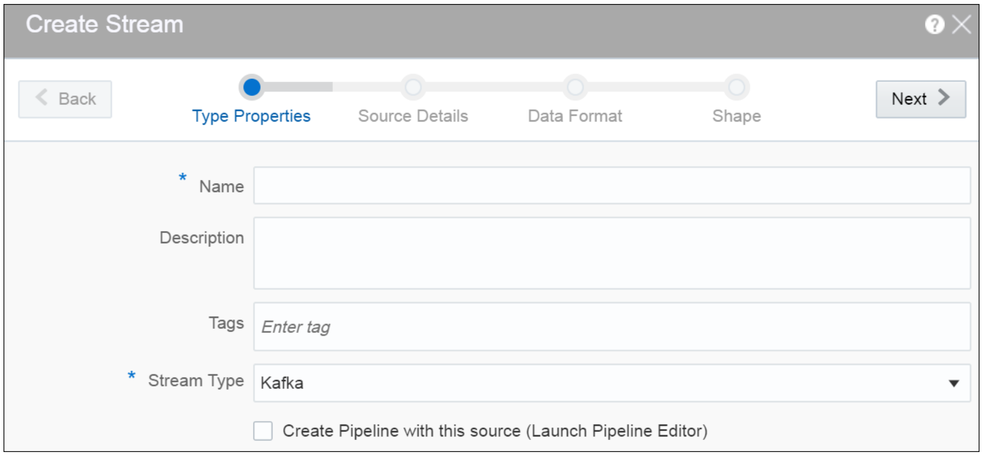
```
NamedCache cache = CacheFactory.getCache("externalcache");
Map<String,Object> order1 = new HashMap<String, Object>();
order1.put("orderId", new Integer(1));
order1.put("orderDesc", "HP Deskjet v2");
Map<String,Object> order2 = new HashMap<String, Object>();
order2.put("orderId", new Integer(2));
order2.put("orderDesc", "Oracle Database 12");
MapString,Object> order3 = new HashMap<String, Object>();
order3.put("orderId", new Integer(3));
order3.put("orderDesc", "Apple iPhone6s");
Map<String,Object> order4 = new HashMap<String, Object>();
order4.put("orderId", new Integer(4));
order4.put("orderDesc", "Logitech Mouse");
cache.put(1,order1);
cache.put(2,order2);
cache.put(3,order3);
cache.put(4,order4);
```

Creating a Stream

A stream is a source of events with a given content (shape).

To create a stream:

1. Navigate to **Catalog**.
2. Select **Stream** in the **Create New Item** menu.
3. Provide details for the following fields on the **Type Properties** page and click **Next**:
   - **Name** — name of the stream
   - **Description** — description of the stream
   - **Tags** — tags you want to use for the stream
   - **Stream Type** — select suitable stream type. Supported types are File, GoldenGate, JMS, and Kafka.



* Name

Description

Tags: *Enter tag*

* Stream Type: Kafka

□ Create Pipeline with this source (Launch Pipeline Editor)

4. Provide details for the following fields on the **Source Details** page and click **Next**:

   When the stream type is File:
   - **File Path or URL** — the location of the file that you want to upload
   - **Read whole content** — select this option if you want to read the whole content of the file
   - **Number of events per batch** — the number of events that you want to process per batch
   - **Loop** — select this option if you want to process the file in a loop
   - **Data Format** — the format of the data. The supported types are: CSV and JSON.

   When the stream type is GoldenGate:
   - **Connection** — the connection for the stream
   - **Topic name** — the topic name that receives events you want to analyze

- **Data Format** — the format of the data. The supported types are: CSV, JSON, AVRO. AVRO is a data serialization system.

When the stream type is JMS:

- **Connection** — the connection for the stream
- **Jndi name** — the Jndi that reads messages from topics, distributed topics, queues and distributed queues
- **Client ID** — the client to be used for durable subscriber
- **Message Selector** — the message selector to filter messages. If your messaging application needs to filter the messages it receives, you can use a JMS API message selector, which allows a message consumer to specify the messages it is interested in. Message selectors assign the work of filtering messages to the JMS provider rather than to the application.

  A message selector is a `String` that contains an expression. The syntax of the expression is based on a subset of the SQL92 conditional expression syntax. The message selector in the following example selects any message that has a `NewsType` property that is set to the value `'Sports'` or `'Opinion'`:

  ```
  NewsType = ’Sports’ OR NewsType = ’Opinion’
  ```

  The `createConsumer` and `createDurableSubscriber` methods allow you to specify a message selector as an argument when you create a message consumer.
- **Subscription ID** — the subscription id for durable selector
- **Data Format** — the format of the data. The supported types are: CSV, JSON, AVRO, MapMessage. MapMessage is supported only for JNDI based streams.

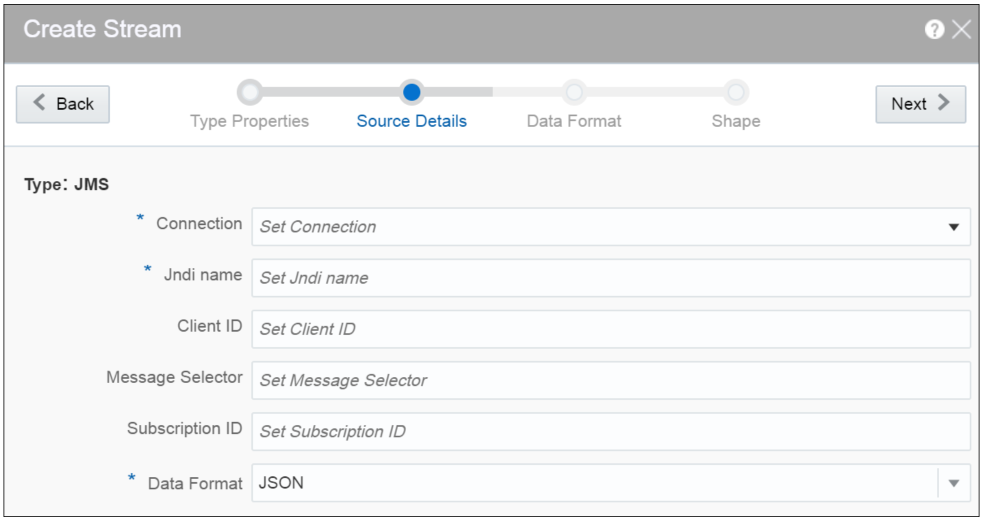
  If the data format is AVRO, you must also specify the message schema by setting `org.apache.kafka.clients.producer.ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG` and `KEY_SERIALIZER_CLASS_CONFIG` parameters as `ByteArraySerializer`.

  A MapMessage object is used to send a set of name-value pairs. The names are String objects, and the values are primitive data types in the Java programming language. The names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined.

When the stream type is Kafka:

- **Connection** — the connection for the stream
- **Topic name** — the topic name that receives events you want to analyze
- **Data Format** — the format of the data within the stream. The supported types are: CSV, JSON, AVRO.

  If the data format is AVRO, you must also specify the message schema by setting `org.apache.kafka.clients.producer.ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG` and `KEY_SERIALIZER_CLASS_CONFIG` parameters as `ByteArraySerializer`.



Type: JMS

* Connection: Set Connection

* Jndi name: Set Jndi name

Client ID: Set Client ID

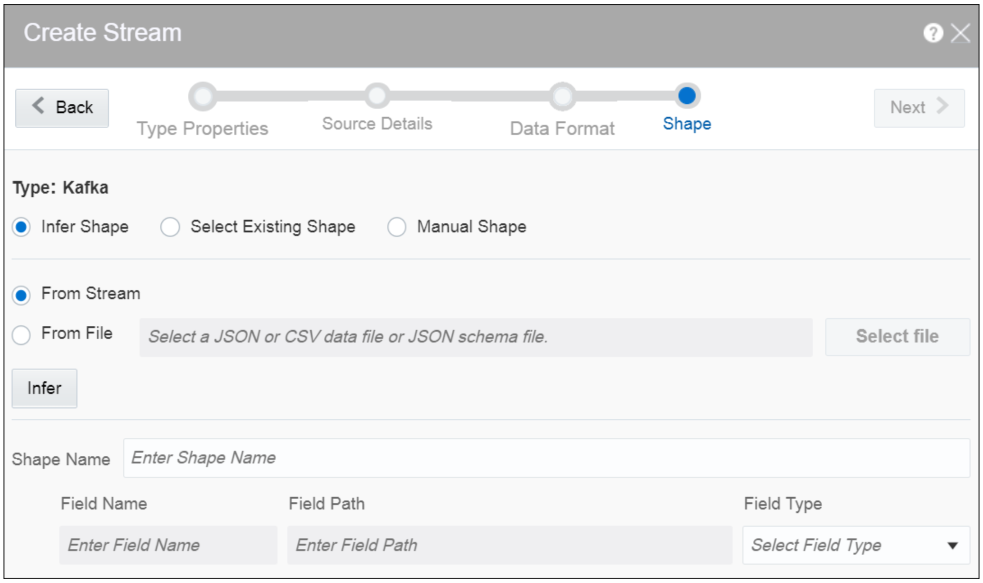
Message Selector: Set Message Selector

Subscription ID: Set Subscription ID

* Data Format: JSON

5. Select one of the mechanisms to define the shape on the **Shape** page:
   - **Infer Shape** — detects the shape automatically from the input data stream.

     You can infer the shape from Kafka, JSON schema file, or CSV message/data file. You can also save the auto detected shape and use it later.
   - **Select Existing Shape** — lets you choose one of the existing shapes from the drop-down list.
   - **Manual Shape** — populates the existing fields and also allows you to add or remove columns from the shape. You can also update the datatype of the fields.



Type: Kafka

● Infer Shape ○ Select Existing Shape ○ Manual Shape

● From Stream

○ From File: Select a JSON or CSV data file or JSON schema file.

Shape Name: Enter Shape Name

Field Name	Field Path	Field Type
Enter Field Name	Enter Field Path	Select Field Type

A stream is created with specified details.

CSV Data for Pre-defined Formats

When your data format is CSV, select a predefined format based on the variations of CSV data that differs due to the originating source of these CSV. The following table describes the CSV data for each of these predefined formats:

CSV Predefined Format	Description
`DEFAULT`	Standard comma separated format, as for `RFC4180` but allowing empty lines
`EXCEL`	Excel file format (using a comma as the value delimiter).
`INFORMIX_UNLOAD_CSV`	Default `Informix CSV UNLOAD` format used by the `UNLOAD TO file_name` operation (escaping is disabled.) This is a comma-delimited format with a LF character as the line separator. Values are not quoted and special characters are escaped with '\'. The default NULL string is "\\N".
`MYSQL`	Default `MySQL` format used by the `SELECT INTO OUTFILE` and `LOAD DATA INFILE` operations. This is a tab-delimited format with a LF character as the line separator. Values are not quoted and special characters are escaped with '\'. The default NULL string is "\\N".
`POSTGRESQL_CSV`	Default `PostgreSQL CSV` format used by the `COPY` operation. This is a comma-delimited format with a LF character as the line separator. The default NULL string is "".
`POSTGRESQL_TEXT`	Default `PostgreSQL` text format used by the `COPY` operation. This is a tab-delimited format with a LF character as the line separator. The default NULL string is "\\N".
`RFC4180`	Comma separated format as defined by `RFC4180`
`TDF`	Tab-delimited format

Capabilities of JMS Source

The capabilities of JMS Source are listed in the following table:

Capability	Description	Comments
Ability to connect to JMS Cluster	JMS consumer should be able to connect to JMS cluster and handle JMS server fail-over	
Message Format support	Map and TextMessage (JSON, CSV and AVRO)	Does not support xml and object
Message selector	JMS message selector to use to filter messages. Only messages that match the selector will produce events.	
Re-connection	Reconnect to JMS server or JMS cluster	
Read messages from queue/distributed queue		
Read messages from topic	Read messages from JMS topic. By default the subscriber is non-durable	
Support for Durable subscriber	A durable subscriber registers a durable subscription by specifying a unique identity that is retained by the JMS provider. If the consumer reconnects to JMS topic, it would read messages from where it last read.	
T3 Support	Weblogic JMS Protocol	

JMS Server Clean Up

When you create a JMS stream and select the durable subscription option (by providing client ID and subscription ID value), Stream Analytics creates the durable subscription (if not already present) when the pipeline using this stream is running. When you come out of the pipeline or unpublish the pipeline(or kill the running pipeline), the durable subscription remains on the JMS Server. It is advisable to delete the durable subscription from the JMS Server and clean up the resources, if you do not intend to publish this pipeline anymore.

Creating a Reference

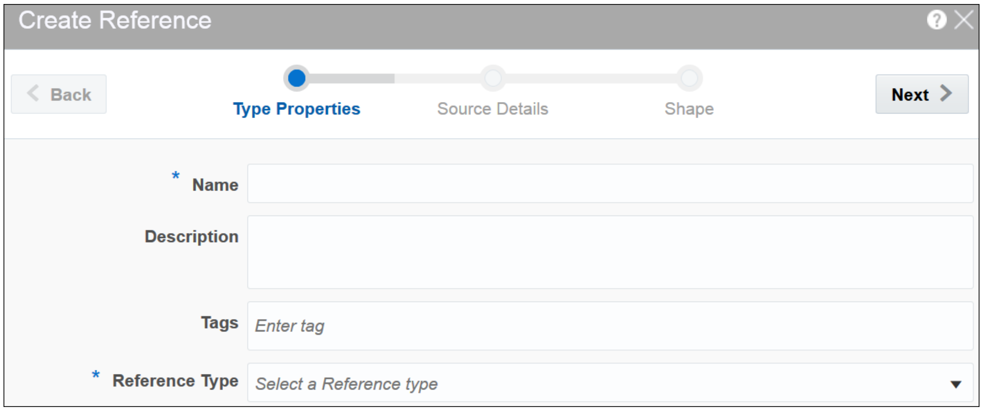
The reference defines a read-only source of reference data to enrich a stream. A stream containing a customer name could use a reference containing customer data to add the customer’s address to the stream by doing a lookup using the customer name.

A database reference is a reference to specified table in the database. With cache enabled for database reference, when the values gets pulled from database, they are maintained in coherence cache for reference from where they can be served on next request. A database reference requires a database connection.

A coherence reference can be any external cache defined in coherence cluster that can have data from an external system.

To create a reference:

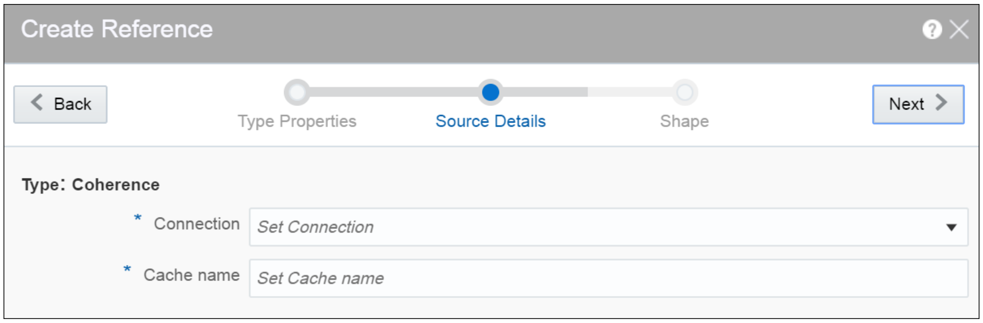
1. Navigate to **Catalog**.
2. Select **Reference** in the **Create New Item** menu.
3. Provide details for the following fields on the **Type Properties** page and click **Next**:
   - **Name** — name of the reference
   - **Description** — description of the reference
   - **Tags** — tags you want to use for the reference
   - **Reference Type** — the reference type of the reference. The supported reference types are: Coherence and Database.



4. Provide details for the following fields on the **Source Details** page and click **Next**:

   When the reference type is **Coherence**, enter or select appropriate values for:

- **Connection** — the connection for the coherence reference



Type: Coherence

* Connection Set Connection

* Cache name Set Cache name

- **Cache name** — the name of the cache to enable caching for better performance at the cost of higher memory usage of the Spark applications. Caching is supported only for single equality join condition. When you update the cache, the application will receive the updated data very quickly.

Coherence reference has data in key-value pairs. Key is object type and value is `Map<String,Object>`. `Map<String,Object>` is map of attribute names and values, attributes list should match with external event type. In this release, only external schema for key and value s supported.

When the reference type is **Database Table**, enter or select appropriate values for:

- **Connection** — the connection for the database reference
- **Enable Caching** — select this option if you want to enable caching
- **Expiry Delay** — the amount of time from last update that entries will be kept by the cache before being marked as expired. Any attempt to read an expired entry will result in a reloading of the entry from the configured cache store. This field is enabled only when caching is enabled.

5. Provide details for the following fields on the **Shape** page and click **Save**:

When the reference type is **Coherence**:

- **Select Existing Shape** — select a shape that you want to use for the reference

> **NOT_SUPPORTED:**
>
> Ensure that you do not use any of the CQL reserved words as the column names. If you use the reserved keywords, you cannot deploy the pipeline.

- **Manual Shape** — select this option if you want to define your own shape

> **Note:**
>
> When you load coherence data, ensure that you include precision and scale for number type. Only when these values are specified, the join works. For example,
>
> ```
> NamedCache cache  =
> CacheFactory.getCache("externalcachetimestamp");
>
>         java.math.BigDecimal big10 = new
> java.math.BigDecimal("10",new
> MathContext(58)).setScale(56, RoundingMode.HALF_UP);
>
>         Map<String,Object> order1 = new HashMap<String,
> Object>();
> order1.put("strValue", "Test");
> order1.put("intervalValue", "+000000002 03:04:11.330000000");
>         order1.put("orderTag", big10);
>
>         cache.put(big10,order1);
> ```

When the reference type is **Database Table**:

- **Shape Name** — select a shape that you want to use for the reference

When the datatype of the table data is not supported, the table columns do not have auto generated datatype. Only the following datatypes are supported:

- `numeric`
- `interval day to second`
- `text`
- `timestamp` (without timezone)
- `date time` (without timezone)

> **Note:**
>
> The `date` column cannot be mapped to `timestamp`. This is a limitation in the current release.

A reference is created with the specified details.

Limitations of Coherence as Reference

With coherence as reference, there are a few limitations:

- You cannot test the connection
- You need to specify the cache name manually
- Only equal operator is allowed while establishing a correlation with coherence reference

- You must use manual shape

Creating a Dashboard

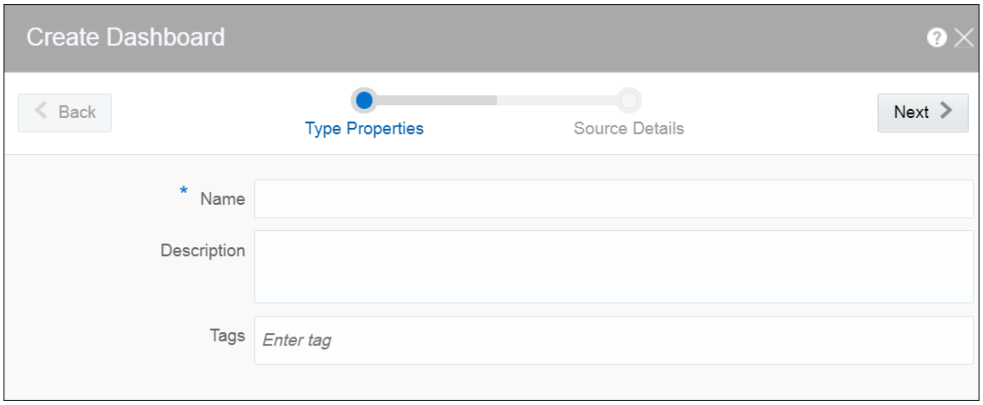
Dashboard is a visualization tool that helps you look at and analyze the data related to a pipeline based on various metrics like visualizations. A dashboard can have visualizations created out of cubes as well.

Dashboard is an analytics feature. You can create dashboards in Stream Analytics to have a quick view at the metrics.

To create a dashboard:

1. Go to the **Catalog**.
2. Select **Dashboard** in the **Create New Item** menu.

   The Create Dashboard screen appears.



* Name

Description

Tags: Enter tag

3. Provide suitable details for the following fields:
   - **Name** — enter a name for the dashboard. this is a mandatory field.
   - **Description** — enter a suitable description for the dashboard. This is an optional field.
   - **Tags** — enter or select logical tags to easily identify the dashboard in the catalog. This is an optional field.
4. Click **Next**.
5. Enter a custom stylesheet for the dashboard. This is an optional step.
6. Click **Save**.

   You can see the dashboard in the Catalog.

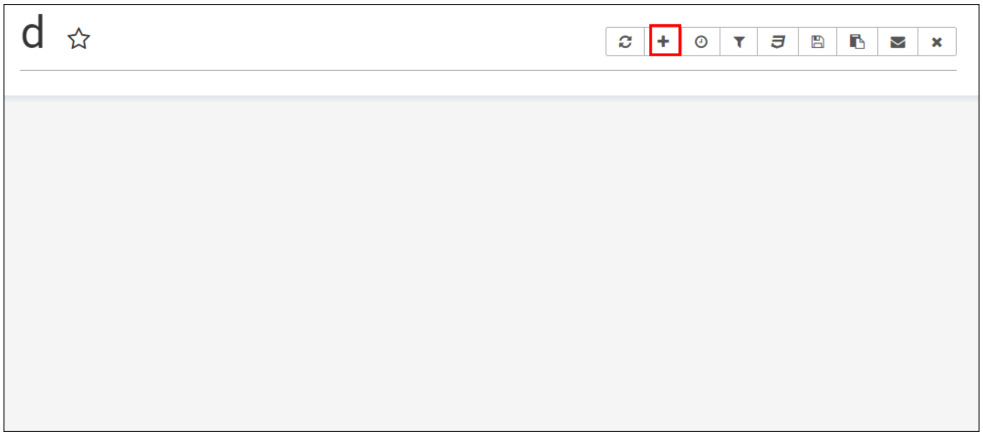
After you have created the dashboard, it is just an empty dashboard. You need to start adding visualizations to the dashboard.

Editing a Dashboard

To edit a dashboard:

1. Click the required dashboard in the catalog.

   The dashboard opens in the dashboard editor.



2. Click the **Add a new visualization** icon to see a list of existing visualizations. Visualizations from the pipelines and as well as from the cube explorations appear here. Go through the list, select one or more visualizations and add them to the dashboard.

3. Click the **Specify refresh interval** icon to select the refresh frequency for the dashboard. This is applicable only for cube based visualizations not applicable for streaming charts created out of pipeline.

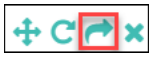
   This just a client side setting and is not persisted with the Superset Version `0.17.0`.

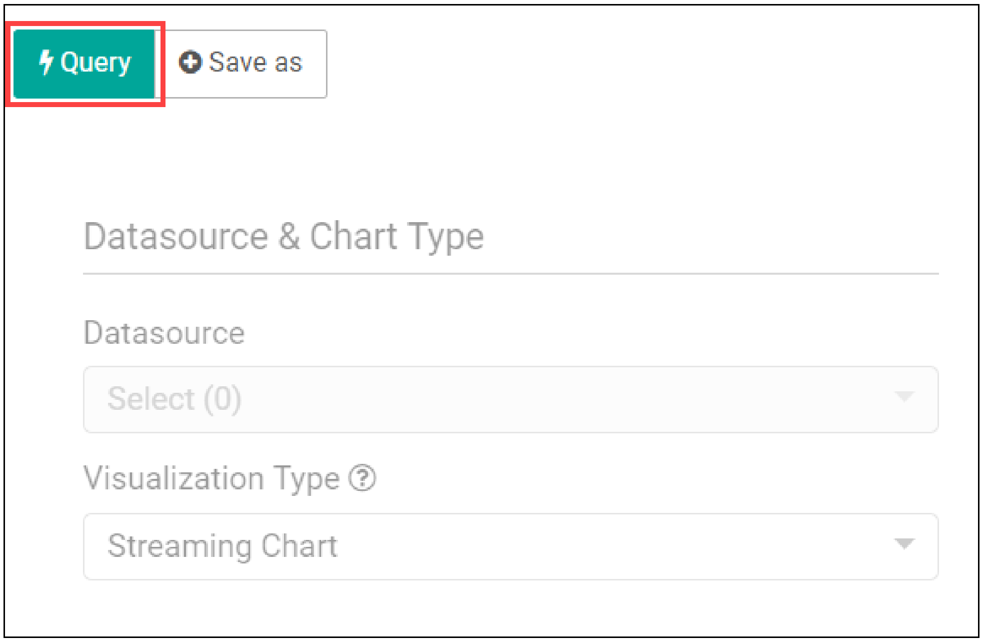
4. Click the **Apply CSS to the dashboard** icon to select a CSS. You can also edit the CSS in the live editor.

   You can also see the active filter applied to the dashboard by clicking the **Active dashboard filters** icon. You can save the link to the dashboard or email the link to someone using the **Copy the link to the clipboard** and **Email the link** icons respectively.

5. Click the **Save** icon to save the changes you have made to the dashboard.

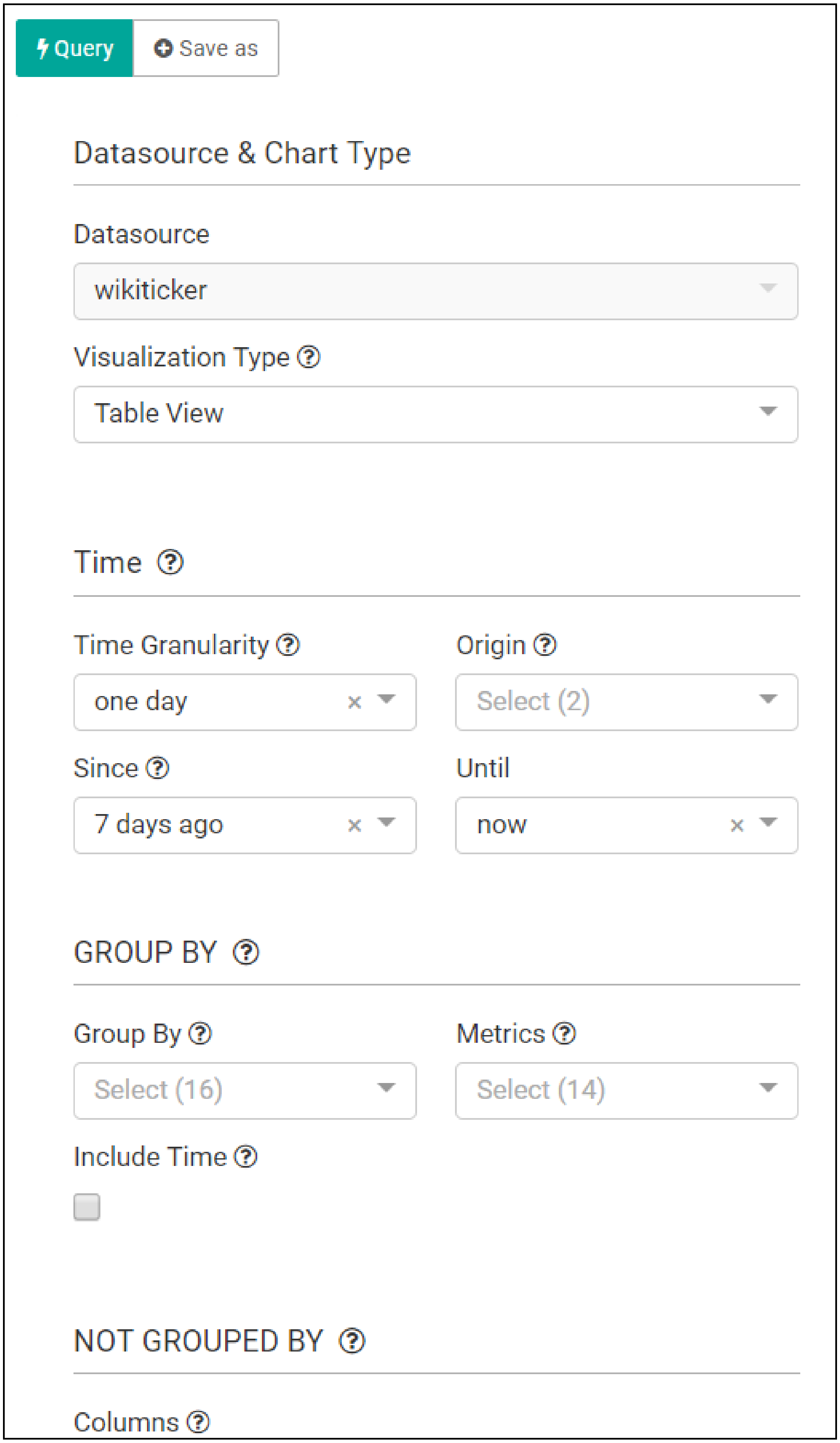
6. Hover over the added visualization, click the **Explore chart** icon to open the chart editor of the visualization.





You can see the metadata of the visualization. You can also move the chart around the canvas, refresh it, or remove it from the dashboard.

A cube exploration looks like the following:



Datasource & Chart Type

Datasource: wikiticker

Visualization Type: Table View

Time

Time Granularity: one day

Origin: Select (2)

Since: 7 days ago

Until: now

GROUP BY

Group By: Select (16)

Metrics: Select (14)

Include Time: □

NOT GROUPED BY

Columns

The various options like time granularity, group by, table timestamp format, row limit, filters, and result filters add more granularity and details to the dashboard.

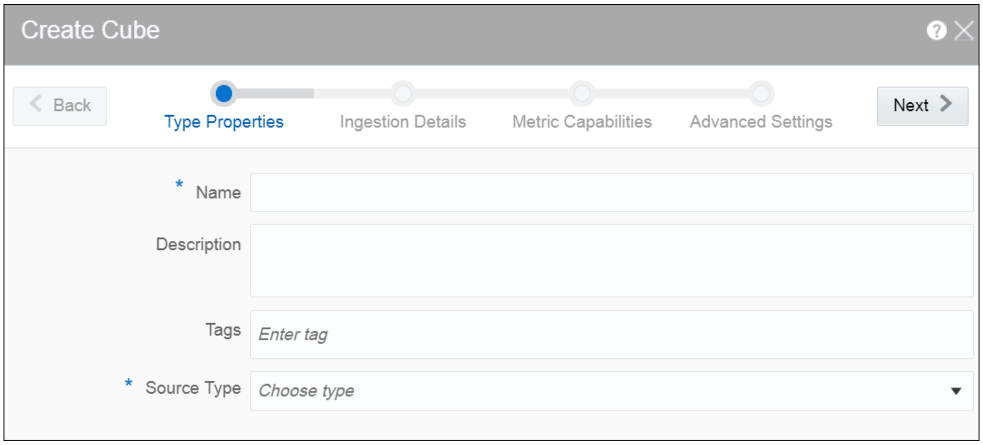
7. Click **Save as** to make the following changes to the dashboard:
   - Overwrite the visualization
   - Overwrite the current visualization with a different name
   - Add the visualization to an existing dashboard
   - Add the visualization to a new dashboard

Creating a Cube

Cube is a data structure that helps in quickly analyzing the data related to a business problem on multiple dimensions.

To create a cube:

1. Go to the **Catalog**.
2. From the **Create New Item** menu, select **Cube**.
3. On the Create Cube — Type Properties screen, provide suitable details for the following fields:
   - **Name** — enter a name for the cube. This is a mandatory field.
     Make sure that the names you use for the underlying sources for the cube like Pipeline Name, Druid Connection, and Kafka Target use names that contain alphanumeric, hyphen, and underscore characters.
   - **Description** — enter a suitable description for the cube. This is an optional field.
   - **Tags** — enter or select logical tags for the cube. This is an optional field.
   - **Source Type** — select the source type from the drop-down list. Currently, Published Pipeline is the only supported type. This is a mandatory field.



Create Cube

* Name

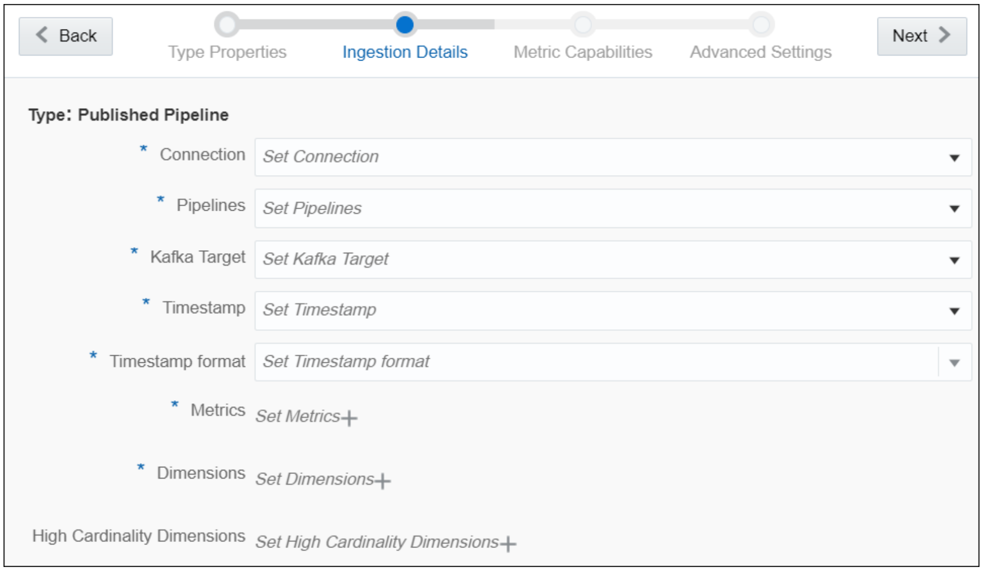
Description

Tags: *Enter tag*

* Source Type: *Choose type*

4. Click **Next** and provide suitable details for the following fields on the Ingestion Details screen:
   - **Connection** — the connection for the cube. This is a mandatory field.
   - **Pipelines** — select a pipeline to be used as the base for the cube. This is a mandatory field.
   - **Kafka Target** — the Kafka target for the cube. This is a mandatory field.

- **Timestamp** — select a column from the pipeline to be used as the timestamp. This is a mandatory field.
- **Timestamp format** — select or set a suitable format for the timestamp using Joda time format. This is a mandatory field. *auto* is the default value.
- **Metrics** — select metrics for creating measures
- **Dimensions** — select dimensions for group by
- **High Cardinality Dimensions** — high cardinality dimensions such as unique IDs. Hyperlog approximation will be used.



Type: Published Pipeline

* Connection: *Set Connection*

* Pipelines: *Set Pipelines*

* Kafka Target: *Set Kafka Target*

* Timestamp: *Set Timestamp*

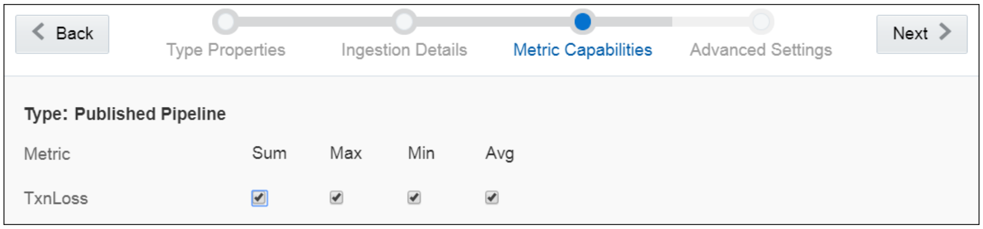
* Timestamp format: *Set Timestamp format*

* Metrics: *Set Metrics*+

* Dimensions: *Set Dimensions*+

High Cardinality Dimensions: *Set High Cardinality Dimensions*+

5. Click **Next** and select the required values for the **Metric** on the Metric Capabilities screen.

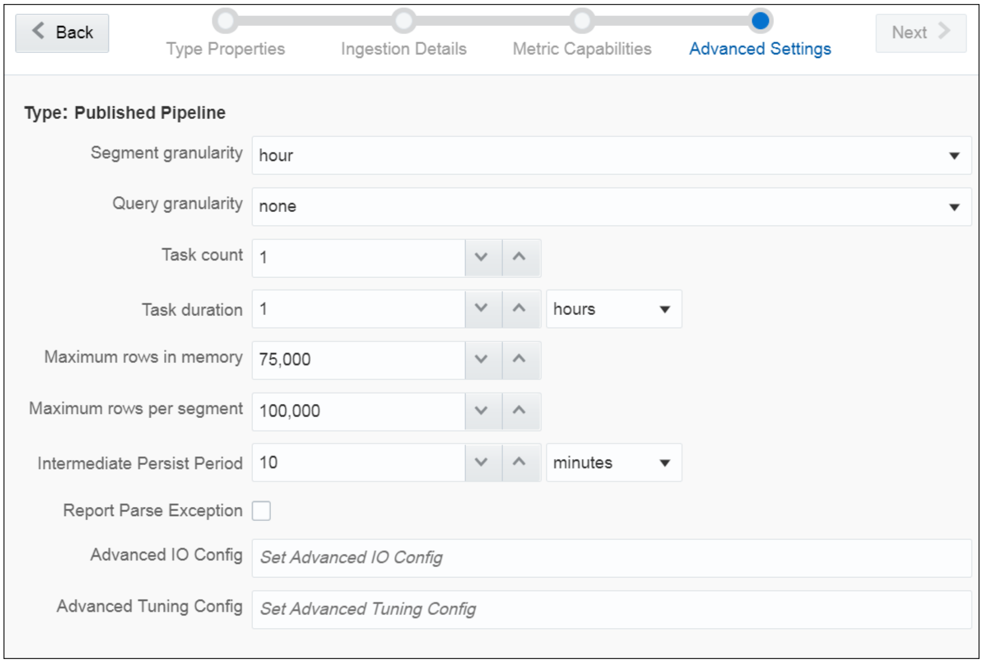


Type: Published Pipeline

Metric	Sum	Max	Min	Avg
TxnLoss	☑	☑	☑	☑

6. Click **Next** and make any changes, if required, on the Advanced Settings screen.
   - **Segment granularity** — select the granularity with which you want to create segments
   - **Query granularity** — select the minimum granularity to be able to query results and the granularity of the data inside the segment
   - **Task count** — select the maximum number of reading tasks in a replica set. This means that the maximum number of reading tasks is `taskCount*replicas` and the total number of tasks (reading + publishing) is higher than this. The number of reading tasks is less than `taskCount if taskCount > {numKafkaPartitions}`.
   - **Task duration** — select the length of time before tasks stop reading and begin publishing their segment. The segments are only pushed to deep storage and loadable by historical nodes when the indexing task completes.

- **Maximum rows in memory** — enter a number greater than or equal to 0. This number indicates the number of rows to aggregate before persisting. This number is the post-aggregation rows, so it is not equivalent to the number of input events, but the number of aggregated rows that those events result in. This is used to manage the required JVM heap size. Maximum heap memory usage for indexing scales with `maxRowsInMemory*(2 + maxPendingPersists)`.
- **Maximum rows per segment** — enter a number greater than or equal to 0. This is the number of rows to aggregate into a segment; this number is post-aggregation rows.
- **Immediate Persist Period** — select the period that determines the rate at which intermediate persists occur. This allows the data cube is ready for query earlier before the indexing task finishes.
- **Report Parse Exception** — select this option to throw exceptions encountered during parsing and halt ingestion.
- **Advanced IO Config** — specify name-value pair in a CSV format. Available configurations are `replicas`, `startDelay`, `period`, `useEarliestOffset`, `completionTimeout`, and `lateMessageRejectionPeriod`.
- **Advanced Tuning Config** — specify name-value pair in CSV format. Available configurations are `maxPendingPersists`, `handoffConditionTimeout`, `resetOffsetAutomatically`, `workerThreads`, `chatThreads`, `httpTimeout`, and `shutdownTimeout`.



Type: Published Pipeline

Segment granularity	hour
Query granularity	none
Task count	1
Task duration	1 hours
Maximum rows in memory	75,000
Maximum rows per segment	100,000
Intermediate Persist Period	10 minutes
Report Parse Exception	□
Advanced IO Config	Set Advanced IO Config
Advanced Tuning Config	Set Advanced Tuning Config

7. Click **Save** to save the changes you have made.

You can see the cube you have created in the catalog.

Exploring a Cube

When you create druid based cube, you can explore data in it.

To explore a cube:

1. In the **Catalog**, click the cube that you want to explore.

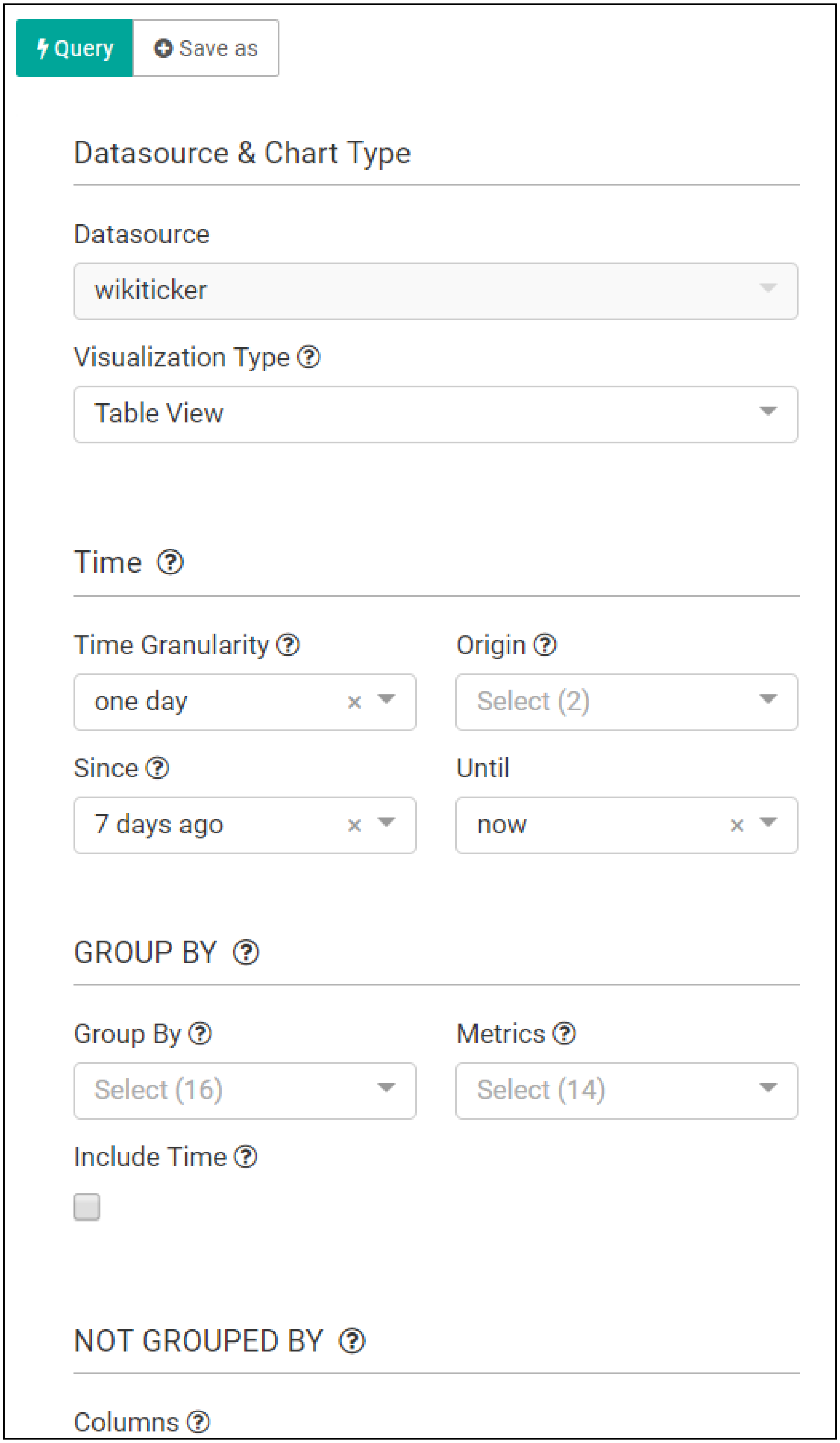
   The Cube Exploration canvas appears.

2. Construct a query by setting the various parameters.

   - Visualization Type — the type of visualization to be used for displaying data. The supported visualizations are:

Distribution - Bar Chart	Separator
Sunburst	Pie Chart
World Cloud	Sankey
Time Series - Line Chart	Treemap
Directed force Layout	Time Series - Dual Axis Line Chart
Calendar Heatmap	World Map
Time Series - Bar Chart	Box Plot
Filter Box	Time Series - Percent Change
Bubble Chart	iFrame
Time Series - Stacked	Bullet Chart
Streaming Chart	Table View
Big Number with Trendline	Parallel Coordinates
Markup	Big Number
Heatmap	Pivot Table
Histogram	Horizon

   - Time — time related form attributes like time granularity, origin (starting point of time), and time range
   - Group By — parameters to aggregate the query data
   - Not Grouped By — parameter to query atomic rows
   - Options
   - Filters — columns that you can use in filters
   - Result Filters — columns that you can use in result filters



Datasource & Chart Type

Datasource: wikiticker

Visualization Type ?: Table View

Time ?

Time Granularity ?: one day

Origin ?: Select (2)

Since ?: 7 days ago

Until: now

GROUP BY ?

Group By ?: Select (16)

Metrics ?: Select (14)

Include Time ?: □

NOT GROUPED BY ?

Columns ?

3. Click **Query** to run the query with the defined parameters.

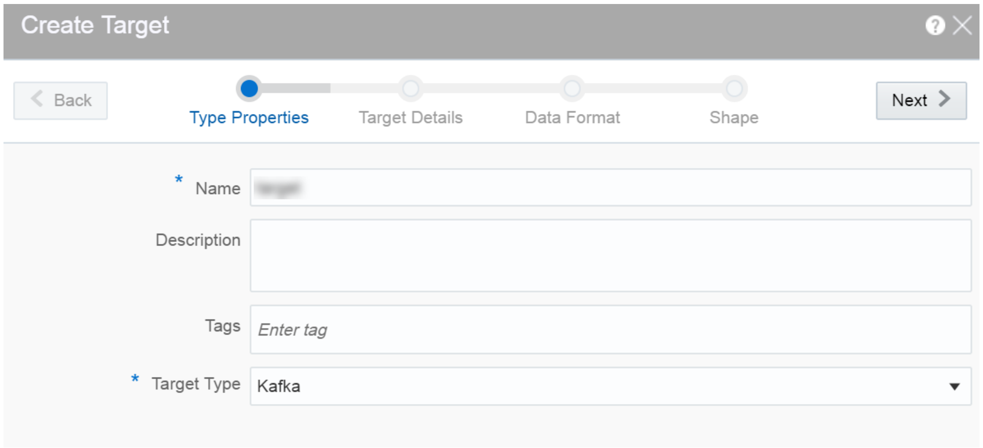
4. Click **Save As** to save the cube exploration. You can save it as a visualization, choose to add it to an existing dashboard, not to add to a dashboard, or to a new dashboard.

Creating a Target

The target defines a destination for output data coming from a pipeline.

To create a target:

1. Navigate to **Catalog**.
2. Select **Target** in the **Create New Item** menu.
3. Provide details for the following fields on the **Type Properties** page and click **Save** and **Next**:
   - **Name** — name of the target
   - **Description** — description of the target
   - **Tags** — tags you want to use for the target
   - **Target Type** — the transport type of the target. Supported types are JMS, Kafka and Rest. The target is a sink for the output event. Each type of target is a different sink system and therefore different configuration parameters are required for different types.



* Name

Description

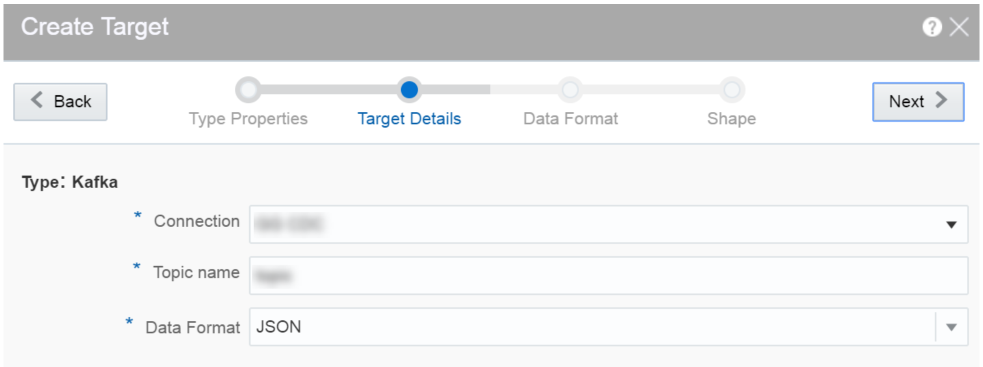
Tags: *Enter tag*

* Target Type: Kafka

4. Provide details for the following fields on the **Target Details** page and click **Next**:

   When the target type is JMS:

   - **Connection** — the connection for the target
   - **Jndi name** — the topic or queue name defined in Jndi to be used in the target



Type: Kafka

* Connection [redacted]

* Topic name [redacted]

* Data Format JSON

- **Data Format** — select a suitable data format. This is a mandatory field. The supported data format types are: CSV and JSON.

When the target type is Kafka:

- **Connection** — the connection for the target
- **Topic Name** — the Kafka topic to be used in the target
- **Data Format** — select a suitable data format. This is a mandatory field. The supported data format types are: CSV and JSON.

When the target type is REST:

- **URL** — enter the REST service URL. This is a mandatory field.
- **Custom HTTP headers** — set the custom headers for HTTP. This is an optional field.
- **Batch processing** — select this option to send events in batches and not one by one. Enable this option for high throughput pipelines. This is an optional field.
- **Data Format** — select a suitable data format. This is a mandatory field.

Click **Test connection** to check if the connection has been established successfully.

Testing REST targets is a heuristic process. It uses proxy settings. The testing process uses GET request to ping the given URL and returns success if the server returns `OK (status code 200)`. The return content is of the type of `application/json`.

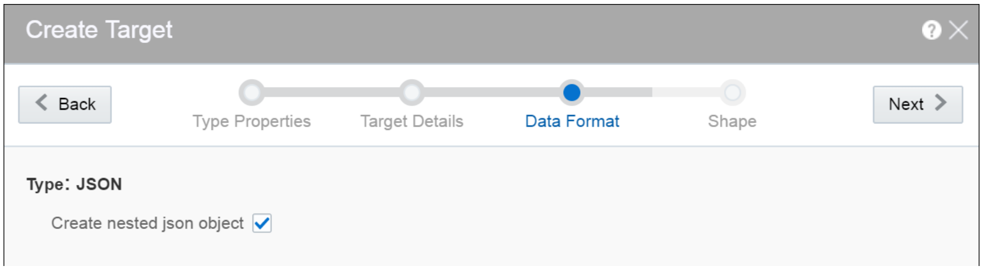
5. Provide details for the following fields on the **Data Format** page and click **Next**:

   When the data format type is CSV:

   - **CSV Predefined Format** — select a predefined CSV format. This supported formats are: Excel, InfomixUnload, InfomixUnloadCsv, MySQL, PostgreSQLCsv, PostgreSQLText.
   - **Create the header row** — select this option if you want to create a header row in the target.

   When the data format type is JSON:

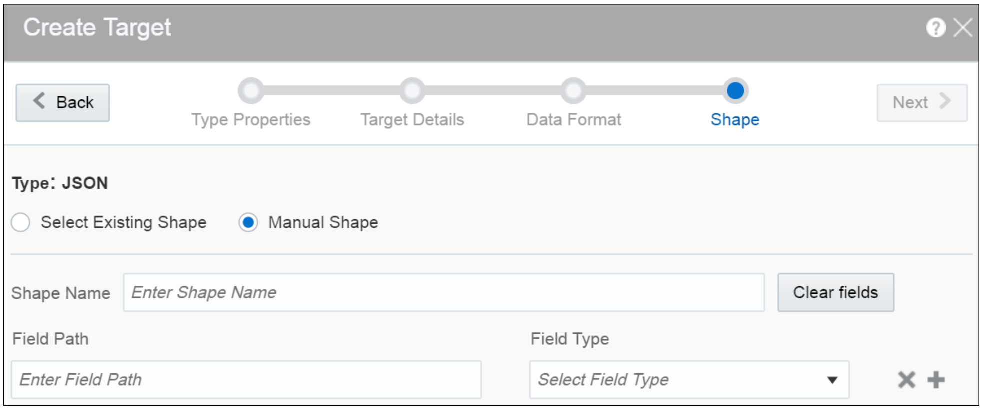
   - **Create nested json object** — select this option if you want a nested json object to be created for the target



Type: JSON

Create nested json object ☑

6. Select one of the mechanisms to define the shape on the **Shape** page and click **Save**:
   - **Select Existing Shape** lets you choose one of the existing shapes from the drop-down list.
   - **Manual Shape** populates the existing fields and also allows you to add or remove columns from the shape. You can also update the datatype of the fields.



Type: JSON

○ Select Existing Shape ● Manual Shape

Shape Name: *Enter Shape Name*

Field Path: *Enter Field Path*

Field Type: *Select Field Type*

A target is created with specified details.

Creating Target from Pipeline Editor

Alternatively, you can also create a target from the pipeline editor. When you click **Create** in the target stage, you are navigated to the **Create Target** dialog box. Provide all the required details and complete the target creation process. When you create a target from the pipeline editor, the shape gets pre-populated with the shape from the last stage.

Creating a Geo Fence

Geo fences are further classified into two categories: manual geo fence and database-based geo fence.

Create a Manual Geo Fence

To create a manual geo fence:

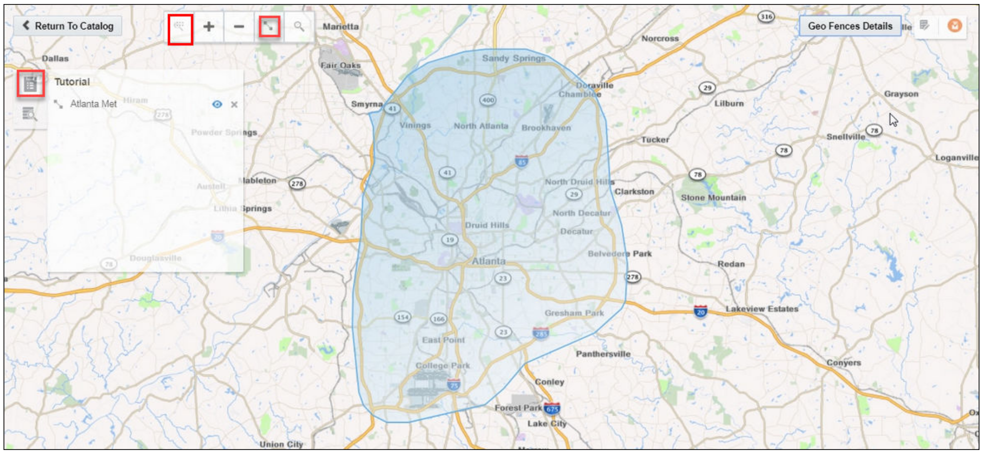
1. Navigate to the **Catalog** page.
2. Click **Create New Item** and select **Geo Fence** from the drop-down list.

   The Create Geo Fence dialog opens.
3. Enter a suitable name for the **Geo Fence**.

4. Select **Manually Created Geo Fence** as the **Type**.
5. Click **Save**.

   The Geo Fence Editor opens. In this editor you can create the geo fence according to your requirement.
6. Within the Geo Fence Editor, **Zoom In** or **Zoom Out** to navigate to the required area using the zoom icons in the toolbar located on the top-left side of the screen.

   You can also use the **Marquee Zoom** tool to zoom a specific area on the map. You can mark an area using the marquee zoom and that area in map is zoomed.
7. Click the **Polygon Tool** and mark the area around a region to create a geo fence.



8. Enter a name and description, and click **Save** to save your changes.

Update a Manual Geo Fence

To update a manual geo fence:

1. Navigate to the **Catalog** page.
2. Click the name of the geo fence you want to update.

   The Geo Fence Editor opens. You can edit/update the geo fence here.

Search Within a Manual Geo Fence

You can search the geo fence based on the country and a region or address. The search field allows you search within the available list of countries. When you click the search results tile in the left center of the geo fence and select any result, you are automatically zoomed in to that specific area.

Delete a Manual Geo Fence

To delete a manual geo fence:

1. Navigate to **Catalog** page.
2. Click **Actions**, then select **Delete Item** to delete the selected geo fence.

Create a Database-based Geo Fence

To create a database-based geo fence:

1. Navigate to **Catalog** page.

2. Click **Create New Item** and then select **Geo Fence** from the drop-down list.

   The Create Geo Fence dialog opens.

3. Enter a suitable name for the geo fence.
4. Select **Geo Fence** from Database as the Type.
5. Click **Next** and select **Connection**.
6. Click **Next**.

   All tables that have the field type as `SDO_GEOMETRY` appear in the drop-down list.

7. Select the required table to define the shape.
8. Click **Save**.

> **Note:**
>
> You cannot edit/update database-based geo fences.

Delete a Database-based Geo Fence

To delete a database-based geo fence:

1. Navigate to **Catalog** page.
2. Click **Actions** and then select **Delete Item** to delete the selected geo fence.

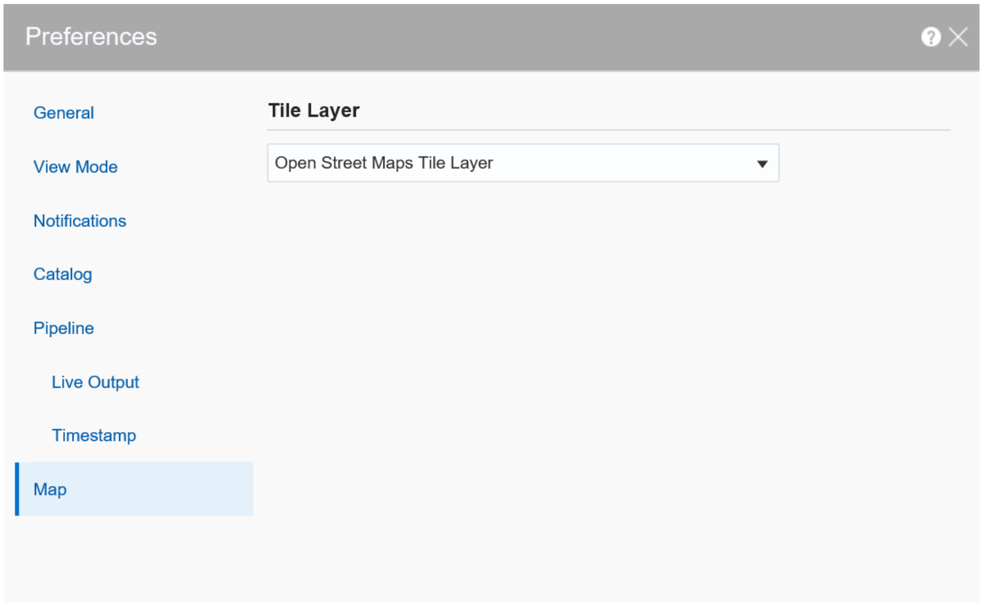
Display the Map Using Tile Layers

Tile layer is the base map that provides immediate geographic context. Tiles are stored in the map tile server. <ph ishcondition="Product_Family=Cloud" varref="streaming">Stream Analytics</ph><ph ishcondition="Product_Family=OnPremise" varref="osa">Oracle Stream Analytics</ph> supports two types of tile layers. Open Street Maps tile layer is a free map. And, Elocation tile layer is an Oracle tile layer. These tile layers contains huge amount of data pertaining to:

- Roads, railways, waterways, etc.
- Restaurants, shops, stations, ATMs, and more
- Walking and cycling paths
- Buildings, campuses, etc.

You can choose if you would like to see the map in Elocation tile layer or Open Street Maps tile layer. To set your preference:

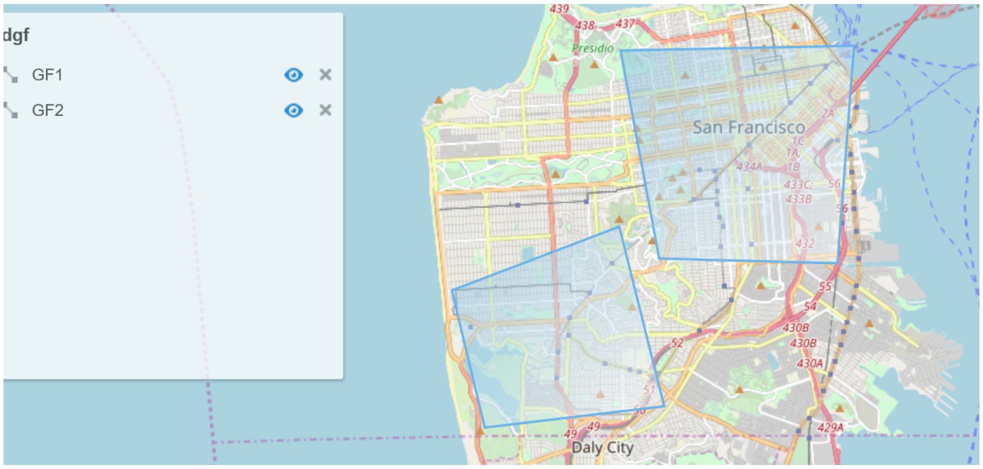
1. Click the user name in the top right corner of the screen.
2. Click **Preferences**. The Preferences page opens.
3. Click **Map**.
4. Under **Tile Layer**, choose **Open Street Maps Tile Layer** option from the drop-down list.



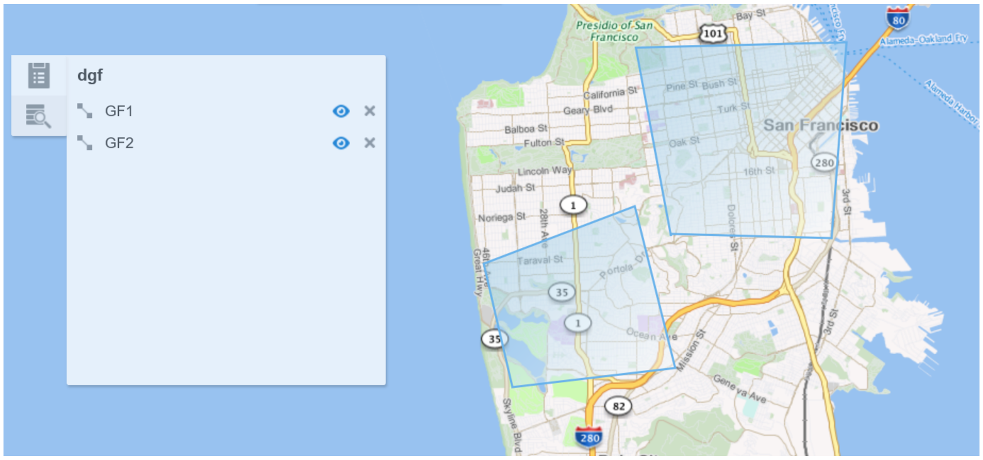
Tile Layer

Open Street Maps Tile Layer

5. Click **Save**. The map looks like this:



6. To display the map in Elocation tile layer, follow steps 1 to 3.
7. From the **Tile Layer** drop-down list, choose **Elocation Tile Layer**.
8. Click **Save**. The map looks like this:



Creating a Predictive Model

To create a predictive model:

1. In the **Create New Item** menu, select **Predictive Model (Beta)**.

   The Create Predictive Model page opens.

2. Under **Type Properties** do the following and then click **Next**:

   a. In the **Name** field, enter a meaningful name for your PMML model.

   b. In the **Predictive Model Type** drop-down list, select **PMML Model**.

   > **Note:**
   >
   > Only PMML Models up to version 4.1 are supported in this release.

3. Under **Predictive Model Details**, do the following and click **Save**:

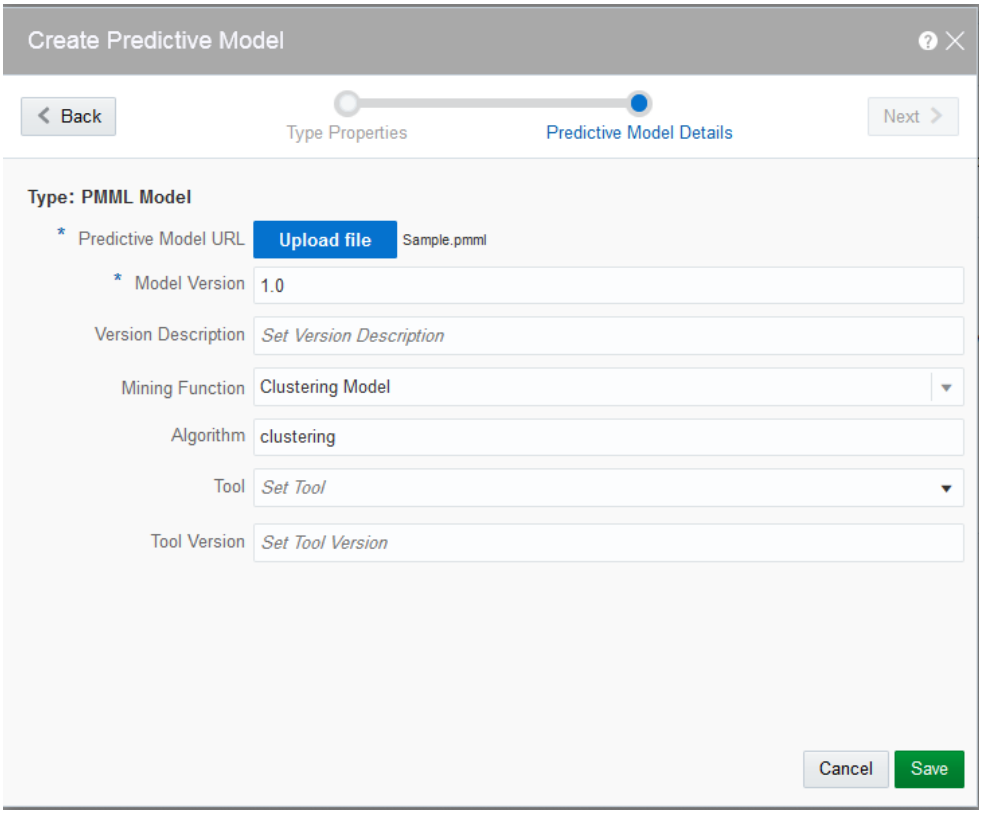
   a. For **Predictive Model URL**, upload your PMML file.

   b. In the **Model Version** field, enter the version of this artifact. For example, `1.0`.

   c. (Optional) In the **Version Description**, enter a meaningful description for your PMML file.

   d. In the **Algorithm** field, accept the default. The algorithm is derived from the PMML file you have uploaded.

   e. (Optional) In the **Tool** drop-down list, select the tool with which you created your PMML file.



Type: PMML Model

* Predictive Model URL: Sample.pmml

* Model Version: 1.0

Version Description: Set Version Description

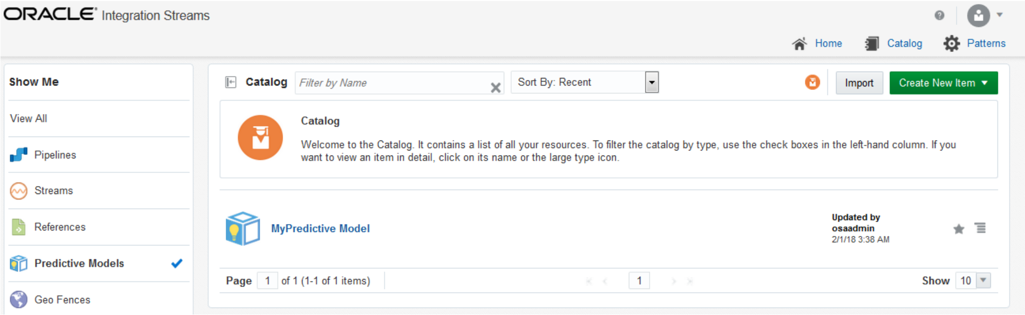
Mining Function: Clustering Model

Algorithm: clustering

Tool: Set Tool

Tool Version: Set Tool Version

Your predictive model has been created. It is displayed in the **Catalog** if you have selected the **Predictive Models** option under **Show Me**.



Catalog

Welcome to the Catalog. It contains a list of all your resources. To filter the catalog by type, use the check boxes in the left-hand column. If you want to view an item in detail, click on its name or the large type icon.

MyPredictive Model — Updated by osaadmin 2/1/18 3:38 AM

Page 1 of 1 (1-1 of 1 items) 1 Show 10

Limited Support for Predictive Models

The menu commands for creating Predictive Models and Scoring Stages are marked **Beta**, for example, **Predictive Model (Beta)**. The **Beta** label indicates that the functionality has been tested, but is not fully supported. The import and scoring of Predictive Models might contain undocumented limitations and you should use them as is.

Creating a Custom Jar

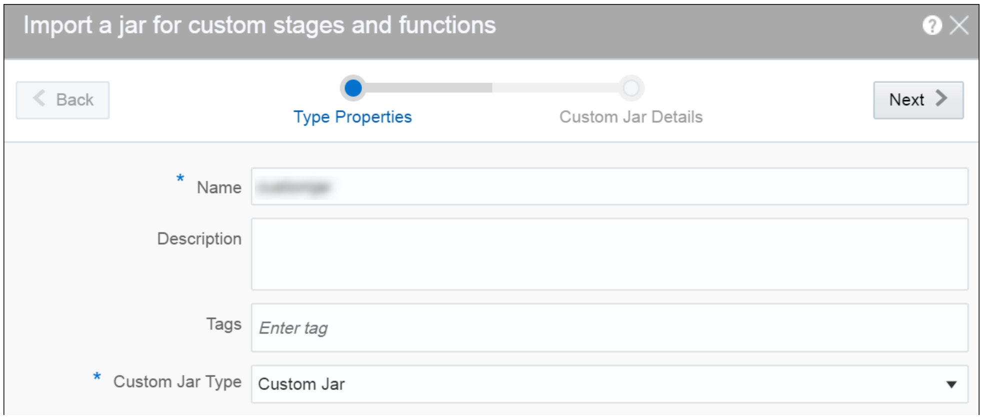
A custom jar is a user-supplied Jar archive containing Java classes for custom stage types or custom functions that will be used within a pipeline.

To create a custom jar:

1. In the **Create New Item** menu, select **Custom Jar**.

The Import a jar for custom stages and functions wizard appears.

2. On the **Type Properties** page, enter/select suitable values and click Next:
    1. In the **Name** field, enter a meaningful name for the custom jar you are trying to import into the application.
    2. In the **Description** field, provide a suitable description.
    3. In the **Tags** field, select one or more of existing tags, or enter your own tags.
    4. In the **Custom Jar Type** drop-down list, select **Custom Jar**.



* Name [redacted]

Description

Tags Enter tag

* Custom Jar Type Custom Jar

3. On the **Custom Jar Details** page, click **Upload file**, select the jar file that you want to import into the application, and then click **Save**.

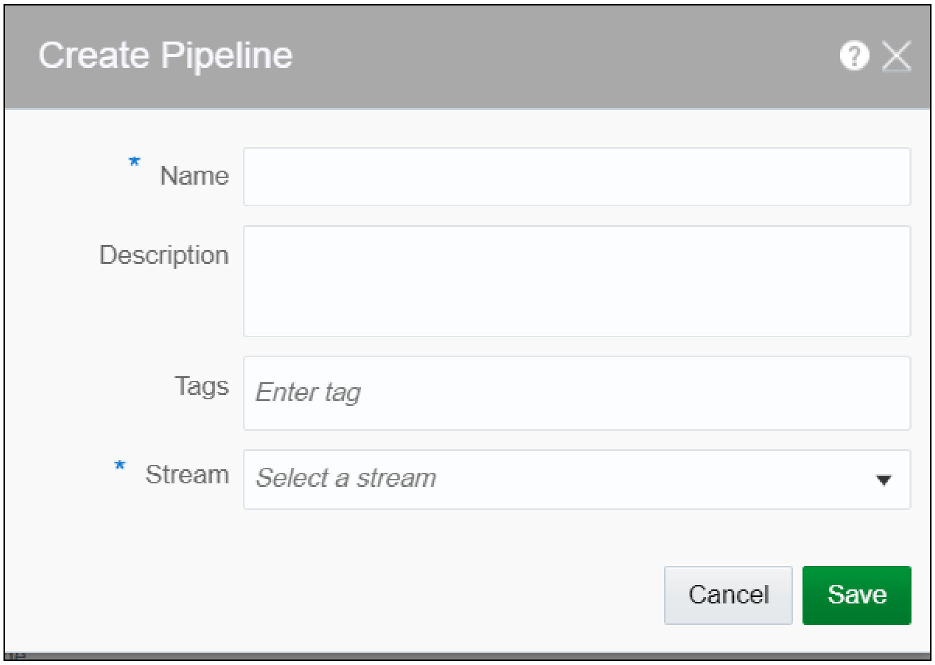
    Make sure that the jar file you select for uploading is a valid jar file and includes all the required dependencies.

Creating a Pipeline

A pipeline is a Spark application where you implement your business logic. It can have multiple stages such as a query stage, a pattern stage, a business rule stage, a query group stage, a custom stage and many more.

To create a pipeline:

1. Navigate to **Catalog**.
2. Select **Pipeline** in the **Create New Item** menu.
3. Provide details for the following fields and click **Save**:
    - **Name** — name of the pipeline
    - **Description** — description of the pipeline
    - **Tags** — tags you want to use for the pipeline
    - **Stream** — the stream you want to use for the pipeline



A pipeline is created with specified details.

Configuring a Pipeline

You can configure the pipeline to use various stages like query, pattern, rules, query group, scoring, and custom stage from custom jars.

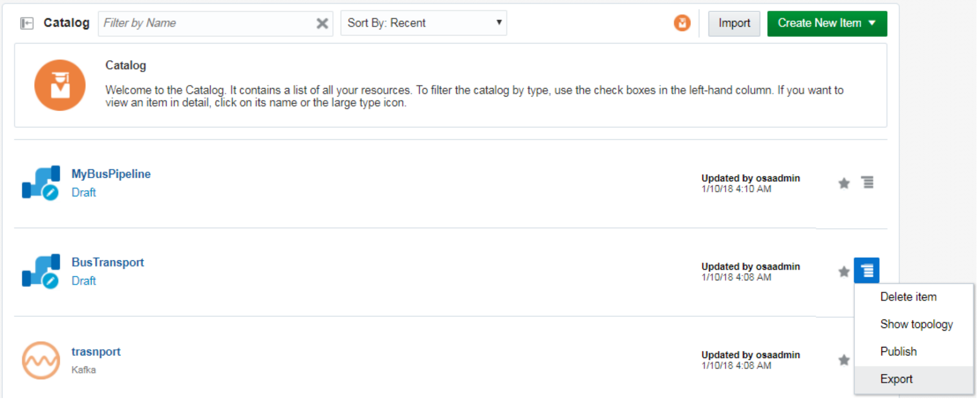
Exporting and Importing a Pipeline and Its Dependent Artifacts

The export and import feature lets you migrate your pipeline and its contents between Stream Analytics systems (such as development and production) in a matter of few clicks. You also have the option to migrate only select artifacts. You can import a pipeline developed with the latest version of Stream Analytics. On re-import, the existing metadata is overwritten with the newly imported metadata if the pipeline is not published. You can delete the imported artifacts by right-clicking them and selecting **Delete**.

You can export and import pipelines and artifacts except for the following:

- Cubes
- Dashboards
- Custom Stages
- Visualizations
- File Streams
- Predictive Models

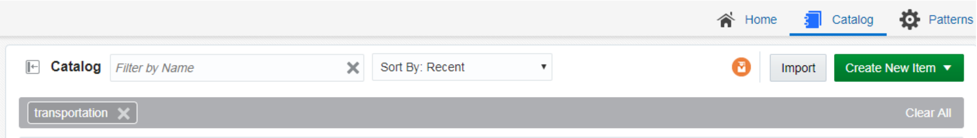
1. In your Stream Analytics instance, under **Catalog**, right-click the pipeline or artifact that you want to export to another instance of the Stream Analytics and then select **Export**.



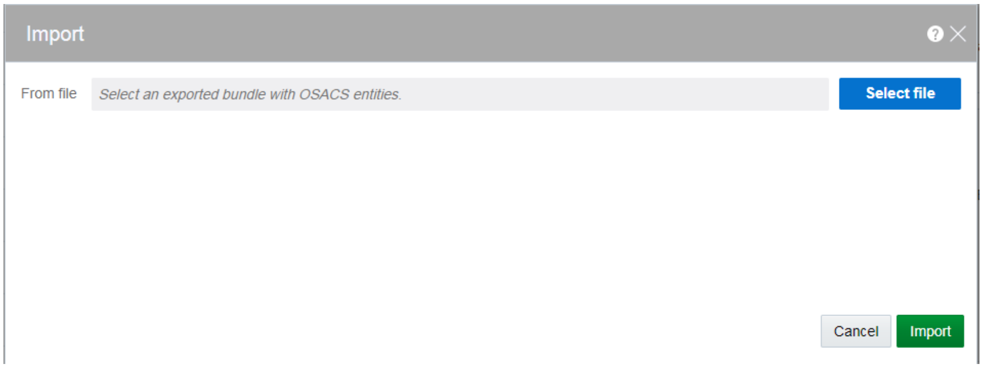
Welcome to the Catalog. It contains a list of all your resources. To filter the catalog by type, use the check boxes in the left-hand column. If you want to view an item in detail, click on its name or the large type icon.

Your items are exported as a ZIP file.

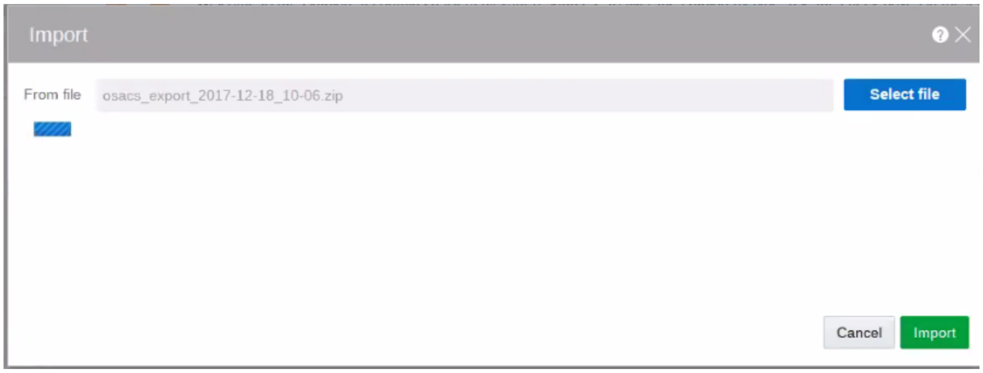
2. Go to the Stream Analytics instance to which you want to import the exported metadata.
3. On the toolbar, click **Import**.



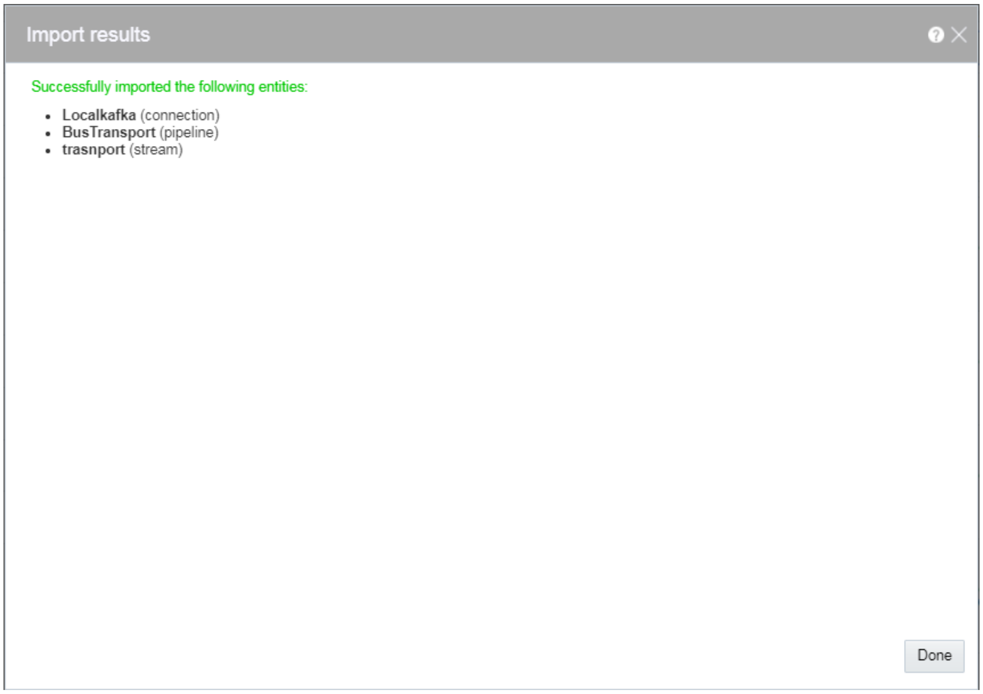
4. In the **Import** dialog box, click **Select file** and then select the exported ZIP file.



5. Click **Import**.



When the metadata is imported successfully, a message similar to the following appears:



Successfully imported the following entities:

- **Localkafka** (connection)
- **BusTransport** (pipeline)
- **trasnport** (stream)

Publishing a Pipeline

You must publish a pipeline to make the pipeline available for all the users of Stream Analytics and send data to targets.

A published pipeline will continue to run on your Spark cluster after you exit the Pipeline Editor, unlike the draft pipelines which are undeployed to release resources.

To publish a pipeline:

1. Open a draft pipeline in the **Pipeline Editor**.
2. Click **Publish**.

   The Pipeline Settings dialog box opens.
3. Update any required settings.

   > **Note:**
   >
   > Make sure to allot more memory to executors in the scenarios where you have large windows.
4. Click **Publish** to publish the pipeline.

   A confirmation message appears when the pipeline is published.

You can also publish a pipeline from the Catalog using the **Publish** option in the **Actions** menu.

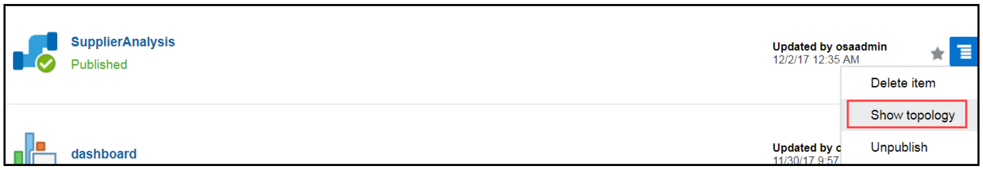
Using the Topology Viewer

Topology is a graphical representation and illustration of the connected entities and the dependencies between the artifacts.

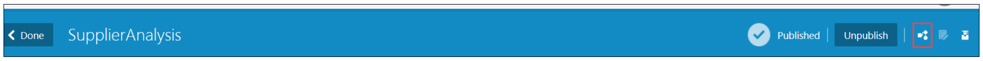
The topology viewer helps you in identifying the dependencies that a selected entity has on other entities. Understanding the dependencies helps you in being cautious while deleting or undeploying an entity. Stream Analytics supports two contexts for the topology — *Immediate Family* and *Extended Family*.

You can launch the Topology viewer in any of the following ways:

- Select **Show topology** from the **Catalog Actions** menu to launch the **Topology Viewer** for the selected entity.



- Click the **Show Topology** icon in the Pipeline Editor.



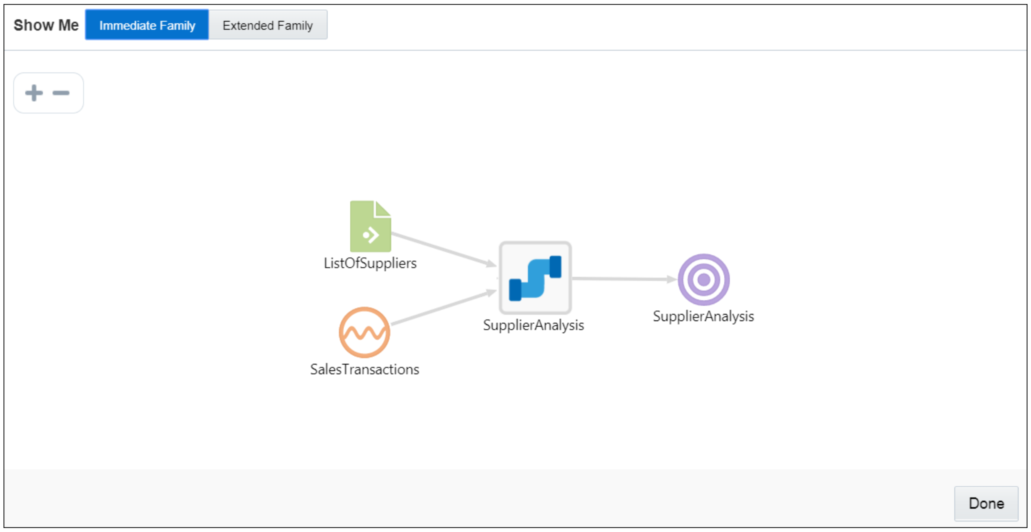
Click the **Show Topology** icon at the top-right corner of the editor to open the topology viewer.By default, the topology of the entity from which you launch the Topology Viewer is displayed. The context of this topology is **Immediate Family**, which indicates that only the immediate dependencies and connections between the entity and other entities are shown. You can switch the context of the topology to display the full topology of the entity from which you have launched the Topology Viewer. The topology in an **Extended Family** context displays all the dependencies and connections in the topology in a hierarchical manner.

> **Note:**
>
> The entity for which the topology is shown has a grey box surrounding it in the Topology Viewer.

Immediate Family

Immediate Family context displays the dependencies between the selected entity and its child or parent.

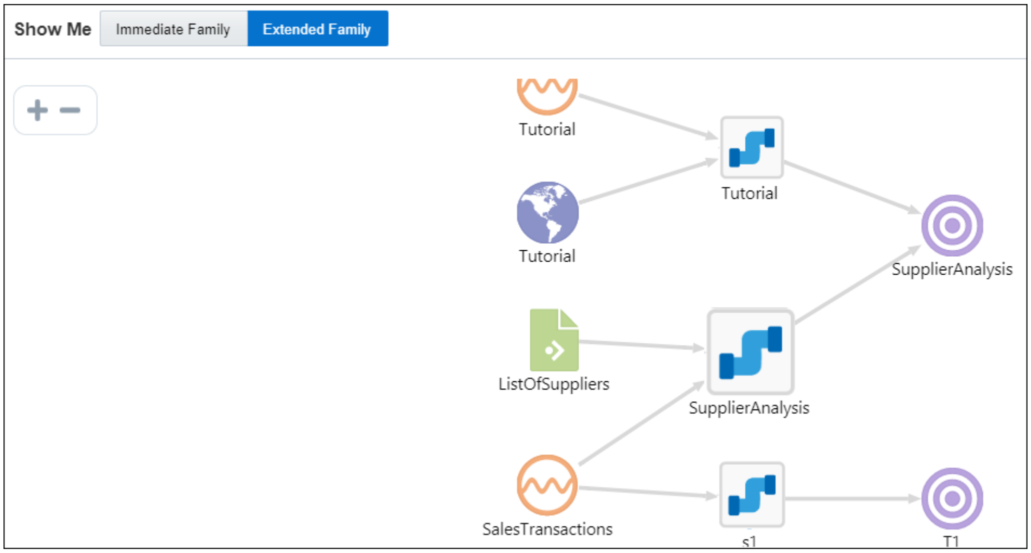
The following figure illustrates how a topology looks in the **Immediate Family**.



Extended Family

Extended Family context displays the dependencies between the entities in a full context, that is if an entity has a child entity and a parent entity, and the parent entity has other dependencies, all the dependencies are shown in the Full context.

The following figure illustrates how a topology looks in the **Extended Family**.



Working with Patterns

This topic applies only to Oracle user-managed services.

Patterns are a stage within a pipeline. When working from a pattern, you need to specify a few key fields to discover an interesting result. You can create pattern stages within the pipeline. Patterns are not stand-alone artifacts, they need to be embedded within a pipeline.

Topics:

- About Stream Analytics Patterns
- Creating a Pipeline Using a Pattern

About Stream Analytics Patterns

This topic applies only to Oracle user-managed services.

The visual representation of the event stream varies from one pattern type to another based on the key fields you choose.

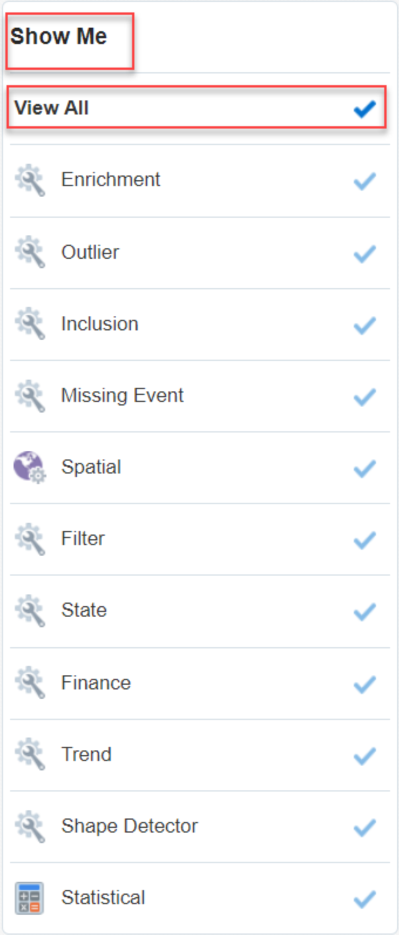
Click **Patterns** on the Home page to see all the available patterns. Use the filters at left to view different categories of pattern. You can see full descriptions and learn more about each pattern by clicking the user assistant icon. Click again to hide the extra information.

A *pattern* provides you with the results displayed in a live output stream based on common business scenarios.

> **Note:**
>
> While entering data in the fields for a specific pattern, ensure that the data you enter corresponds to the datatype of the field. If there is a mismatch between the entered data and the datatype, the pattern will not deploy and throw an error.

You can include or exclude patterns based on their categories using the **View All** link in the left panel under **Show Me**. When you click **View All**, a check mark appears next to it and all the patterns are displayed on the page.

When you want to display/view only a few/selective patterns, deselect **View All** and select the individual patterns. Only the selected patterns are shown in the catalog.



Show Me

View All	✓
Enrichment	✓
Outlier	✓
Inclusion	✓
Missing Event	✓
Spatial	✓
Filter	✓
State	✓
Finance	✓
Trend	✓
Shape Detector	✓
Statistical	✓

The following table lists the categories of patterns:

Category	Pattern
Enrichment	Reverse Geo Code: Near By Left Outer Join
Outlier	Fluctuation
Inclusion	Union Left Outer Join
Missing Event	'A' Not Followed by 'B' Detect Missing Event

Category	Pattern
Spatial	Proximity: Stream with Geo Fence Geo Fence Spatial: Speed Interaction: Single Stream Reverse Geo Code: Near By Geo Code Spatial: Point to Polygon Interaction: Two Stream Proximity: Two Stream Direction Reverse Geo Code: Near By Place Proximity: Single Stream Geo Filter
Filter	Eliminate Duplicates Fluctuation
State	'A' Not Followed by 'B' Inverse W Detect Missing Event W 'A' Followed by 'B' ‘B’ Not Preceded by ‘A’
Finance	Inverse W W
Trend	'A' Not Followed by 'B Top N Change Detector Up Trend Detect Missing Event Down Trend 'A' Followed by 'B' Detect Duplicates Bottom N
Shape Detector	Inverse W W
Statistical	Correlation Quantile

Creating a Pipeline Using a Pattern

This topic applies only to Oracle user-managed services.

Instead of creating a pattern stage from within a pipeline, you can also create a pipeline for a pattern directly.

To create a pipeline using a pattern:

1. Click **Patterns** in the left tree on the Home page.

   The Patterns page appears.

2. Scroll through the list of available patterns and select the required pattern.
3. Click **Use this pattern** within the selected pattern tile.

   The Create pipeline using <Pattern> dialog box appears.

4. Fill in the details for the metadata in the **Pipeline** section.
5. Enter details for the **Pattern Stage**.
6. Click **Save**.

   The pipeline editor opens where you can specify the parameters required for the pattern. The pipeline also appears in the Catalog.

Understanding Expression Builder Functions

This topic applies only to Oracle user-managed services.

Expression Builder is an editor that allows you to build expressions using various existing functions. The expressions help you in achieving the required results for your pipelines.

Topics:

- What are Bessel Functions?
- What are Conversion Functions?
- What are Date Functions?
- What are Geometry Functions?
- What are Interval Functions?
- What are Math Functions?
- What are Null-related Functions?
- What are Statistical Functions?
- What are String Functions?

What are Bessel Functions?

The mathematical cylinder functions for integers are known as Bessel functions.

The following Bessel functions are supported in this release:

Function Name	Description
`BesselI0(x)`	Returns the modified Bessel function of order 0 of the double argument as a double

Function Name	Description
`BesselI0_exp(x)`	Returns the exponentially scaled modified Bessel function of order 0 of the double argument as a double
`BesselI1(x)`	Returns the modified Bessel function of order 1 of the double argument as a double
`BesselI1_exp(x)`	Returns the exponentially scaled modified Bessel function of order 1 of the double argument as a double
`BesselJ(x,x)`	Returns the Bessel function of the first kind of order n of the argument as a double
`BesselK(x,x)`	Returns the modified Bessel function of the third kind of order n of the argument as a double
`BesselK0_exp(x)`	Returns the exponentially scaled modified Bessel function of the third kind of order 0 of the double argument as a double
`BesselK1_exp(x)`	Returns the exponentially scaled modified Bessel function of the third kind of order 1 of the double argument as a double
`BesselY(x)`	Returns the Bessel function of the second kind of order n of the double argument as a double

What are Conversion Functions?

The conversion functions help in converting values from one data type to other.

The following conversion functions are supported in this release:

Function Name	Description
`bigdecimal(value1)`	Converts the given value to bigdecimal
`boolean(value1)`	Converts the given value to logical
`date(value1,value2)`	Converts the given value to datetime
`double(value1)`	Converts the given value to double
`float(value1)`	Converts the given value to float
`int(value1)`	Converts the given value to integer
`long(value1)`	Converts the given value to long
`string(value1,value2)`	Converts the given value to string

What are Date Functions?

The following date functions are supported in this release:

Function Name	Description
`day(date)`	Returns day of the date
`eventtimestamp()`	Returns event timestamp from stream
`hour(date)`	Returns hour of the date
`minute(date)`	Returns minute of the date

Function Name	Description
`month(date)`	Returns month of the date
`nanosecond(date)`	Returns nanosecond of the date
`second(date)`	Returns second of the date
`systimestamp()`	Returns the system’s timestamp on which the application is running
`timeformat(value1,value2)`	Returns the provided timestamp in required time format
`timestamp()`	Returns the current output time
`year(date)`	Returns year of the date

What are Geometry Functions?

The Geometry functions allow you to convert the given values into a geometrical shape.

The following interval functions are supported in this release:

Function Name	Description
`CreatePoint(lat,long,SRID)`	Returns a 2–dimensional point type geometry from the given latitude and longitude. The default SRID is 8307. The return value is of the datatype `sdo geometry`.
`distance(lat1,long1,lat2,long2,SRID)`	Returns distance between the first set of latitude, longitude and the second set of latitude, longitude values. The default SRID is 8307. The return value is of the datatype `double`.

What are Interval Functions?

The Interval functions help you in calculating time interval from given values.

The following interval functions are supported in this release:

Function Name	Description
`numtodsinterval(n,interval_unit)`	Converts the given value to an `INTERVAL DAY TO SECOND` literal. The value of the `interval_unit` specifies the unit of `n` and must resolve to one of the string values: *DAY*, *HOUR*, *MINUTE*, or *SECOND*. The return value is of the datatype `interval`.

Function Name	Description
`to_dsinterval(string)`	Converts a string in format `DD HH:MM:SS` into a `INTERVAL DAY TO SECOND` data type. The `DD` indicates the number of days between 0 to 99. The `HH:MM:SS` indicates the number of hours, minutes and seconds in the interval from 0:0:0 to 23:59:59.999999. The seconds part can accept upto six decimal places. The return value is of the datatype `interval`.

What are Math Functions?

The math functions allow you to perform various mathematical operations and calculations ranging from simple to complex.

The following math functions are supported in this release:

Function Name	Description
`IEEEremainder(value1,value2)`	Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard
`abs(value1)`	Returns the absolute value of a number
`acos(value1)`	Returns arc cosine of a value
`asin(value1)`	Returns arc sine of a value
`atan(value1)`	Returns arc tangent of a value
`atan2(arg1,arg2)`	Returns polar angle of a point (`arg2`, `arg1`)
`binomial(base,power)`	Returns binomial coefficient of the base raised to the specified power
`bitMaskWithBitsSetFromTo(x)`	BitMask with BitsSet (From, To)
`cbrt(value1)`	Returns cubic root of the specified value
`ceil(value1)`	Rounds to ceiling
`copySign(value1,value2)`	Returns the first floating-point argument with the sign of the second floating-point argument
`cos(value1)`	Returns cosine of a value
`cosh(value1)`	Returns cosine hyperbolic of a value
`exp(x)`	Returns exponent of a value
`expm1(x)`	More precise equivalent of `exp(x)`; Returns 1 when x is around zero
`factorial(value1)`	Returns factorial of a natural number
`floor(value1)`	Rounds to floor
`getExponent(value1)`	Returns the unbiased exponent used in the representation of a double
`getSeedAtRowColumn(value1,value2)`	Returns a deterministic seed as an integer from a (seemingly gigantic) matrix of predefined seeds

Function Name	Description
`hash(value1)`	Returns an integer hashcode for the specified double value
`hypot(value1,value2)`	Returns square root of sum of squares of the two arguments
`leastSignificantBit(value1)`	Returns the least significant 64 bits of this UUID's 128 bit value
`log(value1,value2)`	Calculates the log value of the given argument to the given base, where `value 1` is the value and `value 2` is the base
`log1(value1)`	Returns the natural logarithm of a number
`log10(value1)`	Calculates the log value of the given argument to base 10
`log2(value1)`	Calculates the log value of the given argument to base 2
`logFactorial(value1)`	Returns the natural logarithm (base e) of the factorial of its integer argument as a double
`longFactorial(value1)`	Returns the factorial of its integer argument (in the range k >= 0 && k < 21) as a long
`maximum(value1,value2)`	Returns the maximum of 2 arguments
`minimum(value1,value2)`	Returns the minimum of 2 arguments
`mod(value1,value2)`	Returns modulo of a number
`mosttSignificantBit(value1)`	Returns the most significant 64 bits of this UUID's 128 bit value
`nextAfter(value1,value2)`	Returns the floating-point number adjacent to the first argument in the direction of the second argument
`nextDown(value1)`	Returns the floating-point value adjacent to the input argument in the direction of negative infinity
`nextUp(value1)`	Returns the floating-point value adjacent to the input argument in the direction of positive infinity
`Pow(m,n)`	Returns m raised to the nth power
`rint(value1)`	Returns the double value that is closest in value to the argument and is equal to a mathematical integer
`round(value1)`	Rounds to the nearest integral value
`Scalb(d,scaleFactor)`	Returns d × 2scaleFactor rounded as if performed by a single correctly rounded floating-point multiply to a member of the double value set
`signum(value1)`	Returns signum of an argument as a double value
`sin(value1)`	Returns sine of a value
`sinh(value1)`	Returns sine hyperbolic of a value
`sqrt(value1)`	Returns square root of a value
`stirlingCorrection(value1)`	Returns the correction term of the Stirling approximation of the natural logarithm (base e) of the factorial of the integer argument as a double
`tan(value1)`	Returns tangent of a value
`tanh(value1)`	Returns tangent hyperbolic of a value
`toDegrees(value1)`	Converts the argument value to degrees

Function Name	Description
`toRadians(value1)`	Returns the measurement of the angle in radians
`ulp(value1)`	Returns the size of an ulp of the argument

What are Null-related Functions?

The following null-related functions are supported in this release:

Function Name	Description
`nvl(value1,value2)`	Replaces null with a value of the same type

What are Statistical Functions?

Statistical functions help you in calculating the statistics of different values.

The following statistical functions are supported in this release:

Function Name	Description
`beta1(value1,value2,value3)`	Returns the area from zero to `value3` under the beta density function
`betaComplemented(value1,value2,value3)`	Returns the area under the right hand tail (from `value3` to infinity) of the beta density function
`binomial2(value1,value2,value3)`	Returns the sum of the terms 0 through `value1` of the Binomial probability density. All arguments must be positive.
`binomialComplemented(value1,value2,value3)`	Returns the sum of the terms `value1+1` through `value2` of the binomial probability density. All arguments must be positive.
`chiSquare(value1,value2)`	Returns the area under the left hand tail (from 0 to `value2`) of the chi square probability density function with `value1` degrees of freedom. The arguments must both be positive.
`chiSquareComplemented(value1,value2)`	Returns the area under the right hand tail (from `value2` to infinity) of the chi square probability density function with `value1` degrees of freedom. The arguments must both be positive.
`errorFunction(value1)`	Returns the error function of the normal distribution
`errorFunctionComplemented(value1)`	Returns the complementary error function of the normal distribution
`gamma(value1,value2,value3)`	Returns the gamma function of the arguments
`gammaComplemented(value1,value2,value3)`	Returns the integral from `value3` to infinity of the gamma probability density function
`incompleteBeta(value1,value2,value3)`	Returns the incomplete beta function evaluated from zero to `value3`

Function Name	Description
`incompleteGamma(value1,value2)`	Returns the incomplete gamma function
`incompleteGammaComplement(value1,value2)`	Returns the complemented incomplete gamma function
`logGamma(value1)`	Returns the natural logarithm of the gamma function
`negativeBinomial(value1,value2,value3)`	Returns the sum of the terms 0 through `value1` of the negative binomial distribution. All arguments must be positive.
`negativeBinomialComplemented(value1,value2,value3)`	Returns the sum of the terms `value1+1` to infinity of the negative binomial distribution. All arguments must be positive.
`normal(value1,value2,value3)`	Returns the area under the normal (Gaussian) probability density function, integrated from minus infinity to `value1` (assumes mean is zero, variance is one)
`normalInverse(value1)`	Returns the value for which the area under the normal (Gaussian) probability density function is equal to the argument `value1` (assumes mean is zero, variance is one)
`poisson(value1,value2)`	Returns the sum of the first `value1` terms of the Poisson distribution. The arguments must both be positive.
`poissonComplemented(value1,value2)`	Returns the sum of the terms `value1+1` to infinity of the poisson distribution
`studentT(value1,value2)`	Returns the integral from minus infinity to `value2` of the Student-t distribution with `value1` > 0 degrees of freedom
`studentTInverse(value1,value2)`	Returns the value, for which the area under the Student-t probability density function is equal to `1-value1/2`. The function uses the studentT function to determine the return value iteratively.

What are String Functions?

The following String functions are supported in this release:

Function Name	Description
`coalesce(value1,...)`	Returns the first non-null expression in the list. If all expressions evaluate to null, then the COALESCE function will return null
`concat(value1,...)`	Returns concatenation of values converted to strings
`indexof(string,match)`	Returns first index of `\'match\'` in `\'string\'`or 1 if not found
`initcap(value1)`	Returns a specified text expression, with the first letter of each word in uppercase and all other letters in lowercase
`length(value1)`	Returns the length of the specified string
`like(value1,value2)`	Returns a matching pattern
`lower(value1)`	Converts the given string to lower case

Function Name	Description
`lpad(value1,value2,value3)`	Pads the left side of a string with a specific set of characters (when `string1` is not null)
`ltrim(value1,value2)`	Removes all specified characters from the left hand side of a string
`replace(string,match,replacement)`	Replaces all `\'match\'` with `\'replacement\'` in `\'string\'`
`rpad(value1,value2,value3)`	Pads the right side of a string with a specific set of characters (when `string1` is not null)
`rtrim(value1,value2)`	Removes all specified characters from the right hand side of a string
`substr(string,from)`	Returns substring of a 'string' when indices are between 'from' (inclusive) and up to the end of the string
`substring(string,from,to)`	Returns substring of a \'string\' when indices are between \'from\' (inclusive) and \'to\' (exclusive)
`translate(value1,value2,value3)`	Replaces a sequence of characters in a string with another set of characters. However, it replaces a single character at a time.
`upper(value1)`	Converts given string to uppercase

14

Integrating Data

This topic only applies to Data Integration Platform Cloud Classic.

Common E-LT tasks such as, connecting to ODI Studio with VNC server, and creating repositories, data models, datastores, and mappings are discussed.

This part contains the following chapters:

- Connecting to ODI Studio with VNC Server
- Setting Up a Topology
- Creating and Using Data Models and Datastores
- Creating and Using Mappings
- Creating and Using Packages
- Using Scenarios
- Using Load Plans
- Running Integration Processes
- Debugging Integration Processes
- Monitoring Integration Processes
- Using Oracle Data Integrator Console

Connecting to ODI Studio with VNC Server

Connecting to ODI Studio with VNC Server requires obtaining the Public IP of your service instance, configuring PuTTy to connect to your VM and starting VNC server, and launching ODI Studio.

Ensure thatyou have your Oracle Cloud account, a Data Integration Platform Cloud subscription, Database Cloud and Data Integration Platform Cloud services provisioned, PuTTy or another SSH client, your private key in a PuTTy-compatible or OpenSSH format, and a VNC Viewer.

Follow the steps in this chapter or this tutorial.

This chapter includes the following sections:

- Obtaining the Public IP of your Data Integration Platform Cloud Service Instance
- Configuring PuTTy to Connect to Your VM and Start VNC Server
- Launching Oracle Data Integrator 12c Studio

Obtaining the Public IP of your Data Integration Platform Cloud Service Instance

Your service instance’s Public IP address is required to configure PuTTy.

To obtain the Public IP of your Data Integration Platform Cloud Service Instance:

1. On your My Services page, from the **Services** menu, select **Data Integration Platform Cloud Service**.
2. Select your Data Integration Platform Cloud service instance from the list of services.
3. In the Services Overview, locate the Resources section and copy the Public IP.

Configuring PuTTy to Connect to Your VM and Start VNC Server

PuTTy is configured to start a session and establish a connection with the VM. The Oracle user then starts the VNC server, and uses Tiger VNC Viewer to connect to the VM with the VNC Server.

Verifying Your Listening Port

Before you configure PuTTy to connect securely to your VM and start VNC server, you’ll need to verify the port that VNC Server is running on.

1. Open PuTTy and connect to your Data Integration Platform Cloud instance using its Public IP.
2. Switch to the opc user.

   ```
   $ sudo su - opc
   ```

3. Start VNC Server.

   ```
   $ vncserver -nolock
   ```

4. Verify that VNC Server is running.

   ```
   $ ps -ef |grep vnc
   ```

5. Verify the listening port.

   ```
   netstat -tulpn
   ```

   Take note of the port number.
6. (Optional) Use Telnet to test the connection.

   ```
   $ telnet 127.0.0.1:<listening_port_number>
   ```

To configure PuTTy to connect to your VM and start VNC server:

1. Open PuTTy.
2. Enter your service instance's Public IP address into the **Host Name** field.

3. In the **Category** pane, expand **Connection**, select **Data**, and then enter `opc` in the **Auto-login username** field.
4. In the **Category** pane, under **Connection**, expand **SSH**, and select **Tunnels**.
5. Enter the `<listening_port_number>` in the **Source port** field and `127.0.0.1:<listening_port_number>` into the **Destination** field, then click **Add**.
6. In the **Category** pane, under **SSH**, select **Auth**.
7. For **Private key file for authentication,** click **Browse** and locate your private key.
8. In the **Category** pane, select **Session,** and then enter a name for this session under **Saved Sessions** and click **Save.**
9. Click **Open** to establish the connection.
10. After you've successfully connected, enter the following command to switch to the `oracle` user:

    ```
    $ sudo su – oracle
    ```

11. Enter the following command to start VNC server:

    ```
    $ vncserver –nolock
    ```

    If you're prompted to enter a password to connect to the VNC server, enter a password and confirm the password. VNC server will start automatically.

    The SSH tunnel redirects the VNC output of your VM to localhost (127.0.0.1) on the appropriate listening port.

12. Open Tiger VNC Viewer, enter `127.0.0.1:<listening_port_number` in the **VNC server** field, and then click **Connect.**
13. Enter the password that you set in step 11 and click **OK.**

You're now successfully connected to your VM with VNC viewer.

Launching Oracle Data Integrator 12c Studio

Oracle Data Integrator 12c Studio is launched by entering its path in a Terminal window.

To launch Oracle Data Integrator 12c Studio, open a Terminal window and enter the following:

```
/u01/app/oracle/suite/odi_studio/odi/studio/odi.sh
```

Oracle Data Integrator Studio 12c starts and is ready to use.

Setting Up a Topology

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

- Setting Up the Topology
- Working with Big Data

See Also:

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

Setting Up the Topology

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

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

Note:

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

Creating a Context

To create a context:

1. In Topology Navigator expand the Contexts navigation tree.
2. Click **New context** in the navigation tree header.
3. Fill in the following fields:
   - **Name**: Name of the context, as it appears in the Oracle Data Integrator graphical interface.
   - **Code**: Code of the context, allowing a context to be referenced and identified among the different repositories.

  - **Password**: Password requested when the user requests switches to this context in a graphical interface. It is recommended to use a password for critical contexts (for example, contexts pointing to Production data).
  - Check **Default** if you want this context to be displayed by default in the different lists in Designer Navigator or Operator Navigator.

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

Creating a Data Server

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

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

Pre-requisites and Guidelines

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

Review the Technology Specific Requirements

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

- Installation of a JDBC Driver. See *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 14-1 Working with Big Data

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

Table 14-1 (Cont.) Working with Big Data

Task	Documentation
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

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
- Creating and Reverse-Engineering a Model
- Creating and Reverse-Engineering a Datastore
- Editing and Viewing a Datastore's Data

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.

Reverse-Engineering Fixed Files

Fixed files can be reversed engineered using a wizard into which the boundaries of the fixed attributes and their parameters can be defined.

1. Go to the **Attributes** tab the file datastore that has a fixed format.
2. Click the **Reverse Engineer** button. A window opens displaying the first records of your file.

3. Click on the ruler (above the file contents) to create markers delimiting the attributes. Right-click in the ruler to delete a marker.
4. Attributes are created with pre-generated names (C1, C2, and so on). You can edit the attribute name by clicking in the attribute header line (below the ruler).
5. In the properties panel (on the right), you can edit the parameters of the selected attribute.
6. You must set at least the **Attribute Name**, **Datatype** and **Length** for each attribute. Note that attribute names of File datastores cannot contain spaces.
7. Click **OK** when the attributes definition is complete to close the wizard.
8. Select **Save** from the File main menu.

Reverse-Engineering Delimited Files

Delimited files can be reversed engineered using a a built-in JDBC which analyzes the file to detect the attributes and reads the attribute names from the file header.

1. Go to the **Attributes** tab the file datastore that has a delimited format.
2. Click the **Reverse Engineer** button.
3. Oracle Data Integrator creates the list of attributes according to your file content. The attribute type and length are set to default values. Attribute names are pre-generated names (C1, C2, and so on) or names taken from the first Header line declared for this file.
4. Review and if needed modify the Attribute **Name**, **Datatype** and **Length** for each attribute. Note that attribute names of File datastores cannot contain spaces.
5. Select **Save** from the File main menu.

Reverse-Engineering COBOL Files

Fixed COBOL files structures are frequently described in Copybook files. Oracle Data Integrator can reverse-engineer the Copybook file structure into a datastore structure.

1. Go to the **Attributes** tab the file datastore that has a fixed format.
2. Click the **Reverse Engineer COBOL Copybook** button.
3. Fill in the following fields:
   - **File**: Location of the Copybook file.
   - **Character Set**: Copybook file character set.
   - **Description format** (EBCDIC or ASCII): Copybook file format
   - **Data format** (EBCDIC or ASCII): Data file format.
4. Click **OK**. The attributes described in the Copybook are reverse-engineered and appear in the attribute list.
5. Select **Save** from the File main menu.

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.

Keys

To create a key for a datastore:

1. In the Designer Navigator, expand in the **Model** tree the model and then the datastore into which you want to add the key.
2. Select the **Constraints** node, right-click and select **New Key**.
3. Enter the **Name** for the constraint, and then select the **Key or Index Type**. Primary Keys and Alternate Keys can be checked and can act as an update key in an interface. Non-Unique Index are used mainly for performance reasons.
4. In the **Attributes** tab, select the list of attributes that belong to this key.
5. In the **Control** tab, select whether this constraint should be checked by default in a **Static** or **Flow** check.
6. By clicking the **Check** button, you can retrieve the number of records that do not respect this constraint.
7. Select **Save** from the File main menu.

References

To create a reference between two datastores:

1. In the Designer Navigator, expand in the **Model** tree the model and then one of the datastores into which you want to add the reference.
2. Select the **Constraints** node, right-click and select **New Reference**.
3. Enter the **Name** for the constraint, and then select the **Type** for the reference. In a **User Reference** the two datastores are linked based on attribute equality. In a **Complex User Reference** any expression can be used to link the two datastores. A **Database Reference** is a reference based on attribute equality that has been reverse-engineered from a database engine.
4. If you want to reference a datastore that exists in a model, select the **Model** and the **Table** that you want to link to the current datastore.

5. If you want to link a table that does not exist in a model, leave the **Model** and **Table** fields undefined, and set the **Catalog**, **Schema** and **Table** names to identify your datastore.
6. If you are defining a User or Database reference, in the **Attributes** tab, define the matching attributes from the two linked datastores.
7. If you are defining a Complex User reference, enter in the **Expression** tab the expression that relates attributes from the two linked datastores.
8. In the **Control** tab, select whether this constraint should be checked by default in a **Static** or **Flow** check.
9. By clicking the **Check** button, you can retrieve the number of records that respect or do not respect this constraint.
10. Select **Save** from the File main menu.

Conditions

To create a condition for a datastore:

1. In the Designer Navigator, expand in the **Model** tree the model and then one of the datastores into which you want to add the condition.
2. Select the **Constraints** node, right-click and select **New Condition**.
3. Enter the **Name** for the constraint, and then select the **Type** for the condition. An **Oracle Data Integrator Condition** is a condition that exists only in the model and does not exist in the database. A **Database Condition** is a condition that is defined in the database and has been reverse-engineered.
4. In the **Where** field enter the expression that implements the condition. This expression is a SQL WHERE expression that valid records should respect.
5. Type in the **Message** field the error message for this constraint.
6. In the **Control** tab, select whether this constraint should be checked by default in a **Static** or **Flow** check.
7. By clicking the **Check** button, you can retrieve the number of records that do not respect this constraint.
8. Select **Save** from the File main menu.

Mandatory Attributes

To define mandatory attributes for a datastore:

1. In the Designer Navigator, expand in the **Model** tree the model containing the datastores.
2. Double-click the datastore containing the attribute that must be set as mandatory. The **Datastore** Editor appears.
3. In the **Attributes** tab, check the **Not Null** field for each attribute that is mandatory.
4. Select **Save** from the File main menu.

Filter

To add a filter to a datastore:

1. In the Designer Navigator, expand in the **Model** tree the model and then one of the datastores into which you want to add the filter.
2. Select the **Filter** node, right-click and select **New Condition**.
3. Enter the **Name** for the filter.
4. In the **Where** field enter the expression that implements the filter. This expression is a SQL WHERE expression used to filter source records.
5. In the **Control** tab, check **Filter Active for Static Control** if you want data from this table to be filtered prior to checking it a static control.
6. Select **Save** from the File main menu.

Editing and Viewing a Datastore's Data

> **Note:**
>
> Viewing or editing the data is not supported for remote files.

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.

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

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 14-2 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.

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.
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 14-3 SALES

YEAR	QUARTER	SALES
2010	Q1	10.5
2010	Q2	11.4
2010	Q3	9.5
2010	Q4	8.7
2011	Q1	9.5
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 14-4 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.
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 14-5 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 14-6 UNPIVOTED DATA

YEAR	QUARTER	SALES
2010	Q1	10.5
2010	Q2	11.4
2010	Q3	9.5
2010	Q4	8.7
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 14-7 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 14-8 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

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 14-9 Keyboard Navigation for Common Tasks

Navigation	Task
Arrow keys	Navigate: move one cell up, down, left, or right

Table 14-9 (Cont.) Keyboard Navigation for Common Tasks

Navigation	Task
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:

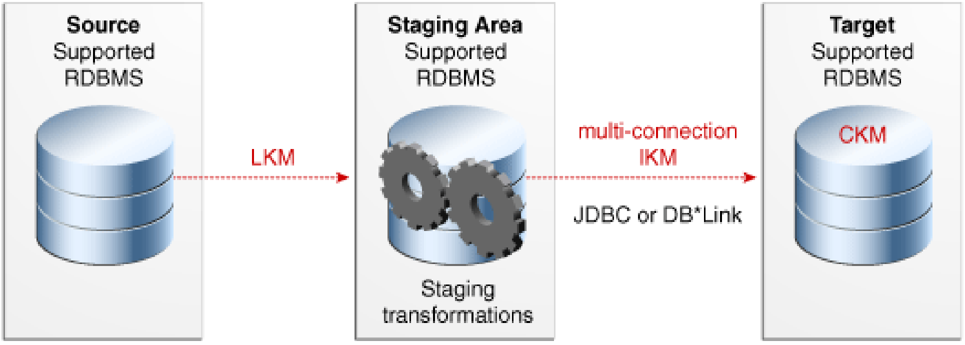
- Using a Multi-connection IKM
- Using an LKM and a mono-connection IKM

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 14-1 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:

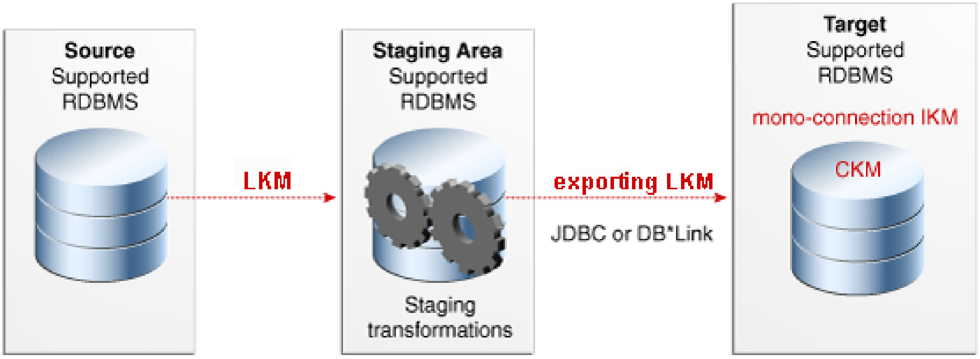
1. Create a mapping using the standard procedure as described in Creating a Mapping. This section describes only the ETL-style specific steps.
2. In the Physical tab of the Mapping Editor, select a physical mapping design by clicking the desired physical mapping design tab and clicking on the diagram 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 14-2 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.

Creating and Using Packages

This chapter includes the following sections:

- Introduction to Packages
- Creating a new Package
- Working with Steps
- Defining the Sequence of Steps
- Running a Package

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.

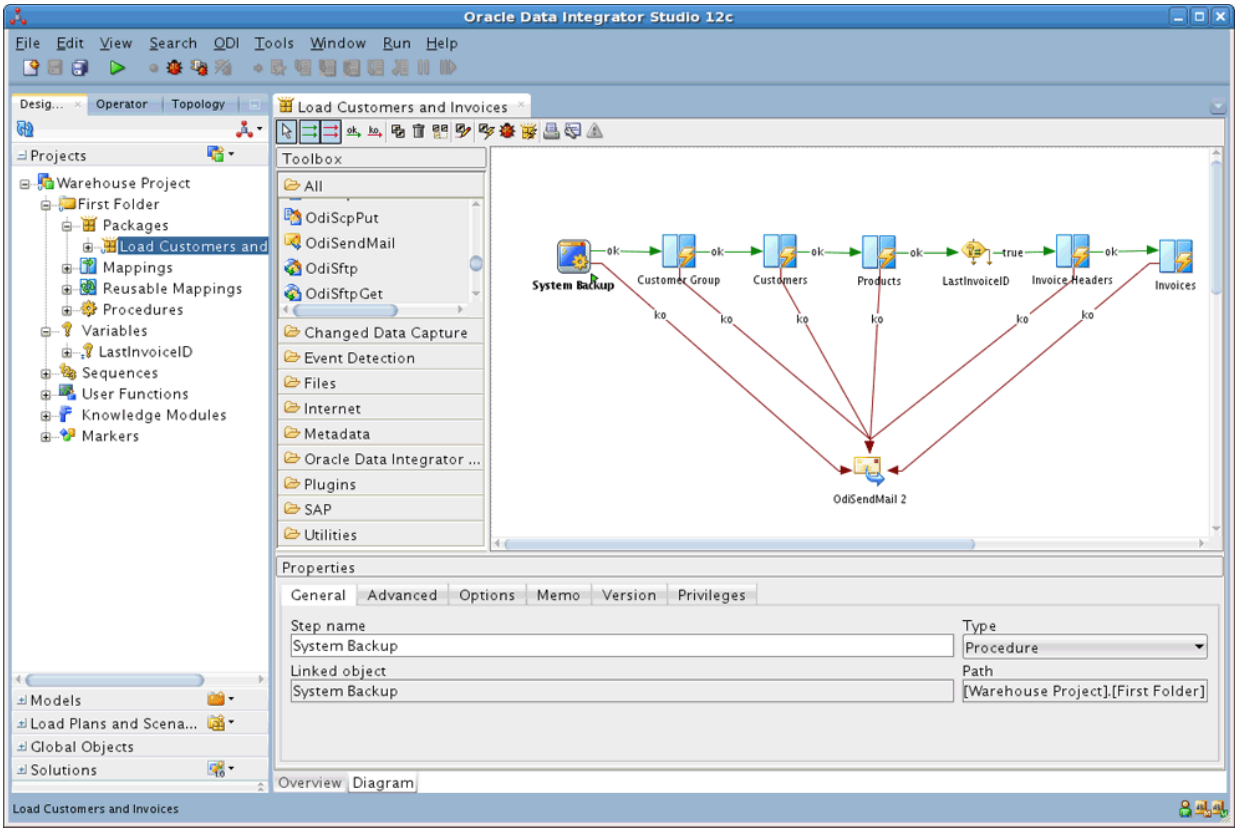
Table 14-10 Step Types

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

Table 14-10 (Cont.) Step Types

Type	Description	See Section
Models, Sub-models, and Datastores	Performs journalizing, static check or reverse-engineering operations on these objects	Adding a Model, Sub-Model or Datastore

Figure 14-3 Sample Package



Step name: System Backup

Type: Procedure

Linked object: System Backup

Path: [Warehouse Project].[First Folder]

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.

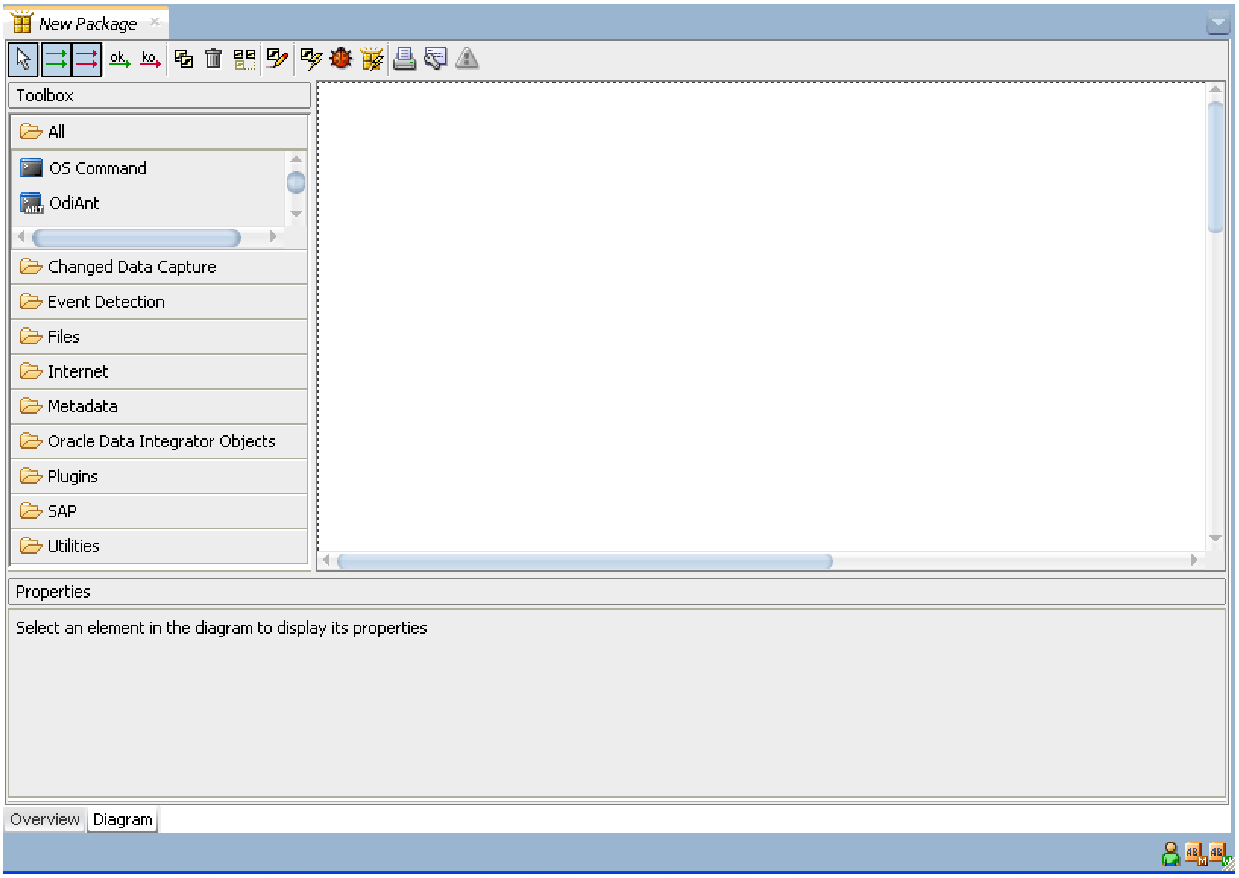
Creating a Package consists of the following main steps:

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 14-4 Package editor



Select an element in the diagram to display its properties

Table 14-11 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.
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.

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.

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.

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.

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 14-12 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.

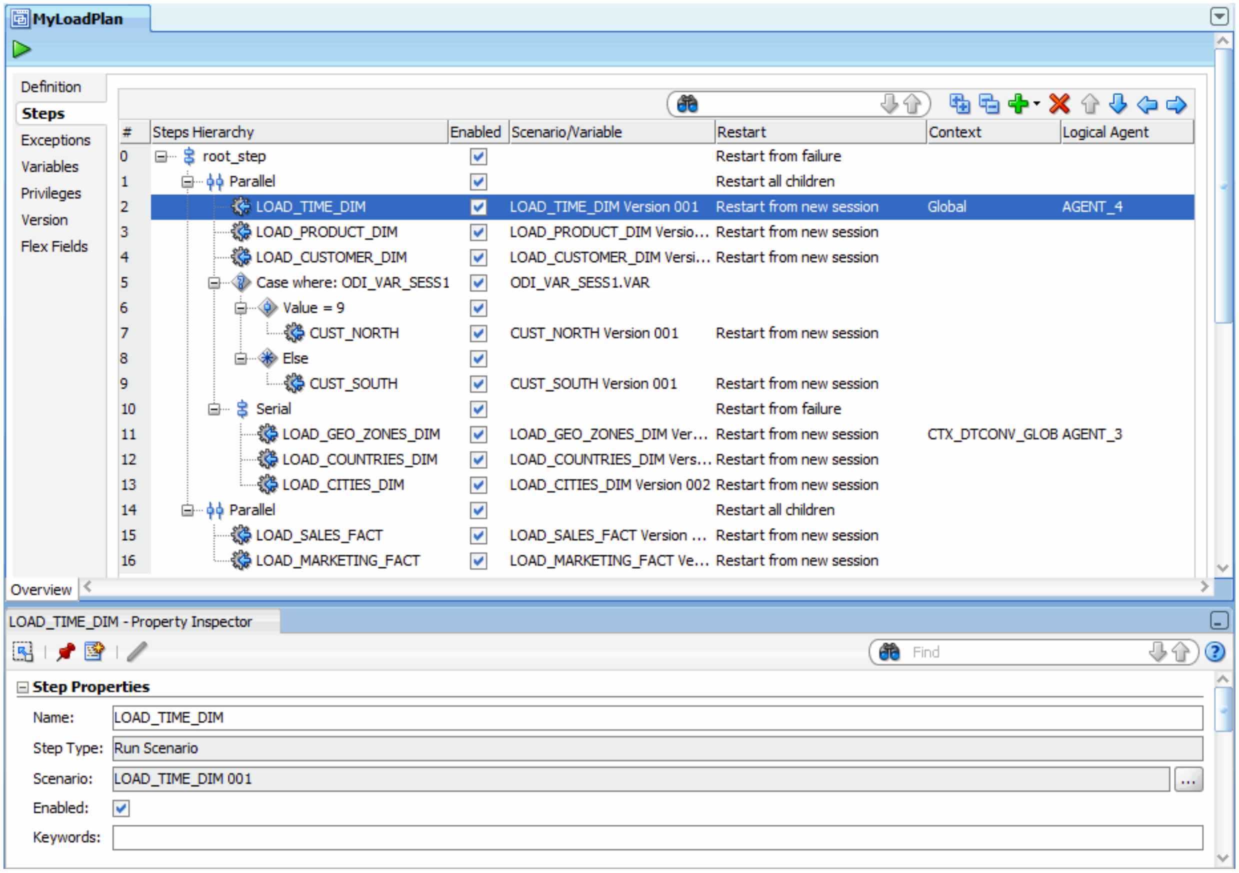
Table 14-12 (Cont.) Load Plan Steps

Type	Description	Possible 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 14-5 Sample Load Plan



#	Steps Hierarchy	Enabled	Scenario/Variable	Restart	Context	Logical Agent
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	AGENT_4
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_DTCONV_GLOB	AGENT_3
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		

Step Properties

Name: LOAD_TIME_DIM

Step Type: Run Scenario

Scenario: LOAD_TIME_DIM 001

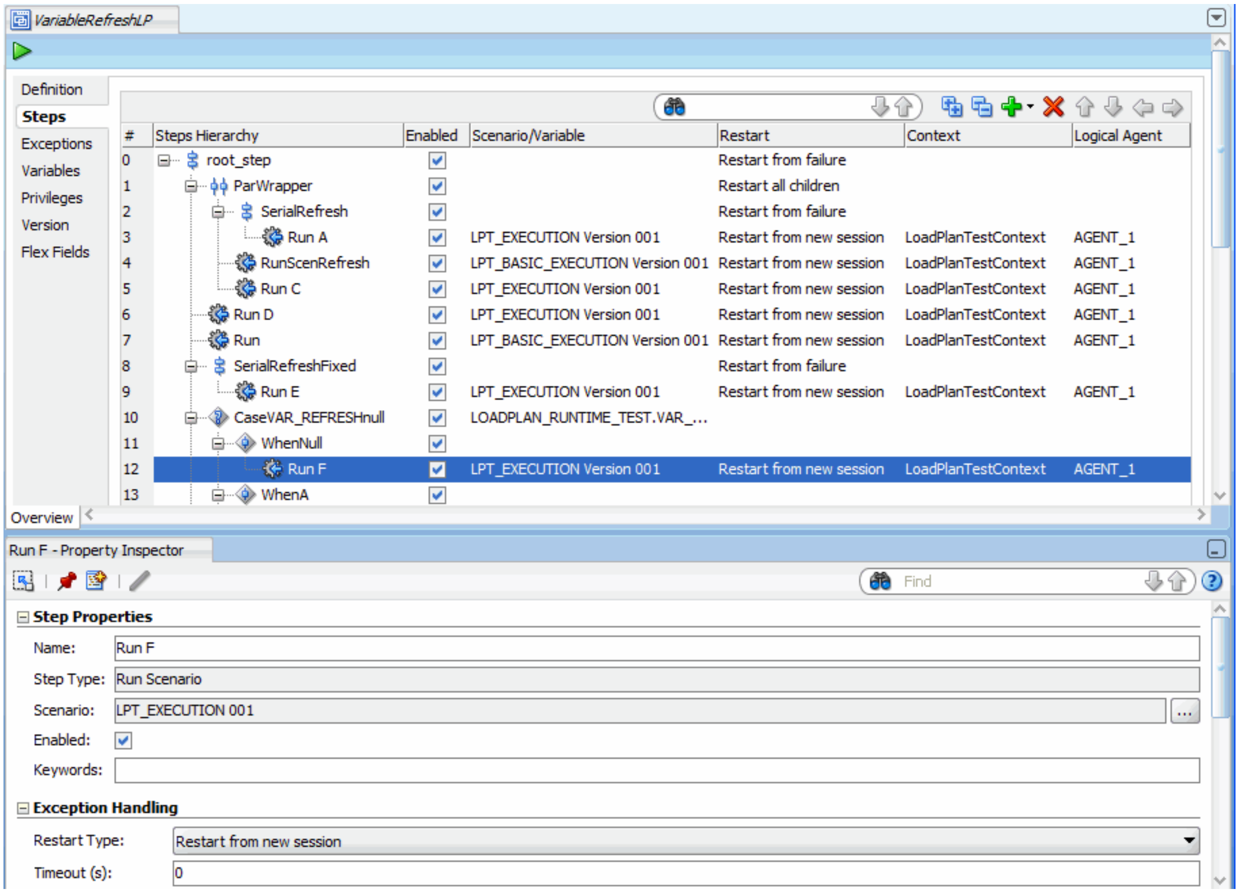
Enabled: ☑

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 14-6 Steps Tab of the Load Pan Editor



#	Steps Hierarchy	Enabled	Scenario/Variable	Restart	Context	Logical Agent
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	LoadPlanTestContext	AGENT_1
4	RunScenRefresh	☑	LPT_BASIC_EXECUTION Version 001	Restart from new session	LoadPlanTestContext	AGENT_1
5	Run C	☑	LPT_EXECUTION Version 001	Restart from new session	LoadPlanTestContext	AGENT_1
6	Run D	☑	LPT_EXECUTION Version 001	Restart from new session	LoadPlanTestContext	AGENT_1
7	Run	☑	LPT_BASIC_EXECUTION Version 001	Restart from new session	LoadPlanTestContext	AGENT_1
8	SerialRefreshFixed	☑		Restart from failure		
9	Run E	☑	LPT_EXECUTION Version 001	Restart from new session	LoadPlanTestContext	AGENT_1
10	CaseVAR_REFRESHnull	☑	LOADPLAN_RUNTIME_TEST.VAR_...			
11	WhenNull	☑				
12	Run F	☑	LPT_EXECUTION Version 001	Restart from new session	LoadPlanTestContext	AGENT_1
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 14-13 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 14-14 Add Step Wizard Actions

Step Type	Description and Action Required
Serial or Parallel step	Enter a **Step Name** for the new Load Plan step.
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.

Table 14-14 (Cont.) Add Step Wizard Actions

Step Type	Description and Action Required
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.
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 14-15 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 14-16 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.

Exporting Load Plans

It is possible to export a single Load Plan or several Load Plans at once.

Exporting one single Load Plan follows the standard procedure described in the section Exporting one ODI Object in *Developing Integration Projects with Oracle Data Integrator*.

For more information on exporting several Load Plans at once, see the section Export Multiple ODI Objects in *Developing Integration Projects with Oracle Data Integrator*.

Note that when you export a Load Plan and you select **Export child objects**, all its child steps, schedules, and variables are also exported.

> **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. How to export scenarios is described in the section Exporting Scenarios in *Developing Integration Projects with Oracle Data Integrator*.

Importing Load Plans

Importing a Load Plan in a development repository is performed via Designer or Operator Navigator. With an execution repository, only Operator Navigator is available for this purpose.

The Load Plan import uses the standard object import method. See the section Importing Objects in *Developing Integration Projects with Oracle Data Integrator* for more information.

> **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 imported separately.

Versioning Load Plans

Load Plans 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.

Running Integration Processes

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

- Understanding ODI Executions
- Executing Mappings, Procedures, Packages and Model Operations

- Executing a Scenario
- Restarting a Session
- Stopping a Session
- Executing a Load Plan
- Restarting a Load Plan Run
- Stopping a Load Plan Run
- Scheduling Scenarios and Load Plans
- Simulating an Execution
- Managing Executions Using Web Services

> **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 14-17.

Table 14-17 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)**.

Table 14-17 (Cont.) Execution Parameters

Properties	Description
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 14-17 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 14-18 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 14-18 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 14-18 (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 14-19 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 14-19 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 14-20 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 14-20 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 14-21 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 14-21 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 14-21 (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 14-22 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 14-22 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.

Table 14-22 (Cont.) Restartloadplan Command Parameters

Parameters	Description
`[-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 14-23 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 14-23 Stoploadplan Command Parameters

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

Note:

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

Scheduling Scenarios and Load Plans

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

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

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

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

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

For example:

- **Schedule 1**: Every Thursday at 9 PM, once only.
- **Schedule 2**: Every day from 8 am to 12 noon, repeated every 5 seconds.

- **Schedule 3**: Every day from 2 PM to 6 PM, repeated every 5 seconds, with a maximum cycle duration of 5 hours.

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.

Properties	Description
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 14-1 Using the View Schedule Dialog

The schedule is displayed in form of a Gantt diagram. Table 14-24 lists the details of the **Schedule** dialog.

Table 14-24 Scheduling Details

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

Table 14-24 (Cont.) Scheduling Details

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

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

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

Scheduling a Scenario or a Load Plan with an External Scheduler

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

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

For more information, see:

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

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

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

Simulating an Execution

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

Note:

No session is created in the log when the execution is started in simulation mode.

To simulate an execution:

1. In the **Project** view of the Designer Navigator, select the object you want to execute.
2. Right-click and select **Run**.
3. In the **Run** dialog, set the execution parameters and select **Simulation**. See Table 14-17 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 `getStopLoadPlanStatus` 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 14-25 lists the different parameters of this command, both mandatory and optional.

Table 14-25 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).

Table 14-25 (Cont.) Startscenremote command Parameters

Parameters	Description
`<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.
`-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 14-26 lists the different parameters of this command, both mandatory and optional.

Table 14-26 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.
`-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.

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.

Icon	Command	Description
	Get Data	Inserts the SQL code of the currently selected task into the SQL command field of the Debug Data window. Both Source Task Data and Target Task Data windows are populated.
	Step into	Steps to the beginning of the first child node of the currently selected node. If the currently selected node does not have a child, this is disabled.
	Run to Task End	Runs to the end of the currently selected task. If the task has children, execute all children until the end of the task is reached. This is disabled if it is not at the beginning of a task.
	Run to Next Task	Runs to the beginning of the next task. If at the last task in a list of tasks, go to the end of the parent task.
	Run to Step End	Runs to the end of the current step. Executes all tasks until the end of the step is reached.
	Run to Next Step	Runs to the beginning of the next step.
⏸	Pause	Suspends the current execution. The execution can be continued by stepping through the steps/tasks, or by resuming execution.
▶	Resume	Resumes execution at the current cursor and continues until the session is finished or a breakpoint is reached.

Differences between Debugging Packages, Mappings, and Procedures

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

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

- In a package, you can set breakpoints on a step.

Starting a Debugging Session

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

Starting a Session from the Toolbar

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

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

   An information message indicates that the session is started.

Starting a Session from the Navigator Tree

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

1. In the navigator tree, select a mapping, package, procedure, or scenario.
2. Right-click and select **Debug**.
3. In the **Debug** window, configure options described in Debug Session Options.
4. Click **OK**.

   An information message indicates that the session has started.

Starting a Session from the Main Menu

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

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

   An information message indicates that the session has started.

Starting a Session for a Scenario

To start a debug session for a scenario:

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

3. In the **Debug** window, configure options described in Debug Session Options.
4. Click **OK**.

   An information message indicates that the session has started.

Connecting to a Running Debugging Session

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

To connect to a session:

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

Built-in Agents and External Agents

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

Debug Session Lifecycle

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

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

Debug Session Options

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

Suspend Before the First Task

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

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

The setting is checked by default.

Associate to Current Client

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

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

The setting is checked by default.

Delay Before Connect

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

The default is 5 seconds.

Debug Descendant Sessions

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

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

The setting is unchecked by default.

Break On Error

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

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

The setting is unchecked by default.

Stepping through a Blueprint in the Session Editor

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

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

Steps and Tasks

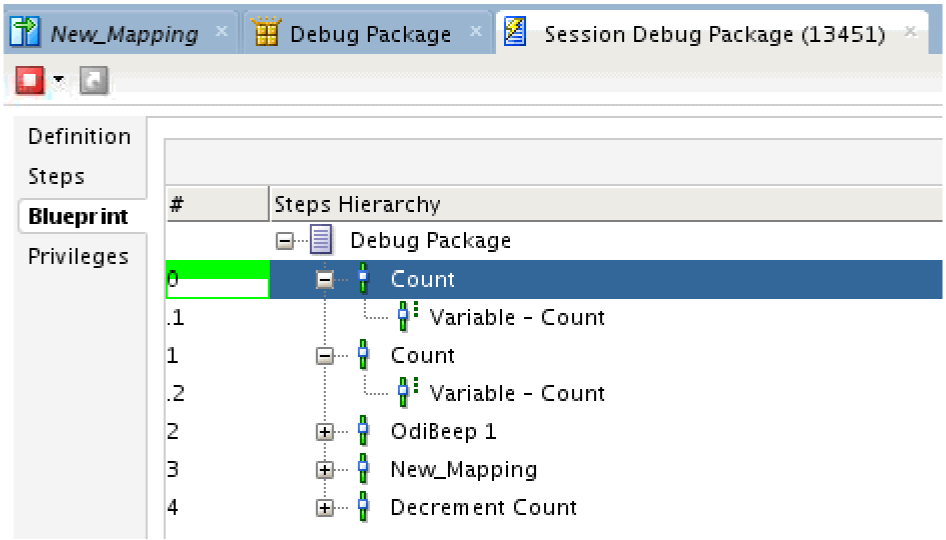
Each row in a blueprint represents a step or task. Tasks in a blueprint can be container tasks, and can execute serially or in parallel. Tasks inside a container task are leaf nodes that get executed. Sessions based on mappings have only one step, while sessions based on packages can have multiple steps.

Using the Debugging Cursor

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

Figure 14-7 shows a cursor position before the execution of a line:

Figure 14-7 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.

Step Into

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

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

Pause

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

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

Resume

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

Run to Task End

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

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

Run to Next Task

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

Run to Step End

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

Run to Next Step

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

Multiple Cursors

You can use multiple cursors for in-session parallelism.

Special Behavior

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

Using Breakpoints

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

About the Debug Breakpoints Window

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

About Design vs. Runtime Breakpoints

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

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

blueprint it is associated with. You can add design breakpoints in the package editor.

Placing Breakpoints

To add a breakpoint:

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

Editing Breakpoints

To edit a breakpoint:

1. Select **Run** > **Breakpoint** > **Edit Breakpoint**.
2. The Breakpoint Properties window appears.
3. Set the following properties.
   - **Enabled**: checked if the breakpoint is enabled.
     - if checked, the breakpoint is active and suspends execution.
     - if unchecked, the breakpoint is ignored during execution.
   - **Suspend after executing the task**:
     - if checked, the breakpoint suspends after executing the task or step.
     - if unchecked, the breakpoint suspends before executing the task or step.
   - **Pass count**: the number of times the breakpoint has to be passed before suspending execution; 0 suspends the execution on the initial pass, 1 passes the breakpoint once before suspending execution.
4. Click **OK**.

Removing Breakpoints

To remove a single breakpoint:

- Right-click on the step or task and select **Remove Breakpoint**.

To remove all breakpoints (available only from the breakpoint window):

- Select **Run** > **Breakpoint** > **Remove All**.

Enabling and Disabling Breakpoints

To enable or disable a single breakpoint, select:

- **Run** > **Breakpoint** > **Enable**, or,
- **Run** > **Breakpoint** > **Disable**.

To enable or disable all breakpoints (available only from the breakpoint window), select:

- **Run** > **Breakpoint** > **Enable All**, or,
- **Run** > **Breakpoint** > **Disable All**.

Pass Count

The pass count indicates the number of times the breakpoint has to be passed before suspending execution; 0 suspends the execution on every pass; 1 suspends the execution only on the first pass; 5 suspends the execution only on the fifth pass.

Debugging Data

Use the Debug Data tab to query data in the context of the current debugger task.

The data window has two tabs, Source Task Data and Target Task Data, to query the data servers associated with the source and the target for the current task. The **Get Data** command in the main toolbar inserts the SQL code of the current task into both Source Task Data and Target Task Data fields. The window provides a field to enter a SQL command and execute it by clicking the **Run SQL Code** button.

SQL commands use the transaction context of the current session, uncommitted data can be queried. If there are multiple tasks at debug execution in the session, the commands use the context of the task synced to the data window.

Get Data

Get Data inserts the SQL code of the currently selected task into the SQL command field of the Debug Data window. Both the Source Task Data and Target Task Data windows are populated.

The **Get Data** command associates with the original cursor line; if you keep stepping the data window remains attached to the original cursor line. You must press **Get Data** again to go to the new current cursor line.

Source Task Data

Typically, the task code for Source command contains a Select query to retrieve data from source. By default, this query is displayed in the Source Task Data field. You must review and modify the query before executing, as the content may be inappropriate to execute unchanged.

Target Task Data

The Target Task Data typically contains a DML operation such as Insert or Update. Oracle Data Integrator attempts to parse the target SQL to identify the target table name and provide a simple query, such as:

```
select * from <parsedTargetTableName>
```

If Oracle Data Integrator is unable to parse the target SQL to identify the target table name, a suggested query appears, such as:

```
select * from <SpecifyTargetTableName>
```

You must review and modify the query before executing, as the content may be inappropriate to execute unchanged.

Run SQL Code

You are required to execute **Get Data** before doing a **Run SQL Code**. You can run queries to view or select data, but you cannot not apply changes to the data.

Editing SQL Code

To edit SQL code and query data:

1. Select the **Debug Data** tab in a session.
2. Select **Get Data** to insert the SQL code of the current task into the Source Task Data and Target Task Data fields.
3. Edit the SQL command. You can use a `select` statement to query the data on the target uncommitted transaction.
4. Click **Run SQL Code**.

Debugging Variables

Use the **Debug Variables** tab to view all the variables used in the current session, and see how a variable may change as you step through the steps and tasks. You can change the value of a variable to affect the execution of the session.

You can view the following information for each variable:

Properties	Description
Name	Name of the variable.
Value	Value of the variable. Can be modified to change execution.
Type	Type of the variable.

Modifying Variables

To modify a variable:

1. Set breakpoints, if required. See Using Breakpoints.
2. Position the cursor in the blueprint.
3. In the **Debug Variables** tab, select the variable whose value you want to modify.
4. Start or resume the execution of the session.

Debugging Threads

Use the **Debug Threads** tab to see all threads for the connected Session Editor in focus. The icon of the thread shows whether the thread is suspended or executing.

Go To Source

The **Go To Source** context menu command on each thread jumps to the thread location inside the blueprint.

Managing Debugging Sessions

You can see a list of all active debug sessions in the Debug Session window by selecting **Window** > **Debugger** > **Debug Sessions**.

From the Debug Sessions window, you can connect or disconnect the listed sessions.

Properties	Description
Agent	Agent executing the session.
Session Number	ID of the session.
Session Name	Name of the session.
Session Connected	Shows whether the session is connected or not connected to the debugger client.

Stop Normal and Stop Immediate

You can disconnect a session normally or immediately:

1. In the Debug Sessions window, right-click a session in the list and select **Disconnect Debug Session**.

   The session becomes active in the Studio.
2. Select the **Disconnect** button in the upper left of the session window, and choose:
   - **Stop Normal**: The session is stopped once the current task is finished.
   - **Stop Immediate**: The current task is immediately interrupted and the session is stopped. This mode allows to stop long-running tasks, as for example long SQL statements before they complete.
3. Confirm the disconnection by clicking **OK**.
4. When the session is stopped, you see the message 'Session debugging completed'. The session is removed from the Debug Sessions list.

See Stopping a Session.

Restart Execution

In the session window, you can restart a stopped session.

To restart a session:

1. In the upper left of the session window, select **Restart Execution**.

   The Restart Execution window appears.
2. To select the agent to execute the session, select one of the following options:
   - **Use the Previous Agent**: **<agent name>** to use the agent that was used to execute a previous session.
   - **Choose another Agent**: to select the agent that you want to use to execute the session. Select **Internal** to use the ODI Studio built-in agent.
3. Select the Log Level. Log level 6 has the same behavior as log level 5, but with the addition of variable and sequence tracking.

4. Click **OK** to restart the indicated session and to close the dialog.

   You see an informational message, 'Session Launched', and the session is added to the Debug Sessions list.

To restart a failed session in debug mode:

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

   An information message indicates that the session has restarted.

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 14-27.

Table 14-27 Operator Navigator Toolbar Menu Items

Icon	Menu Item	Description
🔄	Refresh	Click **Refresh** to refresh the trees in the Operator Navigator.
⏷ ⏷	Filter Filter activated	Click **Filter** to define the filters for the sessions to display in Operator Navigator.
⟳	Auto Refresh	Click **Auto Refresh** to refresh automatically the trees in the Operator Navigator.
🔍	Connect Navigator	Click **Connect Navigator** to access the Operator Navigator toolbar menu. Through the Operator Navigator toolbar menu you can: • Import a scenario • Import and export the log • Perform multiple exports • Purge the log • Display the scheduling information • Clean stale sessions • Remove temporary objects

Scenarios

A *scenario* is designed to put a source component (mapping, package, procedure, variable) into production. A scenario results from the generation of code (SQL, shell, etc.) for this component.

When a scenario is executed, it creates a *Session*.

Scenarios are imported into production environment and can be organized into *Load Plan and Scenario* folders. See Managing Scenarios and Load Plansfor more details.

Sessions

In Oracle Data Integrator, an execution results in a *Session*. Sessions are viewed and managed in Operator Navigator.

A *session* is an execution (of a scenario, a mapping, a package or a procedure, and so forth) undertaken by an execution agent. A session is made up of *steps* which are themselves made up of *tasks*.

A *step* is the unit of execution found between a task and a session. It corresponds to a step in a package or in a scenario. When executing a mapping or a single variable, for example, the resulting session has only one step.

Two special steps called *Command On Connect* and *Command On Disconnect* are created if you have set up On Connect and Disconnect commands on data servers used in the session. See Setting Up On Connect/Disconnect Commandsfor more information.

The *task* is the smallest execution unit. It corresponds to a command in a KM, a procedure, and so forth.

Sessions can be grouped into *Session folders*. Session folders automatically group sessions that were launched with certain keywords. Refer to Organizing the Log with Session Foldersfor more information.

Note that certain privileges are required for you to stop a session or clean stale sessions. These privileges are listed below:

You can stop the session if you have one of the following Sessions privileges:

- You are a Supervisor user.
- You have started the session and you are the owner of the session.
- You have been explicitly assigned the Stop Immediate or the Stop Normal privileges. Note that you can view these method objects in the Security Navigator by expanding the Objects tree and expanding the Session Object node of a given profile.

You can clean stale sessions if you have one of the following Sessions privileges:

- You are a Supervisor user.
- You have been explicitly assigned the Clean Stale Sessions privileges. Note that you can view this method object in the Security Navigator by expanding the Objects tree and expanding the Agent Object node of a given profile.

You can also stop a session or clean stale sessions if the OPERATOR or NG OPERATOR profile is assigned to you.

Load Plans

A *Load Plan* is the most course-grained type of executable object in Oracle Data Integrator, used to organize and run finer-grained objects. It uses *Scenarios* 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 14-28 lists the six possible status values:

Table 14-28 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.

Table 14-28 (Cont.) Status Values

Status Name	Status Icon for Sessions	Status Icon for Load Plans	Status Description
Error	⊗	⊠	The Load Plan, session, step or task has terminated due to an error.
Running	▶	⚡	The Load Plan, session, step or task is being executed.
Waiting	🕒	W	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 14-29.

Table 14-29 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.
Serial Task		The children tasks are executed in a sequential order. The serial task is completed and marked to DONE status when all its children tasks have completed successfully. It is considered failed and marked with error status when the first child task completes in error status.
Parallel Task		The children tasks are executed concurrently. The parallel task is considered completed and marked with DONE status when all its children tasks have completed successfully. It is considered failed and marked with ERROR status if any of its child tasks has failed.

Monitoring Executions Results

In Oracle Data Integrator, an execution results in a *session* or in a *Load Plan run* if a Load Plan is executed. A *session* is made up of steps which are made up of tasks. Sessions are viewed and managed in Operator Navigator.

Load Plan runs appear in the Operator Navigator. To review the steps of a Load Plan run, you open the editor for this run. The sessions attached to a Load Plan appear with the rest of the sessions in the Operator Navigator.

Monitoring Sessions

To monitor your sessions:

1. In the Operator Navigator, expand the Session List navigation tree.
2. Expand the All Executions node and click **Refresh** in the Navigator toolbar.
3. Optionally, activate a Filter to reduce the number of visible sessions. For more information, see Filtering Sessions.
4. Review in the list of sessions the status of your session(s).

Monitoring Load Plan Runs

To monitor your Load Plan runs:

1. In the Operator Navigator, expand the Load Plan Executions navigation tree.
2. Expand the All Executions node and click **Refresh** in the Navigator toolbar.
3. Review in the list the status of your Load Plan run.
4. Double-click this Load Plan run to open the Load Plan Run editor.

5. In the Load Plan Run editor, select the Steps tab.
6. Review the state of the Load Plan steps. On this tab, you can perform the following tasks:
   - Click **Refresh** in the Editor toolbar to update the content of the table.
   - For the Run Scenario steps, you can click in the Session ID column to open the session started by this Load Plan for this step.

Handling Failed Sessions

When your session ends in error or with a warning, you can analyze the error in Operator Navigator.

To analyze an error:

1. In the Operator Navigator, identify the session, the step and the task in error.
2. Double click the task in error. The Task editor opens.
3. On the Definition tab in the Execution Statistics section, the return code and message give the error that stopped the session. The metrics about the performance of the loading task and any configuration data that have an impact on performance are displayed in the Execution Details section.
4. On the Code tab, the source and target code for the task is displayed and can be reviewed and edited.

   Optionally, click **Show/Hide Values** to display the code with resolved variable and sequence values. Note that:

   - If the variable values are shown, the code becomes read-only. You are now able to track variable values.
   - Variables used as passwords are never displayed.

   See the Tracking Variables and Sequences section in *Developing Integration Projects with Oracle Data Integrator* for more information.
5. On the Connection tab, you can review the source and target connections against which the code is executed.

You can fix the code of the command in the Code tab and apply your changes. Restarting a session (see Restarting a Session) is possible after performing this action. The session will restart from the task in error.

> **Note:**
>
> Fixing the code in the session's task does not fix the source object that was executed (mapping, procedure, package or scenario). This source object must be fixed in Designer Navigator and the scenario (if any) must be regenerated. Modifying the code within the session is useful for debugging issues.

WARNING:

When a session fails, all connections and transactions to the source and target systems are rolled back. As a consequence, uncommitted statements on transactions are not applied.

Reviewing Successful Sessions

When your session ends successfully, you can view the changes performed in Operator Navigator. These changes include record statistics such as the number of inserts, updates, deletes, errors, and the total number of rows as well as execution statistics indicating start and end time of the execution, the duration in seconds, the return code, and the message (if any).

Session level statistics aggregate the statistics of all the steps of this session, and each step's statistics aggregate the statistics of all the tasks within this step.

To review the execution statistics:

1. In the Operator Navigator, identify the session, the step, or the task to review.
2. Double click the session, the step, or the task. The corresponding editor opens.
3. The record and execution statistics are displayed on the Definition tab.

Record Statistics

Properties	Description
No. of Inserts	Number of rows inserted during the session/step/task.
No. of Updates	Number of rows updated during the session/step/task.
No. of Deletes	Number of rows deleted during the session/step/task.
No. of Errors	Number of rows in error in the session/step/task.
No. of Rows	Total number of rows handled during this session/step/task.

Execution Statistics

Properties	Description
Start	Start date and time of execution of the session/step/task.
End	End date and time of execution of the session/step/task.
Duration (seconds)	The time taken for execution of the session/step/task.
Return code	Return code for the session/step/task.
Execution Details	The metrics about the performance of the loading task and any configuration data that have an impact on performance.

For session steps in which a mapping has been executed or a datastore check has been performed, the target table details are displayed. Also, for session steps in which a mapping has been executed, the Knowledge Modules used in the mapping are displayed in the Knowledge Module Details section of the Session Step Editor.

If tracking is enabled for a session, information regarding the variable or sequence is displayed in the Variable and Sequence Values section of the Session Step Editor and Session Task Editor.

You can view the options selected to limit concurrent executions of a session in the Concurrent Execution Controller section of the Session Editor. See the Controlling Concurrent Execution of Scenarios and Load Plans section in *Developing Integration Projects with Oracle Data Integrator* for more information.

Handling Failed Load Plans

When a Load Plan ends in error, review the sessions that have failed and caused the Load Plan to fail. Fix the source of the session failure.

You can restart the Load Plan instance. See Restarting a Load Plan Runfor more information.

Note that it will restart depending on the Restart Type defined on its steps. See the Handling Load Plan Exceptions and Restartability section in *Developing Integration Projects with Oracle Data Integrator* for more information.

You can also change the execution status of a failed Load Plan step from **Error** to **Done** on the Steps tab of the Load Plan run Editor to ignore this particular Load Plan step the next time the Load Pan run is restarted. This might be useful, for example, when the error causing this Load Plan step to fail is not possible to fix at the moment and you want to execute the rest of the Load Plan regardless of this Load Plan step.

Reviewing Successful Load Plans

When your Load Plan ends successfully, you can review the execution statistics from the Load Plan run editor.

You can also review the statistics for each session started for this Load Plan in the Session Editor.

To review the Load Plan run execution statistics:

1. In the Operator Navigator, identify the Load Plan run to review.
2. Double click the Load Plan run. The corresponding editor opens.
3. The record and execution statistics are displayed on the Steps tab.

Managing your Executions

Managing your development executions takes place in Operator Navigator. You can manage your executions during the execution process itself or once the execution has finished depending on the action that you wish to perform. The actions that you can perform are:

- Managing Sessions
- Managing Load Plan Executions
- Managing the Log
- Managing Scenarios and Load Plans
- Managing Schedules

Managing Sessions

Managing sessions involves the following tasks

- New sessions can be created by executing run-time objects or scenarios. See Running Integration Processesfor more information on starting sessions.
- Sessions in progress can be aborted. How to stop sessions is covered in Stopping a Session.
- Sessions failed, or stopped by user action can be restarted. Restarting sessions is covered in Restarting a Session.

In addition to these tasks, it may be necessary in production to deal with stale sessions.

Cleaning Stale Sessions

Stale sessions are sessions that are incorrectly left in a running state after an agent or repository crash.

The agent that started a session automatically detects when this session becomes stale and changes it to *Error* status. You can manually request specific agents to clean stale sessions in Operator Navigator or Topology Navigator.

To clean stale sessions manually:

1. Do one of the following:
   - From the Operator Navigator toolbar menu, select **Clean Stale Sessions**.
   - In Topology Navigator, from the Physical Architecture navigation tree, select an agent, right-click and select **Clean Stale Sessions**.

   The Clean Stale Sessions Dialog opens.
2. In the Clean Stale Sessions Dialog specify the criteria for cleaning stale sessions:
   - From the list, select the agents that will clean their stale sessions.

     Select **Clean all Agents** if you want all agents to clean their stale sessions.
   - From the list, select the Work Repositories you want to clean.

     Select **Clean all Work Repositories** if you want to clean stale sessions in all Work Repositories.
3. Click **OK** to start the cleaning process. A progress bar indicates the progress of the cleaning process.

Note that certain privileges are required for you to clean stale sessions. See Sessionsfor information.

Removing Temporary Objects

To remove temporary objects that could remain between executions:

1. Select **Remove temporary objects** from the Operator Navigator toolbar menu.
2. In the Remove Temporary Objects dialog set the criteria listed in Table 14-30.

Table 14-30 Remove Temporary Objects Dialog Parameters

Parameter	Description
Session Count	Number of sessions for which to skip cleanup. If the count is zero, all sessions that match the filter criteria are cleaned up.
From	Start date for the cleanup. All sessions started after this date are cleaned up.
To	End date for the cleanup. All sessions started before this date are cleaned up.
Context	Cleans up only those sessions executed in this context.
Agent	Cleans up only those sessions executed by this agent.
User	Cleans up only those sessions launched by this user.
Session Name	Name of the session.

3. Click **OK**.

Managing Load Plan Executions

Managing Load Plan Executions involves the following tasks:

- New Load Plan Instances and Runs can be created by executing Load Plans. See Executing a Load Planfor more information on starting Load Plans.
- Load Plan Runs in progress can be aborted. How to stop Load Plan runs is covered in Stopping a Load Plan Run.
- Load Plan Runs failed, or stopped by user action can be restarted. Restarting Load Plan Runs is covered in Restarting a Load Plan Run.

Managing the Log

Oracle Data Integrator provides several solutions for managing your log data:

- Filtering Sessions to display only certain execution sessions in Operator Navigator
- Purging the Log to remove the information of past sessions
- Organizing the Log with Session Folders
- Exporting and Importing Log Data for archiving purposes
- Runtime Logging for ODI components (ODI Studio, ODI Java EE agent, ODI Standalone agent, and ODI Standalone Colocated agent)

Filtering Sessions

Filtering log sessions allows you to display only certain sessions in Operator Navigator, by filtering on parameters such as the user, status or duration of sessions. Sessions that do not meet the current filter are hidden from view, but they are not removed from the log.

To filter out sessions:

1. In the Operator Navigator toolbar menu, click **Filter**. The Define Filter editor opens.

2. In the Define Filter Editor, set the filter criteria according to your needs. Note that the default settings select all sessions.
   - **Session Number**: Use blank to show all sessions.
   - **Session Name**: Use `%` as a wildcard. For example `DWH%` matches any session whose name begins with `DWH`.
   - Session's execution **Context**
   - **Agent** used to execute the session
   - **User** who launched the session
   - **Status**: Running, Waiting etc.
   - **Date** of execution: Specify either a date **From** or a date **To**, or both.

     While filtering to view running sessions in any version of ODI Operator, specify only the (From) field and leave the (To) field blank.

     If the (From) and (To) fields are specified, sessions that started after the (From) date and finished before the (To) date will be identified . Since running sessions do not have a known finish time, the (To) field should be left null. This will allow for running sessions to be identified.
   - **Duration greater than** a specified number of seconds
3. Click **Apply** for a preview of the current filter.
4. Click **OK**.

Sessions that do not match these criteria are hidden in the Session List navigation tree. The Filter button on the toolbar is activated.

To deactivate the filter click **Filter** in the Operator toolbar menu. The current filter is deactivated, and all sessions appear in the list.

Purging the Log

Purging the log allows you to remove past sessions and Load Plan runs from the log. This procedure is used to keeping a reasonable volume of sessions and Load Plans archived in the work repository. It is advised to perform a purge regularly. This purge can be automated using the OdiPurgeLog tool (see: the OdiPurgeLog section in *Oracle Data Integrator Tools Reference*) in a scenario.

To purge the log:

1. From the Operator Navigator toolbar menu select **Connect Navigator** > **Purge Log...** The Purge Log editor opens.
2. In the Purge Log editor, set the criteria listed in Table 14-31 for the sessions or Load Plan runs you want to delete.

Table 14-31 Purge Log Parameters

Parameter	Description
Purge Type	Select the objects to purge.

Table 14-31 (Cont.) Purge Log Parameters

Parameter	Description
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.

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 14-32 lists examples of how session folder keywords are matched.

Table 14-32 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 14-33.

Table 14-33 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.

Table 14-33 (Cont.) Log Export Parameters

Properties	Description
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 14-34 shows the mapping between the Java log levels and the Oracle Java Debugging Levels.

Table 14-34 Mapping between Java Log Levels and Oracle Java Debugging Levels

Java Log Levels	Oracle Java Debugging Levels
SEVERE intValue()+100	INCIDENT_ERROR:1
SEVERE	ERROR:1
WARNING	WARNING:1
INFO	NOTIFICATION:1
CONFIG	NOTIFICATION:16
FINE	TRACE:1
FINER	TRACE:16
FINEST	TRACE:32

Setting Up Runtime Logging

To set up runtime logging you have to enable different ODI loggers and log handlers. For ODI Java EE agents and ODI Standalone Colocated agents, this can be done through the logging configuration mechanism within Oracle Enterprise Manager Console. For ODI Studio and Standalone agents, set the related log level in the ODI logging system configuration file, as shown below:

1. Open the ODI logging system configuration file of the ODI component.

   Each component has its own configuration file:

   - ODI Studio:

     ```
     $ODI_HOME/odi/studio/bin/ODI-logging-config.xml
     ```

   - ODI Standalone agent:

```
<DOMAIN_HOME>/config/fmwconfig/components/ODI/<INSTANCE_NAME>/ODI-logging-
config.xml
```

2. Make sure that the path to your log files is a valid and existing path. For example:

   ```
   <log_handler name="ODI-file-handler"
   class="oracle.core.ojdl.logging.ODLHandlerFactory"
   level="ALL">
     <property name="format" value="ODL-Text"/>
     <property name="path" value="/u01/oracle/odi11g/oracledi/agent/log/$
   {LOG_FILE}"/>
     <property name="maxFileSize" value="1000000"/> <!-- in bytes -->
     <property name="maxLogSize" value="50000000"/> <!-- in bytes -->
     <property name="encoding" value="UTF-8"/>
   </log_handler>
   ```

   Note the following concerning the log files path:

   - If you are on Windows, the path could be for example:

     `%ODI_HOME%\oracledi\agent\log\${LOG_FILE}`

   - You can use a relative path on Windows and Unix.

3. Enable the logger and set the log level. For example, for logger *oracle.odi.agent* you can enable the most verbose logging setting:

   ```
   <logger name="oracle.odi.agent" level="TRACE:32" useParentHandlers="false">
     <handler name="ODI-file-handler"/>
     <handler name="ODI-console-handler"/>
   </logger>
   ```

4. Save the configuration file and restart ODI Studio and the ODI Standalone agent for the changes to take effect.

Example INFO (NOTIFICATION:1) Message

The `INFO (NOTIFICATION:1)` Message is logged if:

- The logger level (possibly inherited) is <= `NOTIFICATION:1`
- The log_handler level is <= `NOTIFICATION:1` (for example: `TRACE:1`)

The `INFO (NOTIFICATION:1)` Message is *not* logged if:

- The logger level (possibly inherited) is `> NOTIFICATION:1`
- The log_handler level is `> NOTIFICATION:1` (for example: `WARNING:1`)

The Runtime logger is called *oracle.odi.agent*. Its message levels cover the following content:

```
NOTIFICATION:1 (or INFO) Agent Level
NOTIFICATION:16 (or CONFIG) the above + Session Level
TRACE:1 (or FINE) the above + Step Level
TRACE:16 (or FINER) the above + Task + SQL
TRACE:32 (or FINEST) the above + more detail
```

It is recommended that the console level for *oracle.odi.agent* is set so that agent startup messages are displayed (`NOTIFICATION:1`).

Managing Scenarios and Load Plans

You can also manage your executions in Operator Navigator by using scenarios or Load Plans.

Before running a scenario, you need to generate it in Designer Navigator or import from a file. See the Using Scenarios chapter in *Developing Integration Projects with Oracle Data Integrator*. Load Plans are also created using Designer Navigator, but can also be modified using Operator Navigator. See the Using Load Plans chapter in *Developing Integration Projects with Oracle Data Integrator* for more information.

Launching a scenario from Operator Navigator is covered in Executing a Scenario from ODI Studio, and how to run a Load Plan is described in Executing a Load Plan.

Load Plan and Scenario Folders

In Operator Navigator, scenarios and Load Plans can be grouped into Load Plan and Scenario folders to facilitate organization. Load Plan and Scenario folders can contain other Load Plan and Scenario folders.

To create a Load Plan and Scenario folder:

1. In Operator Navigator go to the Load Plans and Scenarios navigation tree.
2. From the Load Plans and Scenarios toolbar menu, select **New Load Plan and Scenario Folder**.
3. On the Definition tab of the Load Plan and Scenario Folder editor enter a name for your folder.
4. From the File menu, select **Save**.

You can now reorganize your scenarios and Load Plans. Drag and drop them into the Load Plan and Scenario folder.

Importing Load Plans, Scenarios, and Solutions in Production

A Load Plan or a scenario generated from Designer can be exported and then imported into a development or execution repository. This operation is used to deploy Load Plans and scenarios in a different repository, possibly in a different environment or site.

Importing a Load Plan or scenario in a development repository is performed via Designer or Operator Navigator. With a execution repository, only Operator Navigator is available for this purpose.

See the Importing Scenarios in Production section in *Developing Integration Projects with Oracle Data Integrator* for more information on how to import a scenario in production, and the Importing Load Plans section in *Developing Integration Projects with Oracle Data Integrator* for more information on the Load Plan import.

Similarly, a solution containing several scenarios can be imported to easily transfer and restore a group of scenarios at once. See the Using Version Control chapter in *Developing Integration Projects with Oracle Data Integrator* for more information. Note that when connected to an execution repository, only scenarios may be restored from solutions.

Managing Schedules

A schedule is always attached to one scenario or one Load Plan. Schedules can be created in Operator Navigator. See Scheduling Scenarios and Load Plansfor more information.

You can also import an already existing schedule along with a scenario or Load Plan import. See the Importing Scenarios in Production and Exporting, Importing and Versioning Load Plans sections in *Developing Integration Projects with Oracle Data Integrator* for more information.

You can view the scheduled tasks of all your agents or you can view the scheduled tasks of one particular agent. See Displaying the Schedulefor more information.

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.

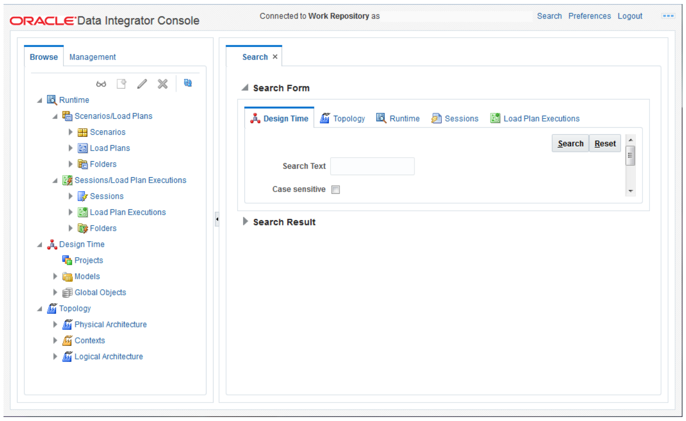
This web interface integrates seamlessly with Oracle Fusion Middleware Control Console and allows Fusion Middleware administrators to drill down into the details of Oracle Data Integrator components and sessions.

> **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 14-8 shows the layout of Oracle Data Integrator Console.

Figure 14-8 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 (ODI) Console connects to a repository via a Repository Connection, defined by an administrator.

You can access ODI Console from the user menu in Data Integration Platform Cloud.

Connecting to ODI Console

To open ODI Console:

1. From the Data Integration Platform Cloud user menu, select **Open ODI**.
2. Select the repository you want to connect to from the **Repository** menu, and then click **Proceed**.

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 14-35.

   Table 14-35 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 a Topologyfor more information on Java EE agents, datasources and templates.

To add a datasource to a physical agent:

1. Select the **Browse** Navigation tab.
2. Navigate to **Topology > Agents > Physical Agents**.
3. Select the agent you want to manage.
4. Click **Edit** in the Navigation tab toolbar.
5. Click **Add Datasource**
6. Provide a **JNDI Name** for this datasource and select the **Data Server Name**. This datasource will be used to connect to this data server from the machine into which the Java EE agent will be deployed.
7. Click **OK**.
8. Click **Save** to save the changes to the physical agent.

To create a template for a physical agent:

1. Select the **Browse** Navigation tab.
2. Navigate to **Topology > Agents > Physical Agents**.
3. Select the agent you want to manage.
4. Click **Edit** in the Navigation tab toolbar.
5. Click **Agent Deployment**.

6. Follow the steps of the **Agent Deployment** wizard. This wizard reproduces in a web interface the Server Template Generation wizard. See Deploying an Agent in a Java EE Application 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 a Topologyfor information about how to declare the agents in the master repositories.
- One or several Oracle Data Integrator Console applications. An Oracle Data Integrator Console application is used to browse master and work repositories.

The Master Repositories and Agent pages display both application metrics and information about the master and work repositories. You can also navigate to Oracle Data Integrator Console from these pages, for example to view the details of a session. In order to browse Oracle Data Integrator Console, the connections to the work and master repositories must be declared in Oracle Data Integrator Console. See *Installing and Configuring Oracle Data Integrator* for information on how to create and configure a domain.

Oracle Enterprise Manager Fusion Middleware Control

Oracle Enterprise Manager Fusion Middleware Control organizes a wide variety of performance data and administrative functions into distinct, Web-based home pages for the farm, cluster, domain, servers, components, and applications.

Oracle Data Integrator provides a plug-in that integrates with Oracle Enterprise Manager Fusion Middleware Control. Using this plug-in, Oracle Enterprise Manager Fusion Middleware Control can be used in conjunction with ODI Console to obtain information about your ODI agents, repositories, sessions, and load plan executions.

This section includes the following topics:

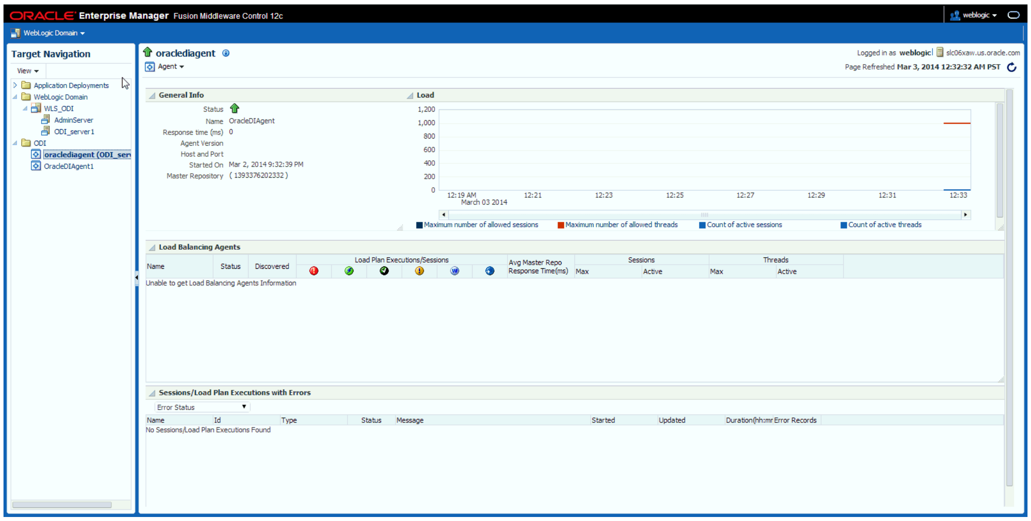
- Configuring Oracle Fusion Middleware Control with ODI Plug-in
- Searching Sessions
- Searching Load Plan Executions

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 14-9 shows the agents in the Deployment pane.

Figure 14-9 ODI Console Within Oracle Enterprise Manager Fusion Middleware Control


ORACLE Enterprise Manager Fusion Middleware Control 12c
weblogic
WebLogic Domain
Target Navigation
View
Application Deployments
WebLogic Domain
WLS_ODI
AdminServer
ODI_server1
ODI
oraclediagent (ODI_serv
OracleDIAgent1
oraclediagent
Agent
Logged in as weblogic | slc06xaw.us.oracle.com
Page Refreshed Mar 3, 2014 12:32:32 AM PST
Load
1,200
1,000
800
600
400
200
0
12:19 AM
March 03 2014
12:21
12:23
12:25
12:27
12:29
12:31
12:33
■ Maximum number of allowed sessions
🟥 Maximum number of allowed threads
■ Count of active sessions
■ Count of active threads

General Info

Status	⬆
Name	OracleDIAgent
Response time (ms)	0
Agent Version	
Host and Port	
Started On	Mar 2, 2014 9:32:39 PM
Master Repository	(1393376202332)

Load Balancing Agents

Name	Status	Discovered	Load Plan Executions/Sessions						Avg Master Repo Response Time(ms)	Sessions		Threads	
			●	●	●	●	●	●		Max	Active	Max	Active
Unable to get Load Balancing Agents Information													

Sessions/Load Plan Executions with Errors

Error Status ▾

Name	Id	Type	Status	Message	Started	Updated	Duration(hh:mr	Error Records
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*.

15 Validating Data Quality

This topic only applies to Data Integration Platform Cloud Classic.

You can access EDQ from the DIPC console user menu. To use EDQ, you must have the Administrator role.

EDQ provides a comprehensive data quality management environment that is used to understand, improve, protect and govern data quality. EDQ facilitates best practice master data management, data integration, business intelligence, and data migration initiatives. EDQ provides integrated data quality in customer relationship management and other applications.

Following are the key features of EDQ:

- Integrated data profiling, auditing, cleansing and matching
- Browser-based client access
- Ability to handle all types of data (for example, customer, product, asset, financial, and operational)
- Connection to any Java Database Connectivity (JDBC) compliant data sources and targets
- Multi-user project support (role-based access, issue tracking, process annotation, and version control)
- Services Oriented Architecture (SOA) support for designing processes that may be exposed to external applications as a service
- Designed to process large data volumes
- A single repository to hold data along with gathered statistics and project tracking information, with shared access
- Intuitive graphical user interface designed to help you solve real world information quality issues quickly
- Easy, data-led creation and extension of validation and transformation rules
- Fully extensible architecture allowing the insertion of any required custom processing

Understanding the Software Components

EDQ is a Java Web Application that uses a Java Servlet Engine, a Java Web Start graphical user interface, and a Structured Query Language (SQL) relational database management system (RDBMS) system for data storage.

EDQ is a client-server architecture. It is comprised of several client applications that are Graphical User Interfaces (GUIs), a data repository, and a business layer. This section provides details on the architecture of these components, their data storage, data access, and I/O requirements.

What Are the Client Applications?

Provides a number of client applications that are used to configure and operate the product. Most are Java Web Start applications, and the remainder are simple web pages. The following table lists all the client applications, how they are started, and what each does:

Application Name	Starts In	Purpose
Director	Web Start	Design and test data quality processing
Server Console	Web Start	Operate and monitor jobs
Match Review	Web Start	Review match results and make manual match decisions
Dashboard	Browser	Monitor data quality key performance indicators and trends
Case Management	Web Start	Perform detailed investigations into data issues through configurable workflows
Case Management Administration	Web Start	Configure workflows and permissions for Case Management
Web Service Tester	Browser	Test EDQ Web Services
Configuration Analysis	Web Start	Report on configuration and perform differences between versions of configuration
Issue Manager	Web Start	Manage a list of DQ issues
Administration	Browser	Administer the EDQ server (users, groups, extensions, launchpad configuration)
Change Password	Browser	Change password
Configuration Analysis	Web Start	Analyze project configurations, and report on differences'.

The client applications can be accessed from the Launchpad on the server. When a client launches one of the Java Web Start applications, such as Director, the application is downloaded, installed, and run on the client machine. The application communicates with the server to instantiate changes and receive messages from the server, such as information about tasks that are running and changes made by other users.

Since it is an extensible system, it can be extended to add further user applications when installed to work for a particular use case. For example, Oracle Watchlist Screening extends to add a user application for screening data against watchlists.

Note:

Many of the client applications are available either separately (for dedicated use) or within another application. For example, the Configuration Analysis, Match Review and Issue Manager applications are also available in Director.

Where Is Data Stored?

The client computer only stores user preferences for the presentation of the client applications, while all other information is stored on the EDQ server.

Network Communications

The client applications communicate over either an Hypertext Transfer Protocol (HTTP) or a Secure Hypertext Transfer Protocol (HTTPS) connection, as determined by the application configuration on start-up. For simplicity, this connection is referred to as 'the HTTP connection' in the remainder of this document.

How is Data Stored in the EDQ Repository?

EDQ uses a repository that contains two database schemas: the Config schema and the Results schema.

> **Note:**
>
> Each EDQ server must have its own Config and Results schemas. If multiple servers are deployed in a High Availability architecture, then the configuration cannot be shared by pointing both servers to the same schemas.

What Is the Config Schema?

The Config schema stores configuration data for EDQ. It is generally used in the typical transactional manner common to many web applications: queries are run to access small numbers of records, which are then updated as required.

Normally, only a small amount of data is held in this schema. In simple implementations, it is likely to be in the order of several megabytes. In the case of an exceptionally large EDQ system, especially where Case Management is heavily used, the storage requirements could reach 10 GB.

Access to the data held in the Config schema is typical of configuration data in other relational database management system (RDBMS) applications. Most database access is in the form of read requests, with relatively few data update and insert requests.

What Is the Results Schema

The Results schema stores snapshot, staged, and results data. It is highly dynamic, with tables being created and dropped as required to store the data handled by processors running on the server. Temporary working tables are also created and dropped during process execution to store any working data that cannot be held in the available memory.

The amount of data held in the Results schema will vary significantly over time, and data capture and processing can involve gigabytes of data. Data may also be stored in the Results database on a temporary basis while a process or a job runs. In the case of a job, several versions of the data may be written to the database during processing.

The Results schema shows a very different data access profile to the Config schema, and is extremely atypical of a conventional web-based database application. Typically, tables in the Results schema are:

- Created on demand
- Populated with data using bulk JDBC application programming interfaces (APIs)
- Queried using full table scans to support process execution
- Indexed
- Queried using complex SQL statements in response to user interactions with the client applications
- Dropped when the process or snapshot they are associated with is run again

The dynamic nature of this schema means that it must be handled carefully. For example, it is often advisable to mount redo log files on a separate disk.

Where does EDQ Store Working Data on Disk?

EDQ uses two configuration directories, which are separate from the installation directory that contains the program files. These directories are:

- The *base* configuration directory: This directory contains default configuration data. This directory is named `oedq.home` in an Oracle WebLogic installation but can be named anything in an Apache Tomcat installation.
- The *local* configuration directory: This directory contains overrides to the base configuration, such as data for extension packs or overrides to default settings. EDQ looks in this directory first, for any overrides, and then looks in the base directory if it does not find the data it needs in the local directory. The local configuration directory is named `oedq.local.home` in an Oracle WebLogic installation but can be named anything in an Apache Tomcat installation.

Some of the files in the configuration directories are used when processing data from and to file-based data stores. Other files are used to store server configuration properties, such as which functional packs are enabled, how EDQ connects to its repository databases, and other critical information.

The names and locations of the home and local home directories are important to know in the event that you need to perform any manual updates to templates or other individual components.

These directories are created when you install EDQ.

What Is the Business Layer?

The business layer fulfills three main functions:

- Provides the API that the client applications use to interact with the rest of the system.
- Notifies the client applications of server events that may require client applications updates.
- Runs the processes that capture and process data.

The business layer stores configuration data in the Config schema, and working data and results in the Results schema.

When passing data to and from the client application, the business layer behaves in a manner common to most traditional Java Web Applications. The business layer makes small database transactions and sends small volumes of information to the front-end using the HTTP connection. This is somewhat unusual in that the application front-ends are mostly rich GUIs rather than browsers. Therefore the data sent to the client application consists mostly of serialized Java objects rather than the more traditional HTML.

However, when running processes and creating snapshots, the business layer behaves more like a traditional batch application. In its default configuration, it spawns multiple threads and database connections in order to handle potentially very large volumes of data, and uses all available CPU cores and database I/O capacity.

It is possible to configure EDQ to limit its use of available resources, but this has clear performance implications. For further information, see the EDQ Installation Guide and EDQ Admin Guide.

Understanding Key Concepts of Enterprise Data Quality

This chapters provides the key concepts of EDQ.

Understanding EDQ Terms

The most important terms used in EDQ are:

Project
A group of related processes working on a common set, or sets, of data using shared reference data.

Data Store
A connection to a store of data, whether the data is stored in a database or in one or more files. The data store may be used as the source of data for a process, or you may export the written Staged Data results of a process to a data store, or both.

Process
Specifies a set of actions to be performed on some specified data. It comprises a series of **processors**, each specifying how data is to be handled and the rules that should be applied to it. A process may produce:

- **Staged data**: data or metrics produced by processing the input data and choosing to write output data to the results database.
- **Results data**: metric information summarizing the results of the process. For example, a simple validation process may record the number of records that failed and the number of records that passed validation.

Processor
A logical element that performs some operation on the data. Processors can perform statistical analysis, audit checks, transformations, matching, or other operations. Processors are chained together to form processes.

Published Processors
Additional processors (in addition to those in the Processor Library) that have been installed from an Extension Pack (such as, the Customer Data pack) or published onto the server by EDQ users. They are included in the Project Browser so that they can be packaged like other objects.There are three types of Published processors: Template, Reference, and Locked Reference

Reference Data
Consists of lists and maps that can be used by a processor to perform checking, matching, transformations and so on. Reference data can be supplied as part of EDQ or by a third party, or can be defined by the user.

Staged Data
Consists of data snapshots and data written by processes and is stored within the Results schema.

Snapshot
A captured copy of external data stored within the EDQ repository.

Job
A configured and ordered set of tasks that may be instigated either by EDQ or externally. Examples of tasks include executions of file downloads, snapshots, processes, and exports.

Images
Customizing the icon associated to a processor.

Exports
There are two types of exports:

- A prepared export (Staged Data Export or Results Book Export) that uses a saved configuration.
- An ad-hoc export of the current results from the Results Browser to an Excel file.

Web Services
Used to deploy processes (as configured in Director) to provide easy integration with source systems for real time data auditing, cleansing, and matching (for example, for real time duplicate prevention).

Run Profiles
Optional templates that specify configuration settings that you can use to override the default job run settings.

Issues
Allows users to keep a record of their key findings when analyzing data, and also provides a way for work on a project to be allocated and tracked amongst several users.

Project Notes
Allows you to use Director as the definitive repository for all information associated with the project. Any information that needs to be made available to all users working on a project can be added as a note throughout the progress of a project.

What Is Data Capture?

The data capture process begins with retrieving the data to be captured from an external data source. Data can be captured from databases, text files, XML files and so on. For a comprehensive list of possible types of data source, refer to the Data Stores topic in the Concepts section of the Online Help.

Depending on the type of data source, data capture may involve:

- Running a single SQL query on the source system.
- Sequentially processing a delimited or fixed format file.
- Processing an XML file to produce a stream of data.

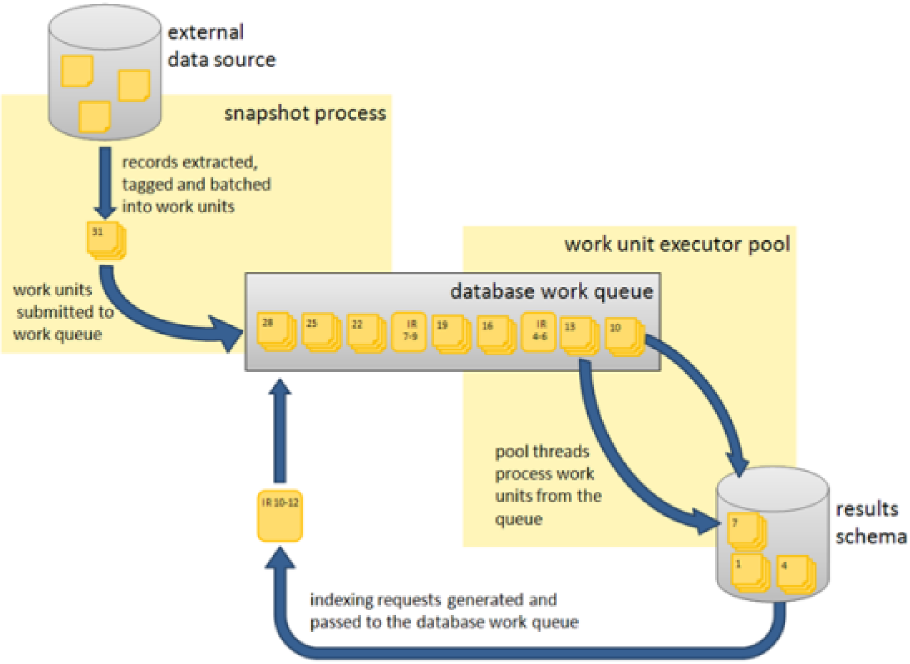
As the data is retrieved, it is processed by a single thread. This involves:

- Assigning an internal sequence number to each input record. This is usually a monotonically increasing number for each row.
- Batching the rows into work units. Once a work unit is filled, it is passed into the results database work queue.

The database work queue is made up of work requests — mostly data insertion or indexing requests — to be executed on the database. The queue is processed by a pool of threads that retrieve work units from the queue, obtain a database connection to the appropriate database, and execute the work. In the case of snapshotting, the work will consist of using the JDBC batch API to load groups of records into a table.

Once all the data has been inserted for a table, the snapshot process creates one or more indexing requests and adds them to the database work queue. At least one indexing request will be created per table to index the unique row identifier, but depending on the volume of data in the snapshot and the configuration of the snapshot process other columns in the captured data may also be used to generate indexes into the snapshot data.

Figure 15-1 The Data Capture Process



Understanding Network Communications and CPU Load

Snapshotting is expected to generate:

- I/O and CPU load on the machine hosting the data source while data is read
- CPU load on the web application server caused by the snapshot process reading data and grouping it for insertion
- I/O and CPU load on the web application server, caused by the database work unit executor threads
- A significant amount of I/O on the machine hosting the EDQ Results database as the data is inserted into a new table
- A significant amount of I/O and some CPU load on machine hosting the Results database as the data is indexed

For example, a default EDQ installation on a 4-core server taking a snapshot of 10,000 rows of data 10 columns wide would generate SQL of the following form:

```
DROP TABLE DN_1;
CREATE TABLE DN_1 (record-id, column1, column2, ..., column10);
```

100 bulk insert statements of the form:

```
INSERT INTO DN_1 (record_id, column1, column2, ..., column10) VALUES ( ?, ?, ..., ? )
```

each taking a group of 100 parameters. The bulk inserts would be executed in parallel over four separate database connections, one per CPU core.

```
ANALYZE TABLE DN_1 ESTIMATE STATISTICS SAMPLE 10 PERCENT
```

And finally, eleven `CREATE INDEX...` statements, indexing each of the columns in the new table (the original ten columns, plus the record_id). The `CREATE INDEX` statements would also be executed in parallel over four database connections.

What Is General Data Processing?

Once the data has been captured, it is ready for processing. The reader processor provides the downstream processors with managed access to the data, and the downstream processors produce results data. If any writer processors are present, they will write the results of processing back to the staged data repository.

Running a process causes the web application server to start a number of process **execution threads**. The default configuration of EDQ will start as many threads as there are cores on the EDQ application server machine.

What Is Streaming?

Instead of capturing data in a snapshot and storing it in the results database (other than temporarily during collation), it can be pulled from a source and pushed to targets as a stream.

What Is Streaming?

Each process execution thread is assigned a subset of the data to process. When the input data for a process is a data set of known size, such as snapshot or staged data, each thread will execute a query to retrieve a subset of the data, identified by the unique row IDs assigned during snapshotting. The queries would be of the form:

```
SELECT record_id, column1, column2, … , column10

FROM DN_1

WHERE record_id > 0 AND record_id <= 2500;
```

In the case where the process is not run against a data set of known size, such as a job scheduled to run directly against a data source, records are shared among the process execution threads by reading all records into a queue, which is then consumed by the process execution threads.

Each process execution thread is also made aware of the sequence of processors that comprise the process. The process execution threads pass the records through each of the appropriate processors. As the processors work, they accumulate results that need to be stored in the Results schema and, in the case of writer processors, they may also accumulate data that needs to be written to staged data. All this data is accumulated into insertion groups and added into database work units, which are processed as described in the 4.1 Data capture section.

Once an execution thread has processed all its assigned records, it waits for all other process execution threads to complete. The process execution threads then enter a **collation phase**, during which the summary data from the multiple copies of the process are accumulated and written to the Results database by the database work queue.

The following behavior is expected during batch processing:

- Read load on the Results schema as the captured data is read.

- CPU load on the web application server as the data is processed.
- Significant write load on the Results schema as results and staged data are written to the schema.
- Reduced CPU load as the collation phase is entered.
- A small amount of further database work as any outstanding database work units are processed and accumulated results written.
- Further write load on the Results schema at the end of the collation phase, in the form of requests to index the results and staged data tables, as necessary. The size and number of the index requests will vary, depending on data volumes and system configuration.

Processes that are heavily built around cleaning and validation operations will tend to be bound by the I/O capacity of the database. Some processors consume significant CPU resource, but generally the speed of operation is determined by how quickly data can be provided from and written to the Results schema.

What Is Whole Record Set Processing?

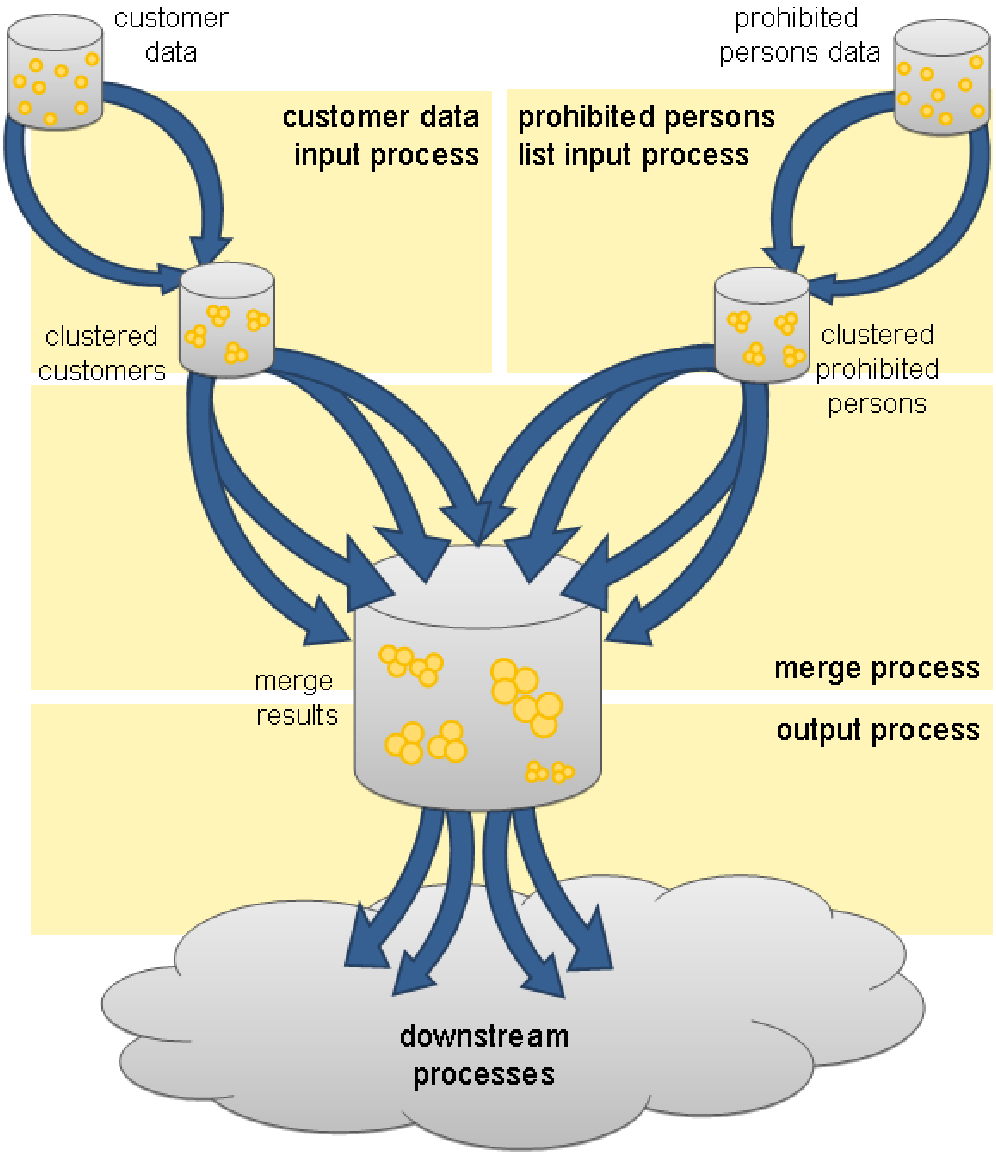
There are a number of processors, such as the Record Duplication Profiler and Duplicate Check processors, that require access to the whole record set in order to work. If these processors only had access to a subset of the data, they would be unable to detect duplicate records with any accuracy. These processes use multiple threads to absorb the input records and build them into a temporary table. Once all the records have been examined, they are re-emitted by distributing the records amongst the various process execution threads. There is no guarantee that a record will be emitted on the same process execution thread that absorbed it.

What Is Match Processing?

EDQ match processors are handled in a significantly different way from the simpler processors. Due to the nature of the work carried out by match processors, multiple passes through the data are required.

A match processor is executed by treating it as a series of sub-processes. For example, consider a process designed to match a customer data snapshot against a list of prohibited persons. The process contains a match processor that is configured to produce a list of customer reference numbers and related prohibited person identifiers. Each data stream that is input to, or output from, the match processor, is considered to be a sub-process of the match processor. Therefore, there are three sub-processes in this example, representing the customer data input stream, the prohibited persons input stream and the output data stream of the match processor. The match processor itself forms a fourth sub-process, which effectively couples the data inputs to its outputs. Each sub-process is assigned the normal quota of process execution threads, so on a 4-core machine, each sub-process would have four process execution threads.

Figure 15-2 Match Process Threads



When execution of the match processor begins, the input data sub-processes run first, processing the input data. At this point, there is no work available for the match or match output sub-processes, which remain dormant. The input data sub-processes generate cluster values for the data streams and store the cluster values and incoming records in the Results schema, using the normal database work units mechanism.

Once the input data sub-processes have processed all the available records, they terminate and commence collation of their sub-process results. Meanwhile, the match sub-process will become active. The match sub-process then works through a series of stages, with each process execution thread waiting for all the other process execution threads to complete each stage before they progress to the next. Each time a new stage begins, the work will be subdivided amongst the processor executor threads in a manner that is appropriate at that stage. The processing stages are:

Phase	Description
Comparison phase	The customer data and prohibited people data is retrieved, ordered by cluster values. The data is gathered into groups of equal cluster values, queued and passed to the match process threads to compare the records. Where relationships are found between records the relationship information is written to the Results schema.
Provisional grouping phase	The relationship information detected during the comparison phase is retrieved in chunks and provisional groups of related records are formed. The relationship chunks are processed in parallel by the match processor threads. These provisional groups are written back to the Results database.
Final grouping phase	The provisional group table is inspected by a single thread to check for groups that have been artificially split by chunk boundaries. If any such cross-chunk groups are found they are merged into a single group.
Merged output phase	Each of the match processor threads retrieves an independent subset of the match groups and forms the merged output, merging multiple records into the single output records.

This completes the match sub-process, and so the match processor execution threads now move into their collation phase.

At this point, the sub-process associated with the output of match data becomes active. The output data is divided amongst the process execution threads for the output sub-process and passed to the processors down stream from the match processor. From this point onwards, the data is processed in the normal batch processing way.

Benchmarks and production experience have shown that the comparison phase of a match processor is one of the few EDQ operations that is likely to become CPU bound. When anything other than very simple comparison operations are performed, the ability of the CPU to handle the comparison load limits the process. The comparison operations scale very well and are perfectly capable of utilizing all CPU cycles available to the EDQ Web Application Server.

> **Tip:**
>
> Oracle recommends that the reader familiarizes themselves with the material contained in the Online Help regarding matching concepts..

What Is Real-Time Processing?

EDQ is capable of processing messages in real time. Currently, EDQ supports messaging using:

- Web Services
- JMS-enabled messaging software

When configured for real-time message processing, the server starts multiple process execution threads to handle messages as they are received. An incoming message is handed to a free process execution thread, or placed in a queue to await the next process execution thread to become free. Once the message has been processed, any staged data will be written to the Results database, and the process execution thread will either pick up the next message from the queue, if one exists, or become available for the next incoming message.

When processing data in real time, the process may be run in **interval mode**. Interval mode allows the process to save results at set intervals so that they can be inspected by a user and published to the EDQ Dashboard. The interval can be determined either by the number of records processed or by time limit. When an interval limit is reached, EDQ starts a new set of process execution threads for the process. Once all the new process execution threads have completed any necessary initialization, any new incoming messages are passed to the new threads. Once the old set of process execution threads have finished processing any outstanding messages, the system directs those threads to enter the collation phase and save any results, after which the old process execution threads are terminated and the data is available for browsing.

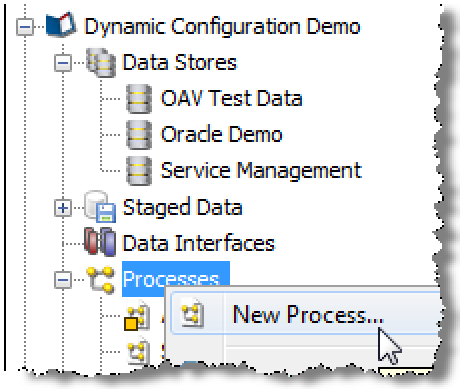
Using Enterprise Data Quality

This chapter tells us how to use EDQ.

Adding a Process

To add a Process to analyze data from a snapshot:

1. From the menu, select **File - New Process**, or
2. Right-click on **Processes** in the Project Browser, and select **New Process**:



3. Select the Staged Data, Data Interface, Reference Data or Real time Data Provider that you want to use in the process, or do not select anything if you want to configure the Reader in the process later.

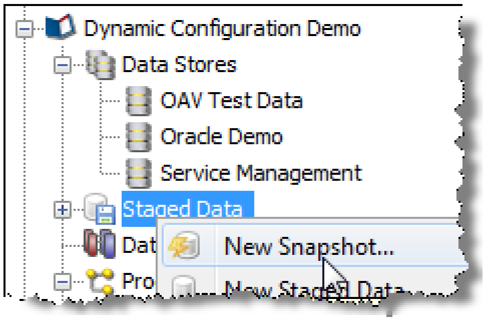
> **Note:**
>
> It may be that you do not want to stage the data you are analyzing; that is, you may want to stream the data directly from the source. This can be done by selecting the Staged Data configuration, and changing the Process Execution Preferences of the process.

4. Select whether or not to add Profiling processors to the process straight away. This may be useful if you are analyzing the data for the first time.
5. Give the process a **Name** and an optional **Description**.
6. Click **Finish**.

Adding a Snapshot

To add a Snapshot of data from a connected data store:

1. Right-click on Staged Data in the Project Browser, and select New Snapshot:



2. Select the data store that you want to create the snapshot from, or add a new data store if the desired data store is not on the list.
3. Select the table or view to snapshot (or you may specify SQL to snapshot a new view of the data).
4. Select the columns from the table or view that you want to include in the snapshot, and how to enable sorting and filtering on the snapshot.

   By default, intelligent sort and filter enablement is used. This means that results based on the snapshot may be sorted or filtered using any column(s), provided the snapshot is under a certain size (set by the system administrator). If the snapshot is above that size, results based on it cannot be sorted or filtered by any column, though users will be prompted to enable sorting and filtering on specific columns if they attempt to do it using the Results Browser.

   Alternatively, you can switch off intelligent sort and filter enablement, and manually select the columns that you enable for sorting and filtering.

   The default threshold above which sorting and filtering will be disabled for snapshots when using intelligent sort and filter enablement is 10 million cells - so for example a snapshot with 500,000 rows and 15 columns (7,500,000 cells) would have sorting and filtering enabled, but a snapshot with 500,000 rows and 25 columns (12,500,000 cells) would have sorting and filtering disabled.

   > **Note:**
   >
   > It is advisable to select all columns. The columns to work with in a given process can be a subset of these.

5. Optionally filter the table or view to snapshot a subset of it (or you may write your own SQL WHERE clause).

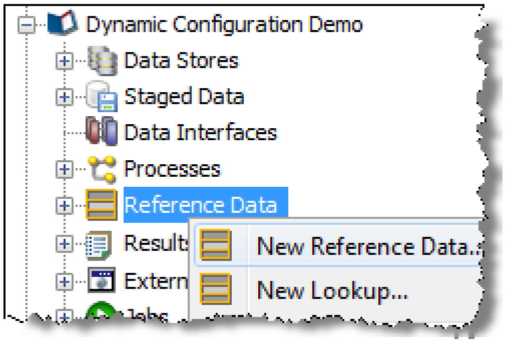
6. Optionally sample the selected data (for example, the first n records, the first n records after an offset, or 1 record in every 100).
7. Optionally perform no data normalization. For more information, see the "No Data Handling" topic in Oracle Enterprise Data Online Help.
8. Give the snapshot a **Name**, and choose whether or not to run it immediately.
9. Click **Finish** to confirm the addition of the snapshot.

The snapshot is created and visible in the project browser. It is now ready to be run (by Right-click, **Run Snapshot**), or used in a process and run later. The snapshot may also be 'streamed'; that is, used as a way of selecting the records to be processed from a Data Store directly; that is, without copying them into the repository.

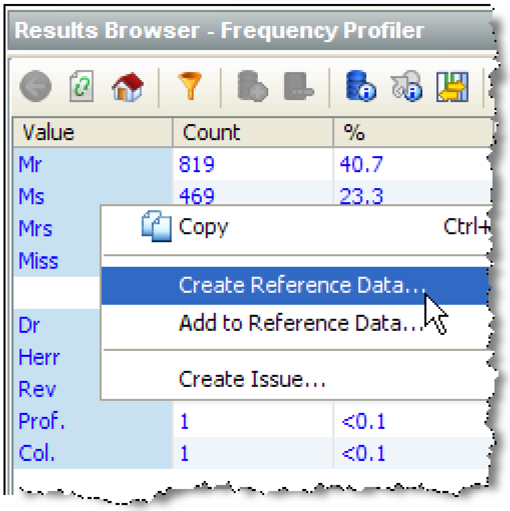
Adding Reference Data

To add a set of Reference Data to use in processors for the validation or transformation of data:

1. Right-click on Reference Data in the Project Browser, and select **New Reference Data**:
2. If you want the Reference Data to be a lookup onto Staged or External Data, or an alternative lookup onto an existing Reference data set, rather than a new set of Data, choose **New Lookup...**.



Or, create the Reference Data using the data in the Results Browser by selecting some data values, right-clicking and selecting **Create Reference Data**. For example, from the results of a Frequency profiler, select a results tab. Select the desired values, right click and **Create Reference Data**:



Value	Count	%
Mr	819	40.7
Ms	469	23.3
Mrs		
Miss		
Dr		
Herr		
Rev		
Prof.	1	<0.1
Col.	1	<0.1

Note:

You can use the normal windows Shift-select, and Control-select options to select data. Take care not to drill down when attempting to select the desired values. Alternatively, to select all the loaded data in the results browser for a given column, Control-select the column at the top (for example, the Value column in the screenshot above). Press Escape to de-select all selected data.

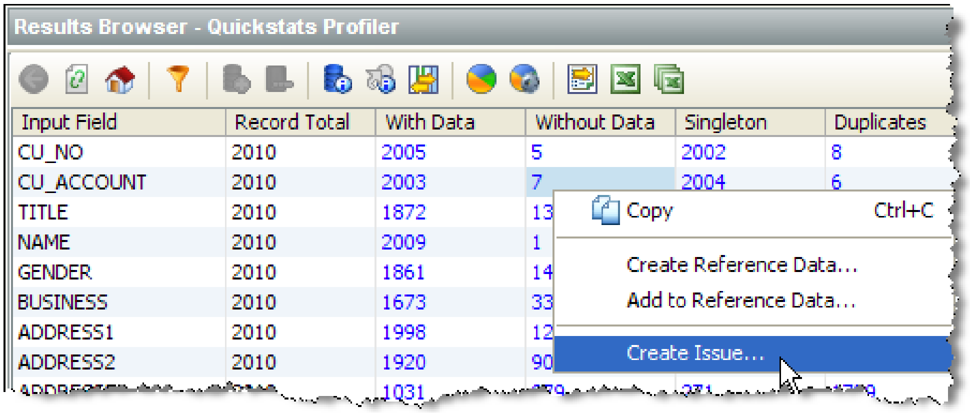
3. If you are adding new Reference Data (rather than a new Lookup onto existing Reference Data), define the columns that you require in the Reference Data. For example, for a simple list of values, define a single column. If you would like the Reference Data Editor to add a uniqueness constraint so that duplicate entries cannot be created, select the **Unique?** option on the column.
4. Select the column or columns that you want to use when performing lookups on the data.
5. Select the column or columns that you want to use when returning values from a lookup.
6. Optionally, select the Category of Reference Data that you want to create, if you are creating Reference Data of a specific type (such as a list of regular expressions).
7. Give the Reference Data a **Name** (for example, Valid Titles) and optional **Description** (for example, 'Created from Customers table') and choose whether or not to edit the data now.
8. If you choose to edit the data now, add or delete any entries in the Reference Data using the Reference Data Editor.
9. Click **OK** to finish.

The Reference Data set now appears under your project in the Project Browser, and is ready for use in processors - for example in a List Check.

Adding an Issue

To add an Issue based on your results (for example to tag an item of interest, or to create an action for another user to follow-up on):

1. Right-click on the data in the Results Browser, and select **Create Issue...**:



Input Field	Record Total	With Data	Without Data	Singleton	Duplicates
CU_NO	2010	2005	5	2002	8
CU_ACCOUNT	2010	2003	7	2004	6
TITLE	2010	1872	13		
NAME	2010	2009	1		
GENDER	2010	1861	14		
BUSINESS	2010	1673	33		
ADDRESS1	2010	1998	12		
ADDRESS2	2010	1920	90		

> **Note:**
>
> The issue will be linked to the specific process and processor where it was created, so that another user can quickly find the related data.

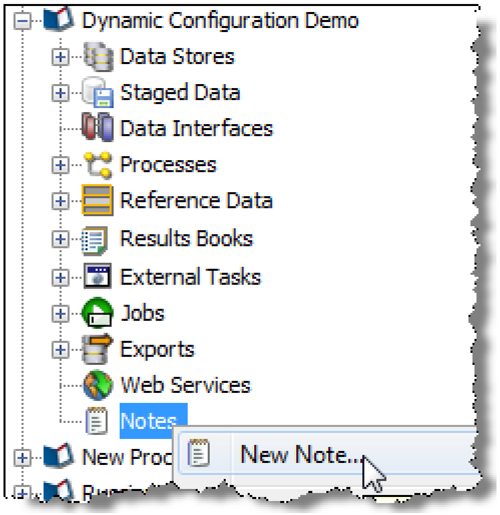
2. Add a **Description** of the issue.
3. Optionally assign the issue to yourself or another user, and specify the **Action** needed (if any).
4. Click **Save** to save the issue.

The issue is added, and available from the Issue Manager. If the issue was assigned to another user, that user will be notified of the outstanding issue immediately, if he/she is logged on.

Adding a Project Note

To add a Note to a project, for example to attach a project plan, or to share some key information amongst project users:

1. Right-click **Notes** in the Project Browser, and select **New Note:**



2. Give the note a **Title**.
3. Add detail to the note (or leave blank if all you need is a Title, and one or more attachments).
4. Browse your file system to add a file attachment to the note (or drag and drop files onto the indicated area).
5. Click **Save**.

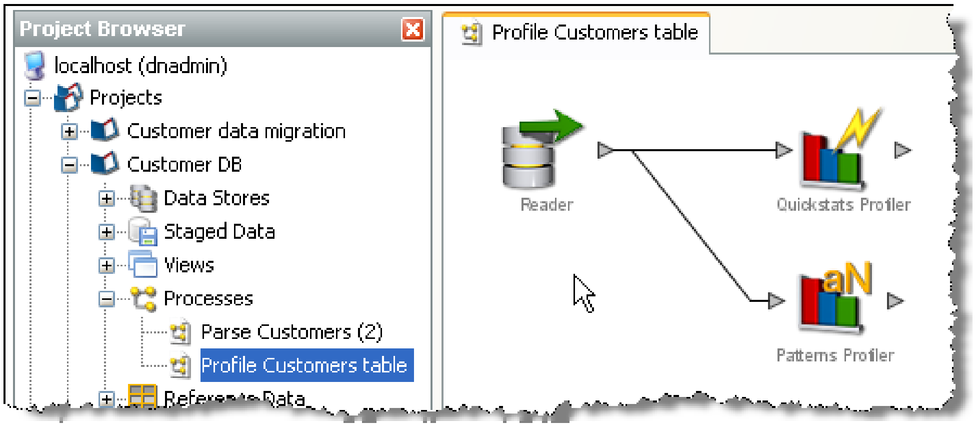
The note is created and visible in the Project Browser.

Adding Processors

EDQ comes with a library of processors for processing your data.

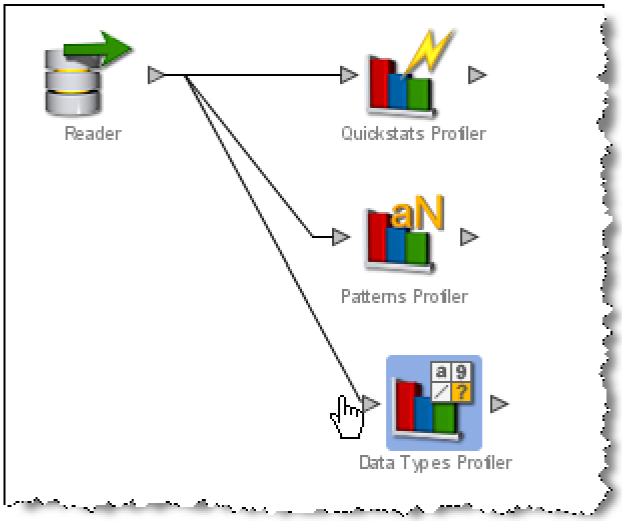
To add a processor to your process:

1. Ensure your process is open on the Canvas:

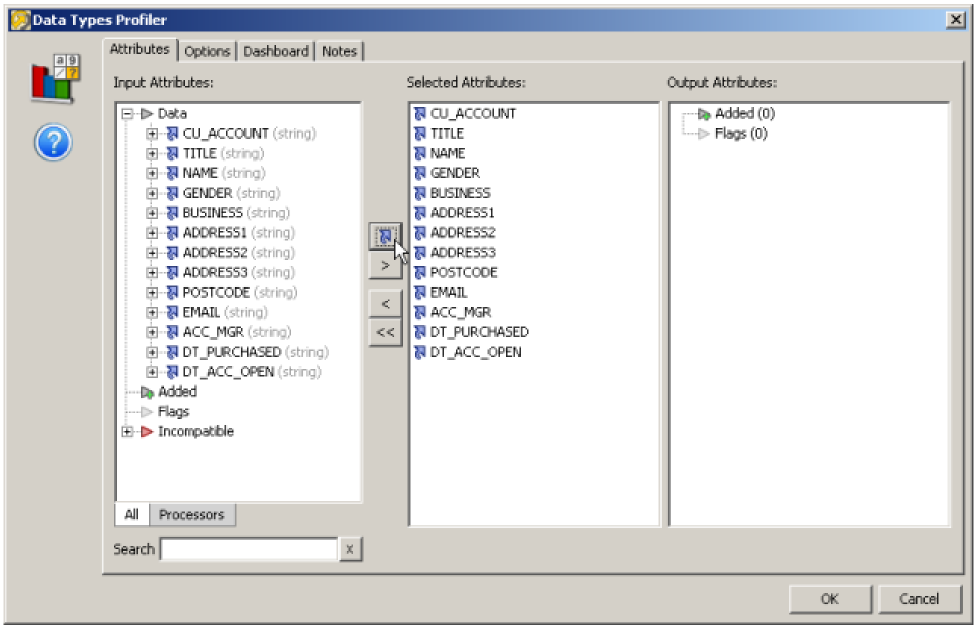


2. Double-click on the Reader to configure it to read data from Staged Data (such as a Snapshot), a View, or a real time data provider.
3. Select the data that you want to read, and the attributes from the data that are relevant to your process.

4. Add a processor from the Tool Palette to the process by clicking on a processor and dragging it to the Canvas.
5. Connect the processor to the data from the Reader by joining the arrows:



6. Configure the processor by selecting its input attributes:



Data Types Profiler

Attributes | Options | Dashboard | Notes

Input Attributes:

- Data
  - CU_ACCOUNT (string)
  - TITLE (string)
  - NAME (string)
  - GENDER (string)
  - BUSINESS (string)
  - ADDRESS1 (string)
  - ADDRESS2 (string)
  - ADDRESS3 (string)
  - POSTCODE (string)
  - EMAIL (string)
  - ACC_MGR (string)
  - DT_PURCHASED (string)
  - DT_ACC_OPEN (string)
- Added
- Flags
- Incompatible

All | Processors

Search

Selected Attributes:

- CU_ACCOUNT
- TITLE
- NAME
- GENDER
- BUSINESS
- ADDRESS1
- ADDRESS2
- ADDRESS3
- POSTCODE
- EMAIL
- ACC_MGR
- DT_PURCHASED
- DT_ACC_OPEN

Output Attributes:

- Added (0)
- Flags (0)

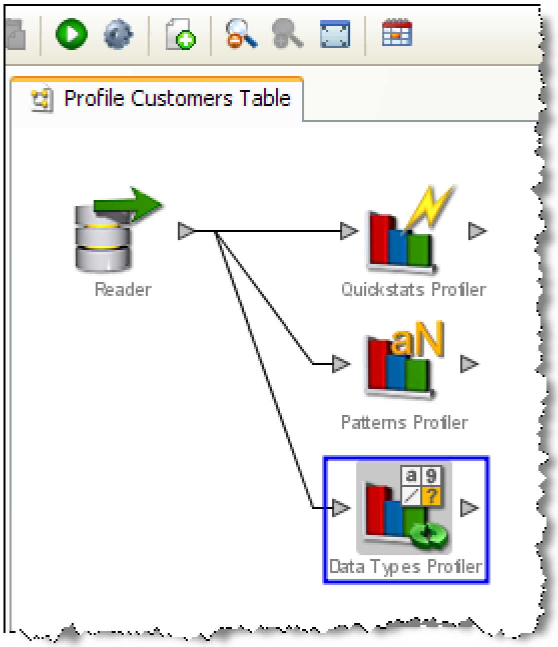
The blue arrow icons indicate that the latest version of the attribute will be used as the input. This is especially important when transformation processors have been used.

See the "About Transformation Processors" topic in the Enterprise Data Quality Online Help for further information.

Note:

- For Profiling processors, it is common to analyze the data in all attributes to discover issues of interest about the data.
- Once a processor is correctly configured, it no longer appears with a blue background.

7. Once you have connected the set of processors that you want to use, click on the Quick Run process button on the Toolbar to run the process and look at the results:

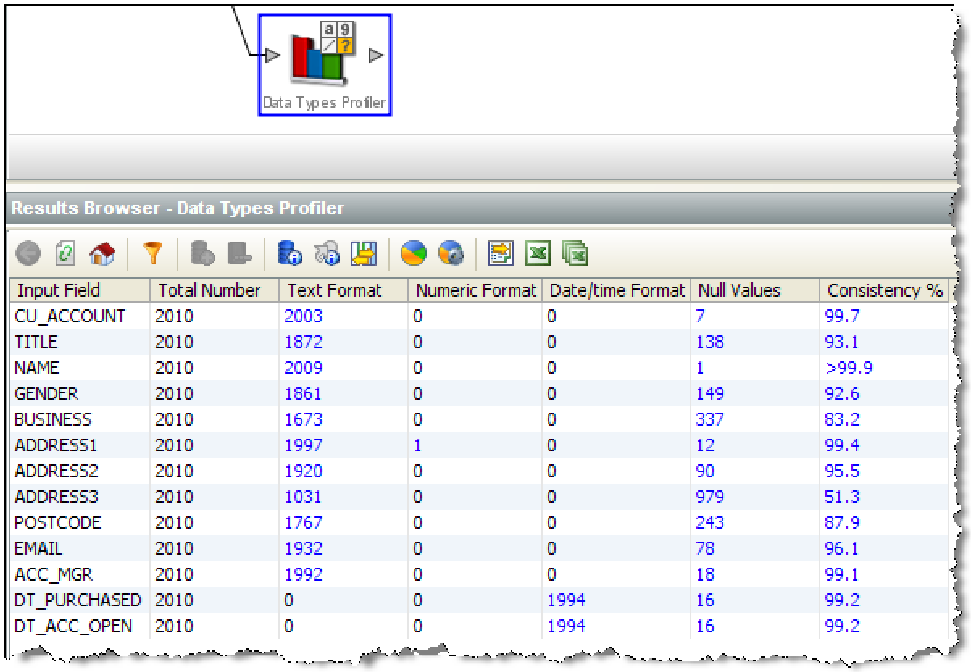


8. The Canvas background changes to blue to show you that the process is running. (Also, the process icon in the Project Browser turns green so that other users connected to the same host can see that it is running.).

Note:

The process is locked and cannot be edited while it is running.

9. When the process has finished, the processors no longer appear with a shaded background, and you can browse on the results for each processor by clicking on the processor on the Canvas, and viewing its results in the Results Browser:



Input Field	Total Number	Text Format	Numeric Format	Date/time Format	Null Values	Consistency %
CU_ACCOUNT	2010	2003	0	0	7	99.7
TITLE	2010	1872	0	0	138	93.1
NAME	2010	2009	0	0	1	>99.9
GENDER	2010	1861	0	0	149	92.6
BUSINESS	2010	1673	0	0	337	83.2
ADDRESS1	2010	1997	1	0	12	99.4
ADDRESS2	2010	1920	0	0	90	95.5
ADDRESS3	2010	1031	0	0	979	51.3
POSTCODE	2010	1767	0	0	243	87.9
EMAIL	2010	1932	0	0	78	96.1
ACC_MGR	2010	1992	0	0	18	99.1
DT_PURCHASED	2010	0	0	1994	16	99.2
DT_ACC_OPEN	2010	0	0	1994	16	99.2

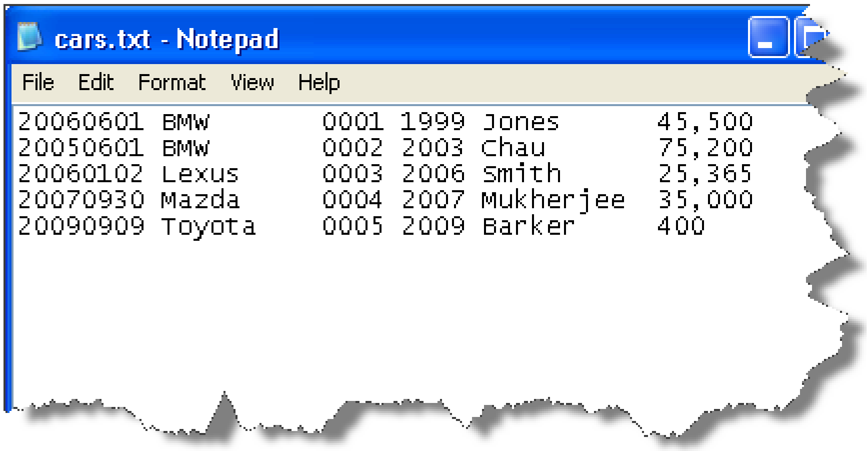
10. Drill-down on the metrics to see the relevant data for the metric.

Having successfully created a process, you can now begin to create Reference Data for the validation and transformation of your data.

Configuring Fixed Width Text File Formats

When you define a new data store that connects to a fixed width text file, the New Data Store wizard will prompt you to define the names and sizes of the data fields in the file.

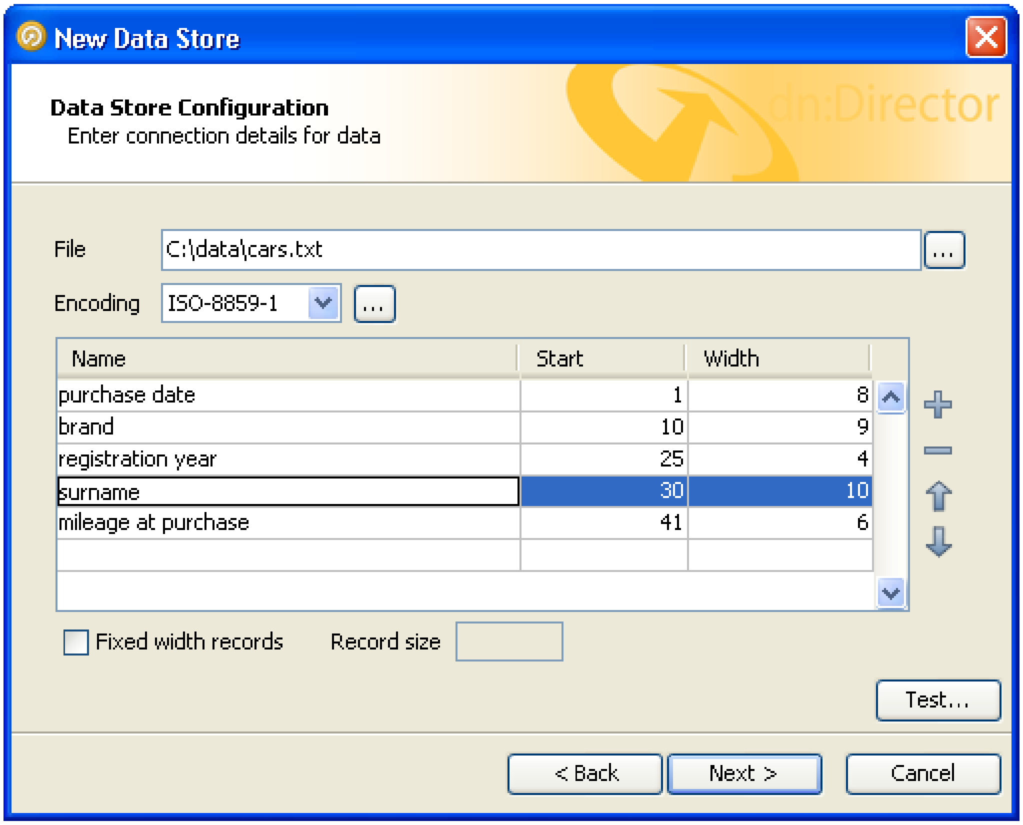
Data in a fixed-width text file is arranged in rows and columns, with one entry per row. Each column has a fixed width, specified in characters, which determines the maximum amount of data it can contain. No delimiters are used to separate the fields in the file. Instead, smaller quantities of data are padded with spaces to fill the allotted space, such that the start of a given column can always be specified as an offset from the beginning of a line. The following file snippet illustrates characteristics common to many flat files. It contains information about cars and their owners, but there are no headings to the columns in the file and no information about the meaning of the data. In addition, the data has been laid out with a single space between each column, for readability:



```
20060601 BMW       0001 1999 Jones      45,500
20050601 BMW       0002 2003 Chau       75,200
20060102 Lexus     0003 2006 Smith      25,365
20070930 Mazda     0004 2007 Mukherjee  35,000
20090909 Toyota    0005 2009 Barker     400
```

In order to parse the data in a fixed width text file correctly, EDQ needs to be informed of the column sizes implicit in that file. This is done in the New Data Store wizard, and can be edited as part of the data store settings later, if required.

When you first enter the data store configuration screen for a fixed width text file, the columns table is empty. In the following screenshot, it has been populated with the mapping information for some of the columns in our sample file:



File: C:\data\cars.txt

Encoding: ISO-8859-1

Name	Start	Width
purchase date	1	8
brand	10	9
registration year	25	4
surname	30	10
mileage at purchase	41	6

□ Fixed width records   Record size

Each column is described to EDQ by its starting position and width, in characters. Each column is also assigned a name, which is used in data snapshots and downstream processing so that the data can be identified. Names are defined by the user at the time the data store is defined and should be descriptive, for maximum downstream usability.

Notice that the positions of the data columns are defined in terms of start point and width. Note also that the first character on a line is at position 1, not zero. Providing a width and a starting point for each column means that EDQ does not assume that one column continues right up until the start of the next, with the result that:

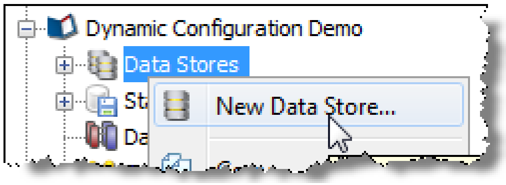
- Any spaces that have been included in the file for readability, such as a single space between columns, can automatically be bypassed.
- It is not necessary to define mappings for every column in the file. If un-needed columns exist, they can simply be omitted from the column definitions in the data store configuration. For example, we have not included the third column from the file in our mappings, but because the boundaries of the surrounding columns are tightly defined, no extraneous data will be included in the data set.
- Columns do not have to be specified in the same order as they occur in the file. The column order specified here will be reflected in any snapshots created from the data source.

The buttons to the right of the columns table can be used to add or remove records, or move the selected record up or down in the list.

Connecting to a Data Store

To connect to a new Data Store in order to process a set of data:

1. Right-click on Data Stores within your project in the Project Browser, and select **New Data Store**:



2. Select the category of data store that you want to connect to - Database, Text file, XML file, MS Office file, or Other (if you want to specify connection details using JDBC or ODBC).
3. Select where the data will be accessed from - the server or the client.

   (See the "Client-side Data Stores" topic in the Enterprise Data Quality Online Help).
4. Select the type of data store that you want to connect to (for example, for databases, select the type of database, for example, Oracle, SQL Server, MS Access etc.).
5. Specify the connection details to the data. For example:.
   - For a **client-side Access** database, browse to the .mdb file on the local file system.
   - For a **client-side Text file**, browse to the directory that contains the text file on the local file system. For **fixed-width text files**, you must also define the fields that are present in the file.

- For a **server-side file (Access, Text, Excel or XML)**, enter the name of the file as it exists (or will exist) in the server landing area, including the file suffix. It is possible to use a project-specific landing area to enforce isolation between data in different projects. Administrators will need to setup a landing area for the projects which require the use of this functionality. Again, for **fixed-width text** files, you must also define the fields that are present in the file.

6. For a **Database**, specify the **Database host**, **Port number** (if not using the default port number), **Database Name**, **User Name**, **Password**, and **Schema** (if different from the default Schema for the User).
7. For a database accessed via a JNDI connection, specify the JNDI name.
8. For any other type of data that can be accessed via an ODBC bridge connector, specify the **ODBC DSN**, **JDBC URL**, **User name** and **Password**.
9. For any **Other** type of data that can be accessed via JDBC, specify the **Driver Class Name**, **JDBC URL**, **User name** and **Password**.
10. If you want to check the connection to the new data store, use the **Test** button. Note that it is possible to specify connection details to a file that is not yet present (such as a file to be created by an export task in EDQ).

> **Note:**
>
> Connecting non-native types of data source requires some knowledge of JDBC connectivity.

11. Give the data store a **Name**, and click **Finish**.

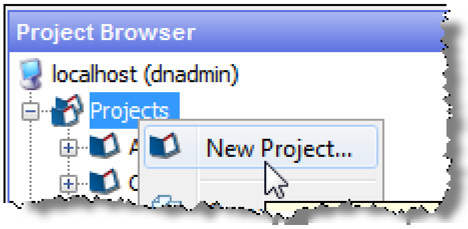
The new data stories are now configured and visible in the Project Browser.

Alternatively, if the data store is going to be shared across projects, you can create it at the System level (outside of any specific project) in the same way as above.

Adding a Project

To create a Project for working on a specific set or sets of data:

1. From the menu, select **File - New Project**, or
2. Right-click on Projects in the Project Browser, and select **New Project**:



3. Follow the steps in the wizard, giving the project a **Name** and an optional **Description**.
4. Assign the user permissions required for the new Project. By default, all user groups will have access to the Project.

The new project is created and visible in the Project Browser.

You may want to Add a Note to the project to share amongst project users.

Exporting Data (Prepared exports)

There are two sources of data for prepared exports: Data from Staged Data and data from Results Books.

> **Note:**
>
> It is also possible to create an ad-hoc export to Excel directly from the Results Browser.

Running a Prepared Export

There are two ways of running a prepared export of a Staged Data table or of a Results Book:

1. Manually run the export
2. Run the export as part of a job

Understanding the Key Tasks In EDQ

This chapter includes the following sections:

About Snapshots

When a Snapshot is configured to run as part of a job, there is a single **Enabled?** option, which is set by default.

Disabling the option allows you to retain a job definition but to disable the refresh of the snapshot temporarily - for example because the snapshot has already been run and you want to re-run later tasks in the job only.

About Processes

There are a variety of different options available when running a process, either as part of a job, or using the Quick Run option and the Process Execution Preferences.

About Readers

For each Reader in a process, the following option is available:

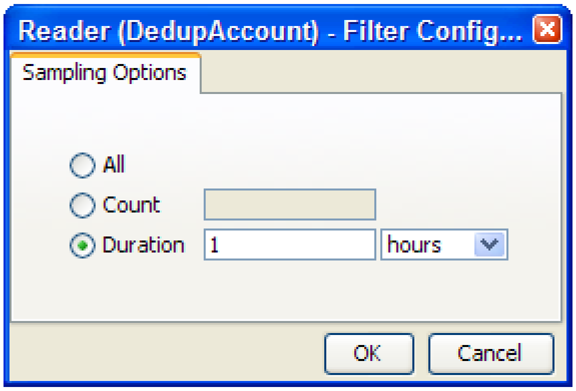
Sample?

The Sample option allows you to specify job-specific sampling options. For example, you might have a process that normally runs on millions of records, but you might want to set up a specific job where it will only process some specific records that you want to check, such as for testing purposes.

Specify the required sampling using the option under **Sampling**, and enable it using the **Sample** option.

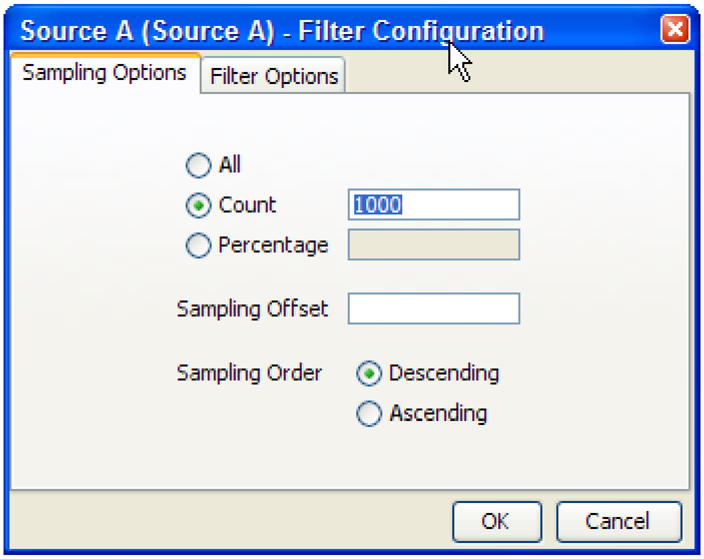
The sampling options available will depend on how the Reader is connected.

For Readers that are connected to real time providers, you can limit the process so that it will finish after a specified number of records using the **Count** option, or you can run the process for a limited period of time using the **Duration** option. For example, to run a real time monitoring process for a period of 1 hour only:



○ All
○ Count
● Duration 1 hours

For Readers that are connected to staged data configurations, you can limit the process so that it runs only on a sample of the defined record set, using the same sampling and filtering options that are available when configuring a Snapshot. For example, to run a process so that it only processes the first 1000 records from a data source:



○ All
● Count 1000
○ Percentage
Sampling Offset
Sampling Order ● Descending ○ Ascending

The Sampling Options fields are as follows:

- **All** - Sample all records.

- **Count** - Sample n records. This will either be the first n records or last n records, depending on the Sampling Order selected.
- **Percentage** - Sample n% of the total number of records.
- **Sampling Offset** - The number of records after which the sampling should be performed.
- **Sampling Order** - Descending (from first record) or Ascending (from last).

> **Note:**
>
> If a Sampling Offset of, for example, 1800 is specified for a record set of 2000, only 200 records can be sampled regardless of the values specified in the Count or Percentage fields.

About Process

The following options are available when running a process, either as part of the Process Execution Preferences, or when running the process as part of a job.

- **Use Intelligent Execution?**

  Intelligent Execution means that any processors in the process which have up-to-date results based on the current configuration of the process will not re-generate their results. Processors that do not have up-to-date results are marked with the rerun marker. Intelligent Execution is selected by default. Note that if you choose to sample or filter records in the Reader in a process, all processors will re-execute regardless of the Intelligent Execution setting, as the process will be running on a different set of records.
- **Enable Sort/Filter in Match processors?**

  This option means that the specified Sort/Filter enablement settings on any match processors in the process (accessed via the Advanced Options on each match processor) will be performed as part of the process execution. The option is selected by default. When matching large volumes of data, running the Sort/Filter enablement task to allow match results to be reviewed may take a long time, so you may want to defer it by de-selecting this option. For example, if you are exporting matching results externally, you may want to begin exporting the data as soon as the matching process is finished, rather than waiting until the Enable Sort/Filter process has run. You may even want to over-ride the setting altogether if you know that the results of the matching process will not need to be reviewed.
- **Results Drill Down**

  This option allows you to choose the level of Results Drill Down that you require.

  - **All** means that drilldowns will be available for all records that are read in to the process. This is only recommended when you are processing small volumes of data (up to a few thousand records), when you want to ensure that you can find and check the processing of any of the records read into the process.
  - **Sample** is the default option. This is recommended for most normal runs of a process. With this option selected, a sample of records will be made available for every drilldown generated by the process. This ensures that you can explore results as you will always see some records when drilling down, but ensures that excessive amounts of data are not written out by the process.

  - **None** means that the process will still produce metrics, but drilldowns to the data will be unavailable. This is recommended if you want the process to run as quickly as possible from source to target, for example, when running data cleansing processes that have already been designed and tested.

- **Publish to Dashboard?**

  This option sets whether or not to publish results to the Dashboard. Note that in order to publish results, you first have to enable dashboard publication on one or more audit processors in the process.

About Run Modes

To support the required Execution Types, EDQ provides three different run modes.

If a process has no readers that are connected to real time providers, it always runs in the Normal mode as mentioned below.

If a process has at least one reader that is connected to a real time provider, the mode of execution for a process can be selected from one of the following three options:

Normal mode

In Normal mode, a process runs to completion on a batch of records. The batch of records is defined by the Reader configuration, and any further sampling options that have been set in the process execution preferences or job options.

Prepare mode

Prepare mode is required when a process needs to provide a real time response, but can only do so where the non real time parts of the process have already run; that is, the process has been prepared.

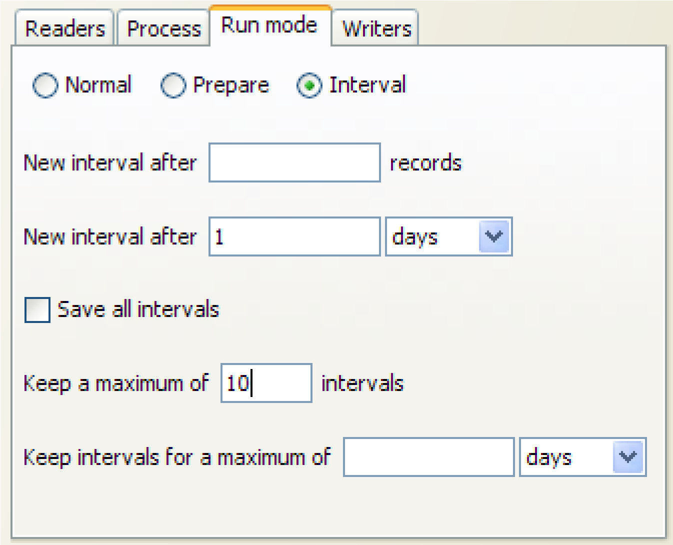
Prepare mode is most commonly used in real time reference matching. In this case, the same process will be scheduled to run in different modes in different jobs - the first job will prepare the process for real time response execution by running all the non real time parts of the process, such as creating all the cluster keys on the reference data to be matched against. The second job will run the process as a real time response process (probably in Interval mode).

Interval mode

In Interval mode, a process may run for a long period of time, (or even continuously), but will write results from processing in a number of intervals. An interval is completed, and a new one started, when either a record or time threshold is reached. If both a record and a time threshold are specified, then a new interval will be started when either of the thresholds is reached.

As Interval mode processes may run for long periods of time, it is important to be able to configure how many intervals of results to keep. This can be defined either by the number of intervals, or by a period of time.

For example, the following options might be set for a real time response process that runs on a continuous basis, starting a new interval every day:



○ Normal ○ Prepare ● Interval

New interval after [ ] records

New interval after [1] days

□ Save all intervals

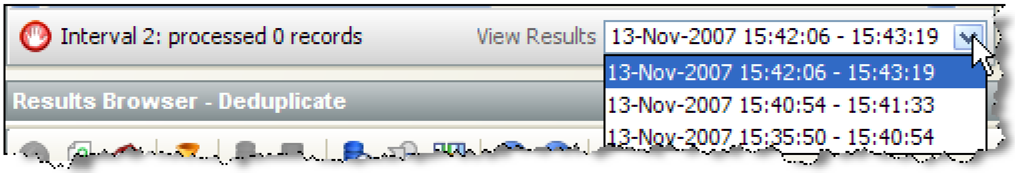
Keep a maximum of [10] intervals

Keep intervals for a maximum of [ ] days

Browsing Results from processing in Interval mode

When a process is running in Interval mode, you can browse the results of the completed intervals (as long as they are not too old according to the specified options for which intervals to keep).

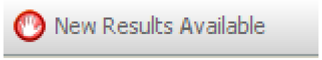
The Results Browser presents a simple drop-down selection box showing the start and end date and time of each interval. By default, the last completed interval is shown. Select the interval, and browse results:


Interval 2: processed 0 records

View Results 13-Nov-2007 15:42:06 - 15:43:19

13-Nov-2007 15:42:06 - 15:43:19
13-Nov-2007 15:40:54 - 15:41:33
13-Nov-2007 15:35:50 - 15:40:54

If you have the process open when a new set of results becomes available, you will be notified in the status bar:


New Results Available

You can then select these new results using the drop-down selection box.

About Writers

For each Writer in a process, the following options are available:

- **Write Data?**

  This option sets whether or not the writer will 'run'; that is, for writers that write to stage data, de-selecting the option will mean that no staged data will be written, and for writers that write to real time consumers, de-selecting the option will mean that no real time response will be written.

This is useful in two cases:

1. You want to stream data directly to an export target, rather than stage the written data in the repository, so the writer is used only to select the attributes to write. In this case, you should de-select the Write Data option and add your export task to the job definition after the process.
2. You want to disable the writer temporarily, for example, if you are switching a process from real time execution to batch execution for testing purposes, you might temporarily disable the writer that issues the real time response.

- **Enable Sort/Filter?**

  This option sets whether or not to enable sorting and filtering of the data written out by a Staged Data writer. Typically, the staged data written by a writer will only require sorting and filtering to be enabled if it is to be read in by another process where users might want to sort and filter the results, or if you want to be able to sort and filter the results of the writer itself.

  The option has no effect on writers that are connected to real time consumers.

About External Tasks

Any External Tasks (File Downloads, or External Executables) that are configured in a project can be added to a Job in the same project.

When an External Task is configured to run as part of a job, there is a single **Enabled?** option.

Enabling or Disabling the Enable export option allows you to retain a job definition but to enable or disable the export of data temporarily.

About Exports

When an Export is configured to run as part of a job, the export may be enabled or disabled (allowing you to retain a Job definition but to enable or disable the export of data temporarily), and you can specify how you want to write data to the target Data Store, from the following options:

Delete current data and insert (default)

EDQ deletes all the current data in the target table or file and inserts the in-scope data in the export. For example, if it is writing to an external database it will truncate the table and insert the data, or if it is writing to a file it will recreate the file.

Append to current data

EDQ does not delete any data from the target table or file, but adds the in-scope data in the export. When appending to a UTF-16 file, use the UTF-16LE or UTF-16-BE character set to prevent a byte order marker from being written at the start of the new data.

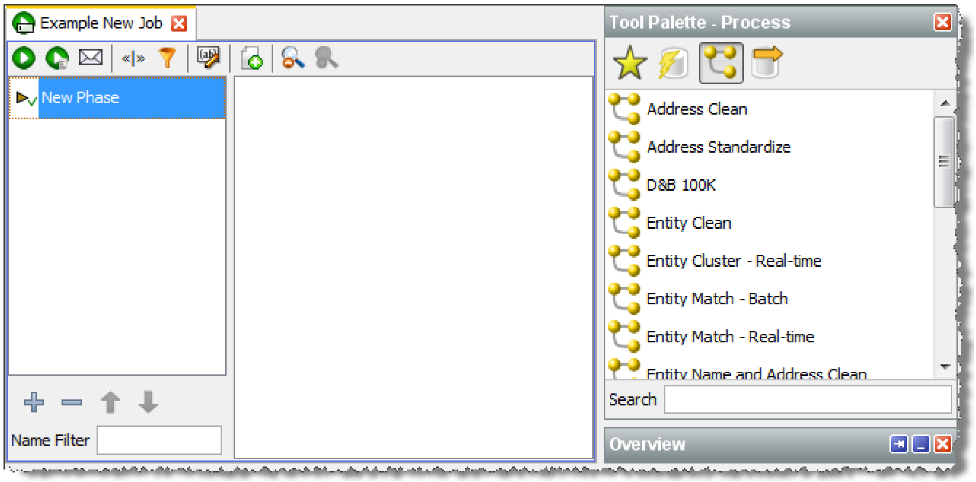
Replace records using primary key

EDQ deletes any records in the target table that also exist in the in-scope data for the export (determined by matching primary keys) and then inserts the in-scope data.

> **Note:**
>
> - When an Export is run as a standalone task in Director (by right-clicking on the Export and selecting **Run**), it always runs in **Delete current data and insert** mode.
> - **Delete current data and insert** and **Replace records using primary key** modes perform **Delete** then **Insert** operations, not **Update**. It is possible that referential integrity rules in the target database will prevent the deletion of the records, therefore causing the Export task to fail. Therefore, in order to perform an **Update** operation instead, Oracle recommends the use of a dedicated data integration product, such as Oracle Data Integrator.

Creating a Job

1. Expand the required project in the Project Browser.
2. Right-click the **Jobs** node of the project and select **New Job**. The **New Job** dialog is displayed.
3. Enter a Name and (if required) Description, then click **Finish**. The Job is created and displayed in the Job Canvas:



4. Right-click **New Phase** in the Phase list, and select **Configure**.
5. Enter a name for the phase and select other options as required:

Field	Type	Description
Enabled?	Checkbox	To enable or disable the Phase. Default state is checked (enabled). **Note:** The status of a Phase can be overridden by a Run Profile or with the 'runopsjob' command on the EDQ Command Line Interface.

Field	Type	Description
Execution Condition	Drop-down list	To make the execution of the Phase conditional on the success or failure of previous Phases. The options are: • Execute on failure: the phase will only execute if the previous phase did not complete successfully. • Execute on success (default): the Phase will only execute if all previous Phases have executed successfully. • Execute regardless: the Phase will execute regardless of whether previous Phases have succeeded or failed. **Note:** If an error occurs in any phase, the error will stop all 'Execute on success' phases unless an 'Execute regardless' or 'Execute on failure' phase runs with the 'Clear Error?' button checked runs first.
Clear Error?	Checkbox	To clear or leave unchanged an error state in the Job. If a job phase has been in error, an error flag is applied. Subsequent phases set to Execute on success will not run unless the error flag is cleared using this option. The default state is unchecked.
Triggers	N/A	To configure Triggers to be activated before or after the Phase has run.

6. Click **OK** to save the settings.
7. Click and drag Tasks from the Tool Palette, configuring and linking them as required.
8. To add more Phases, click the **Add Job Phase** button at the bottom of the Phase area. Phase order can be changed by selecting a Phase and moving it up and down the list using the **Move Phase** buttons. To delete a Phase, click the **Delete Phase** button.
9. When the Job is configured as required, click **File > Save**.

Editing a Job

1. To edit a Job, locate it within the Project Browser and either double click it or right-click and select **Edit...**.
2. The Job is displayed in the Job Canvas. Edit the Phases and/or Tasks as required.
3. Click **File >Save**.

About Exports

When an Export is configured to run as part of a job, the export may be enabled or disabled (allowing you to retain a Job definition but to enable or disable the export of data temporarily), and you can specify how you want to write data to the target Data Store, from the following options:

Delete current data and insert (default)

EDQ deletes all the current data in the target table or file and inserts the in-scope data in the export. For example, if it is writing to an external database it will truncate the table and insert the data, or if it is writing to a file it will recreate the file.

Append to current data

EDQ does not delete any data from the target table or file, but adds the in-scope data in the export. When appending to a UTF-16 file, use the UTF-16LE or UTF-16-BE character set to prevent a byte order marker from being written at the start of the new data.

Replace records using primary key

EDQ deletes any records in the target table that also exist in the in-scope data for the export (determined by matching primary keys) and then inserts the in-scope data.

Note:

- When an Export is run as a standalone task in Director (by right-clicking on the Export and selecting **Run**), it always runs in **Delete current data and insert** mode.
- **Delete current data and insert** and **Replace records using primary key** modes perform **Delete** then **Insert** operations, not **Update**. It is possible that referential integrity rules in the target database will prevent the deletion of the records, therefore causing the Export task to fail. Therefore, in order to perform an **Update** operation instead, Oracle recommends the use of a dedicated data integration product, such as Oracle Data Integrator.

Deleting a Job

Deleting a job does not delete the processes that the Job contained, and nor does it delete any of the results associated with it. However, if any of the processes contained in the Job were last run by the Job, the last set of results for that process will be deleted. This will result in the processors within that process being marked as out of date.

To delete a Job, either:

- select it in the Project Browser and press the Delete key; or
- right-click the job and select **Delete**.

Remember that it is not possible to delete a Job that is currently running.

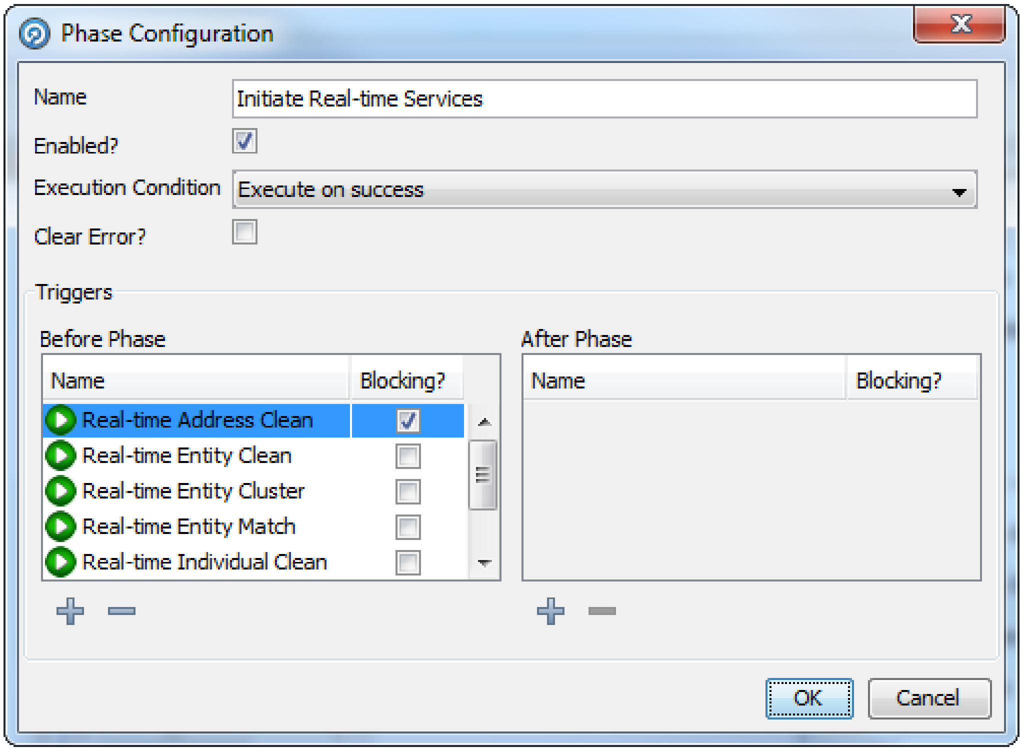
Using Job Triggers

Job Triggers are used to start or interrupt other Jobs. Two types of triggers are available by default:

- Run Job Triggers: used to start a Job.

- Shutdown Web Services Triggers: used to shut down real-time processes.

Further Triggers can be configured by an Administrator, such as sending a JMS message or calling a Web Service. They are configured using the Phase Configuration dialog, an example of which is provided below:



Name: Initiate Real-time Services

Enabled? ☑

Execution Condition: Execute on success

Clear Error? ☐

Triggers

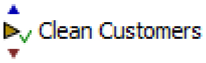
Before Phase

Name	Blocking?
Real-time Address Clean	☑
Real-time Entity Clean	☐
Real-time Entity Cluster	☐
Real-time Entity Match	☐
Real-time Individual Clean	☐

After Phase

Name	Blocking?

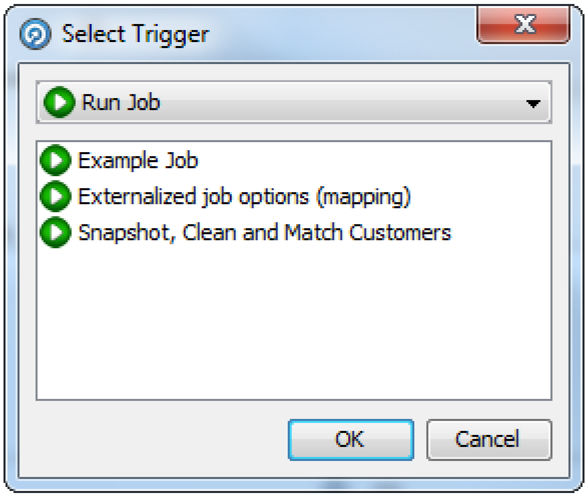
Triggers can be set before or after a Phase. A Before Trigger is indicated by a blue arrow above the Phase name, and an After Trigger is indicated by a red arrow below it. For example, the following image shows a Phase with Before and After Triggers:



Triggers can also be specified as Blocking Triggers. A Blocking Trigger prevents the subsequent Trigger or Phase beginning until the task it triggers is complete.

Configuring Triggers

1. Right-click the required Phase and select Configure. The Phase Configuration dialog is displayed.
2. In the Triggers area, click the **Add Trigger** button under the **Before Phase** or **After Phase** list, as required. The **Select Trigger** dialog is displayed:


Select Trigger
Run Job
Example Job
Externalized job options (mapping)
Snapshot, Clean and Match Customers
OK
Cancel

3. Select the Trigger type in the drop-down field.
4. Select the specific Trigger in the list area.
5. Click **OK**.
6. If required, select the **Blocking?** checkbox next to the Trigger.
7. Set further Triggers as required.
8. When all the Triggers have been set, click **OK**.

Deleting a Trigger from a Job

1. Right-click the required Phase and select **Configure**.
2. In the **Phase Configuration** dialog, find the Trigger selected for deletion and click it.
3. Click the **Delete Trigger** button under the list of the selected Trigger. The Trigger is deleted.
4. Click **OK** to save changes. However, if a Trigger is deleted in error, click **Cancel** instead.

Creating and Managing Processors

In addition to the range of data quality processors available in the Processor Library, EDQ allows you to create and share your own processors for specific data quality functions.

There are two ways to create processors:

- Using an external development environment to write a new processor - see the **Extending EDQ** topic in online help on oracle doc center for more details.
- Using EDQ to create processors - read on in this topic for more details

Creating a Processor From a Sequence of Configured Processors

EDQ allows you to create a single processor for a single function using a combination of a number of base (or 'member') processors used in sequence.

Note that the following processors may not be included in a new created processor:

- Parse
- Match
- Group and Merge
- Merge Data Sets

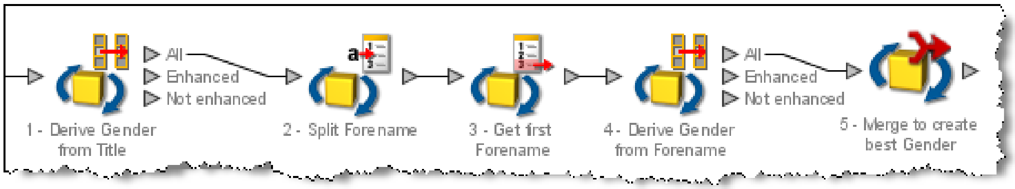
A single configured processor instance of the above processors may still be published, however, in order to reuse the configuration.

Processor creation example

To take a simple example, you may want to construct a reusable Add Gender processor that derives a Gender value for individuals based on Title and Forename attributes. To do this, you have to use a number of member processors. However, when other users use the processor, you only want them to configure a single processor, input Title and Forename attributes (however they are named in the data set), and select two Reference Data sets - one to map Title values to Gender values, and one to map Forename values to Gender values. Finally, you want three output attributes (TitleGender, NameGender and BestGender) from the processor.

To do this, you need to start by configuring the member processors you need (or you may have an existing process from which to create a processor). For example, the screenshot below shows the use of 5 processors to add a Gender attribute, as follows:

1. Derive Gender from Title (Enhance from Map).
2. Split Forename (Make Array from String).
3. Get first Forename (Select Array Element).
4. Derive Gender from Forename (Enhance from Map).
5. Merge to create best Gender (Merge Attributes).


All
Enhanced
Not enhanced
1 - Derive Gender from Title
2 - Split Forename
3 - Get first Forename
All
Enhanced
Not enhanced
4 - Derive Gender from Forename
5 - Merge to create best Gender

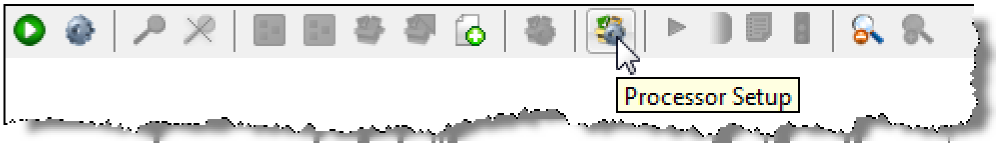
To make these into a processor, select them all on the Canvas, right-click, and select **Make Processor**.

This immediately creates a single processor on the Canvas and takes you into a processor design view, where you can set up how the single processor will behave.

Setting Inputs

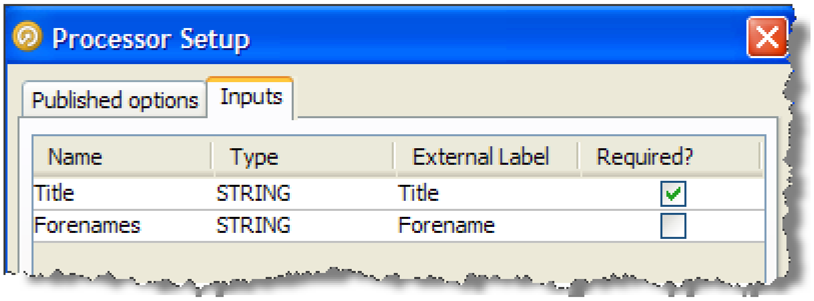
The inputs required by the processor are calculated automatically from the configuration of the base processors. Note that where many of the base processors use the same configured input attribute(s), only one input attribute will be created for the new processor.

However, if required you can change or rename the inputs required by the processor in the processor design view, or make an input optional. To do this, click on the **Processor Setup** icon at the top of the Canvas, then select the Inputs tab.



In the case above, two input attributes are created - Title and Forenames, as these were the names of the distinct attributes used in the configuration of the base processors.

The user chooses to change the External Label of one of these attributes from Forenames to Forename to make the label more generic, and chooses to make the Forename input optional:



Name	Type	External Label	Required?
Title	STRING	Title	☑
Forenames	STRING	Forename	□

Note that if an input attribute is optional, and the user of the processor does not map an attribute to it, the attribute value will be treated as Null in the logic of the processor.

Note:

It is also possible to change the Name of each of the input attributes in this screen, which means their names will be changed within the design of the processor only (without breaking the processor if the actual input attributes from the source data set in current use are different). This is available so that the configuration of the member processors matches up with the configuration of the new processor, but will make no difference to the behavior of the created processor.

Setting Options

The processor design page allows you to choose the options on each of the member processors that you want to expose (or "publish") for the processor you are creating. In our example, above, we want the user to be able to select their own Reference Data sets for mapping Title and Forename values to Gender values (as for example the processor may be used on data for a new country, meaning the provided Forename to Gender map would not be suitable).

To publish an option, open the member processor in the processor design page, select the Options tab, and tick the Show publishing options box at the bottom of the window.

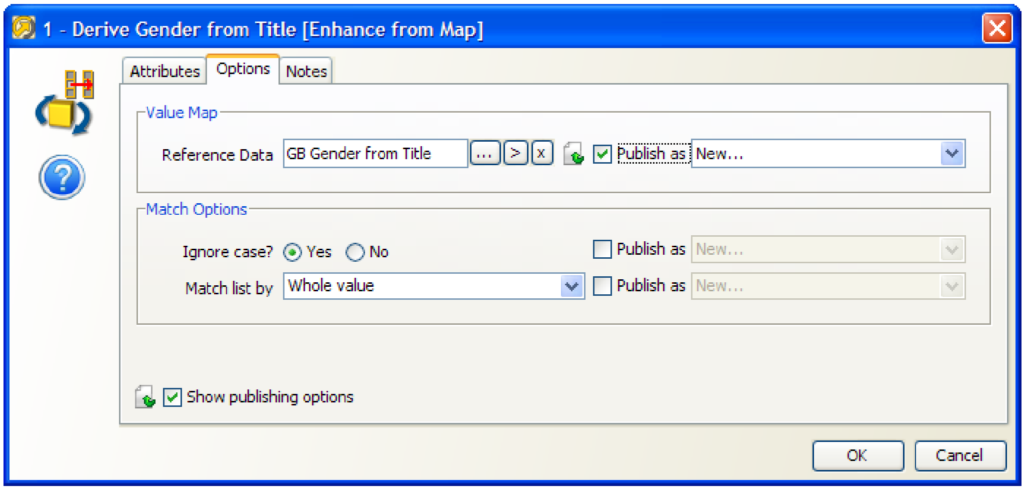
You can then choose which options to publish. If you do not publish an option, it will be set to its configured value and the user of the new processor will not be able to change it (unless the user has permission to edit the processor definition).

There are two ways to publish options:

- Publish as New - this exposes the option as a new option on the processor you are creating.
- Use an existing published option (if any) - this allows a single published option to be shared by many member processors. For example, the user of the processor can specify a single option to Ignore Case which will apply to several member processors.

> **Note:**
>
> If you do not publish an option that uses Reference Data, the Reference Data will be internally packaged as part of the configuration of the new processor. This is useful where you do not want end users of the processor to change the Reference Data set.

In our example, we open up the first member processor (Derive Gender from Title) and choose to publish (as new) the option specifying the Reference Data set used for mapping Title values to Gender values:


1 - Derive Gender from Title [Enhance from Map]

Attributes | Options | Notes

Value Map

Reference Data: GB Gender From Title ☑ Publish as New...

Match Options

Ignore case? ● Yes ○ No □ Publish as New...

Match list by: Whole value □ Publish as New...

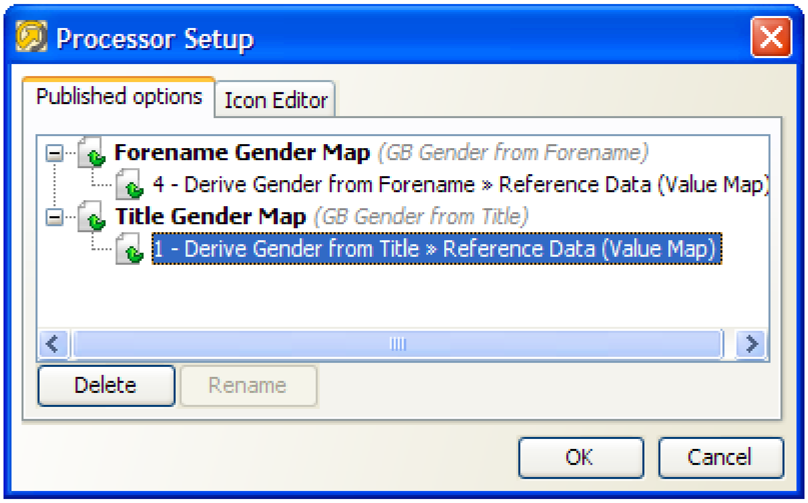
☑ Show publishing options

Note above that the Match Options are not published as exposed options, meaning the user of the processor will not be able to change these.

We then follow the same process to publish the option specifying the Reference Data set used for mapping Forename values to Gender values on the fourth processor (Derive Gender from Forename).

Once we have selected the options that we want to publish, we can choose how these will be labeled on the new processor.

To do this, click on Processor Setup button at the top of the canvas and rename the options. For example, we might label the two options published above **Title Gender Map** and **Forename Gender Map**:

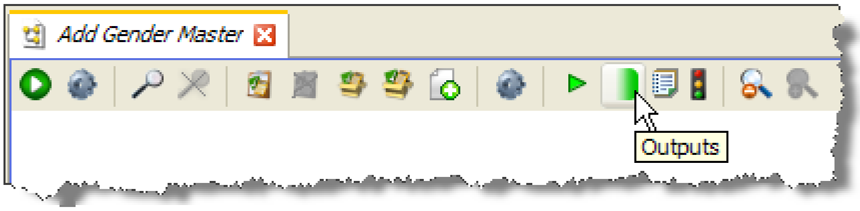


Forename Gender Map (GB Gender from Forename)
4 - Derive Gender from Forename » Reference Data (Value Map)
Title Gender Map (GB Gender from Title)
1 - Derive Gender from Title » Reference Data (Value Map)

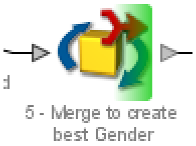
Setting Output Attributes

The Output Attributes of the new processor are set to the output attributes of any one (but only one) of the member processors.

By default, the final member processor in the sequence is used for the Output Attributes of the created processor. To use a different member processor for the output attributes, click on it, and select the **Outputs** icon on the toolbar:



The member processor used for Outputs is marked with a green shading on its output side:

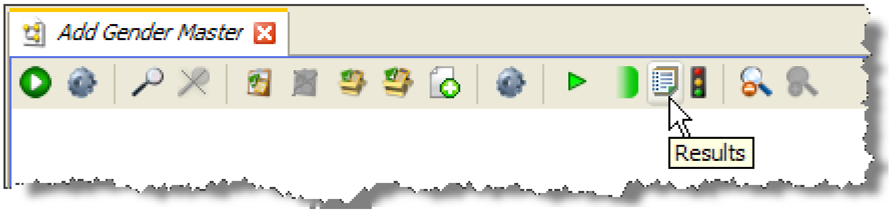


> **Note:**
>
> Attributes that appear in Results Views are always exposed as output attributes of the new processor. You may need to add a member processor to profile or check the output attributes that you want to expose, and set it as the Results Processor (see below) to ensure that you see only the output attributes that you require in the new processor (and not for example input attributes to a transformation processor). Alternatively, if you do not require a Results View, you can unset it and the exposed output attributes will always be those of the Outputs processor only.

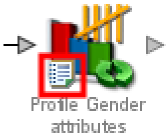
Setting Results Views

The Results Views of the new processor are set to those of any one (but only one) of the member processors.

By default, the final member processor in the sequence is used for the Results of the created processor. To use a different member processor for the results views, click on it, and select the **Results** icon on the toolbar:

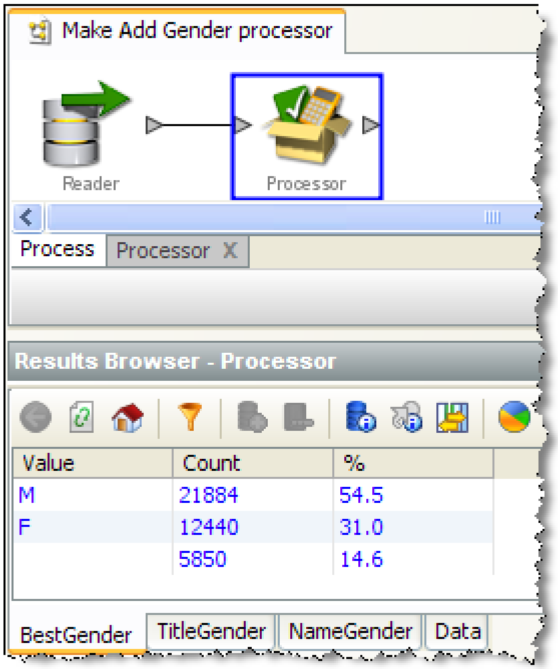


The member processor used for Results is now marked with an overlay icon:



Note that in some cases, you may want to add a member processor specifically for the purpose of providing Results Views. In our example, we may want to add a Frequency Profiler of the three output attributes (TitleGender, ForenameGender and BestGender) so that the user of a new processor can see a breakdown of what the Add Gender processor has done. To do this, we add a Frequency Profiler in the processor design view, select the three attributes as inputs, select it as our Results Processor and run it.

If we exit the processor designer view, we can see that the results of the Frequency Profiler are used as the results of the new processor:

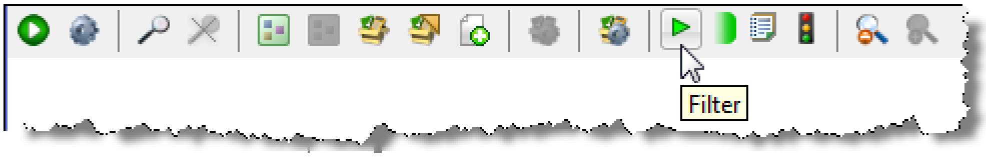


Value	Count	%
M	21884	54.5
F	12440	31.0
	5850	14.6

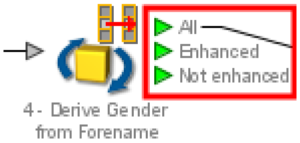
Setting Output Filters

The Output Filters of the new processor are set to those of any one (and only one) of the member processors.

By default, the final member processor in the sequence is used for the Output Filters of the created processor. To use a different member processor, click on it, and select the **Filter** button on the toolbar:



The selected Output Filters are colored green in the processor design view to indicate that they will be exposed on the new processor:

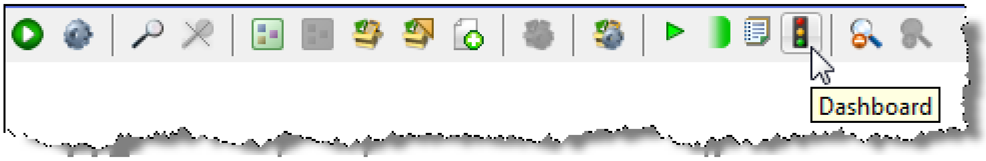


Setting Dashboard Publication Options

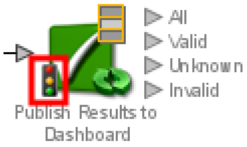
The Dashboard Publication Options of the new processor are set to those of any one (and only one) of the member processors.

If you require results from your new processor to be published to the Dashboard, you need to have an Audit processor as one of your member processors.

To select a member processor as the Dashboard processor, click on it and select the **Dashboard** icon on the toolbar:



The processor is then marked with a traffic light icon to indicate that it is the Dashboard Processor:



> **Note:**
>
> In most cases, it is advisable to use the same member processor for Results Views, Output Filters, and Dashboard Publication options for consistent results when using the new processor. This is particularly true when designing a processor designed to check data.

Setting a Custom Icon

You may want to add a custom icon to the new processor before publishing it for others to use. This can be done for any processor simply by double-clicking on the processor (outside of the processor design view) and selecting the **Icon & Group** tab.

See the Customizing Processor Icons for more details.

Once you have finished designing and testing your new processor, the next step is to publish it for others to use.

Customizing Processor Icons

It is possible to customize the icon for any processor instance in EDQ. This is one way of distinguishing a configured processor, which may have a very specific purpose, from its generic underlying processor. For example, a Lookup Check processor may be checking data against a specific set of purchased or freely available reference data, and it may be useful to indicate that reference data graphically in a process.

The customization of a processor icon is also useful when creating and publishing new processors. When a processor has been published, its customized icons becomes the default icon when using the processor from the Tool Palette.

To customize a processor icon:

1. Double-click on a processor on the Canvas
2. Select the Icon & Family tab
3. To change the processor icon (which appears at the top right of the image), use the left side of the screen.
4. To change the family icon, use the right side of the screen (Note that when publishing a processor, it will be published into the selected group, or a new family created if it does not yet exist)
5. For both processor and family icons, a dialog is launched showing the server image library. You can either select an existing image, or create a new image.
6. If adding a new image, a dialog is shown allowing you to browse for (or drag and drop) an image, resize it, and enter a name and optional description.

Once an image has been created on a server, it is added to the server's image library and available whenever customizing an icon. The image library on a server can be accessed by right-clicking on a server in the Project Browser, and selecting **Images**...

Publishing Processors

Configured single processors can be published to the Tool Palette for other users to use on data quality projects.

It is particularly useful to publish the following types of processor, as their configuration can easily be used on other data sets:

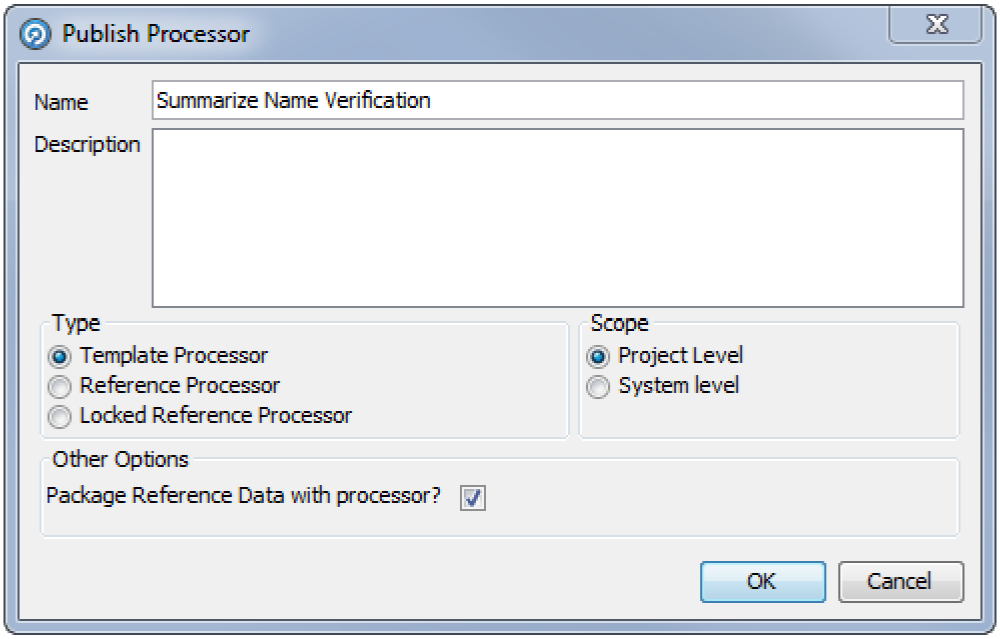
- Match processors (where all configuration is based on Identifiers)
- Parse processors (where all configuration is based on mapped attributes)
- Processors that have been created in EDQ (where configuration is based on configured inputs)

Published processors appear both in the Tool Palette, for use in processes, and in the Project Browser, so that they can be packaged for import onto other EDQ instances.

> **Note:**
>
> The icon of the processor may be customized before publication. This also allows you to publish processors into new families in the Tool Palette.

To publish a configured processor, use the following procedure:

1. Right-click on the processor, and select **Publish Processor**. The following dialog is displayed:



Name: Summarize Name Verification

Description:

Type
- ● Template Processor
- ○ Reference Processor
- ○ Locked Reference Processor

Scope
- ● Project Level
- ○ System level

Other Options

Package Reference Data with processor? ☑

2. In the **Name** field, enter a name for the processor as it will appear on the Tool Palette.
3. If necessary, enter further details in the **Description** field.
4. Select the Published processor Type: Template, Reference, or Locked Reference.
5. Select the Scope: Project (the processor is available in the current project only) or System (the processor is available for use in all projects on the system).
6. If you want to package the associated Reference Data with this published processor, select the **Package Reference Data with processor** checkbox.

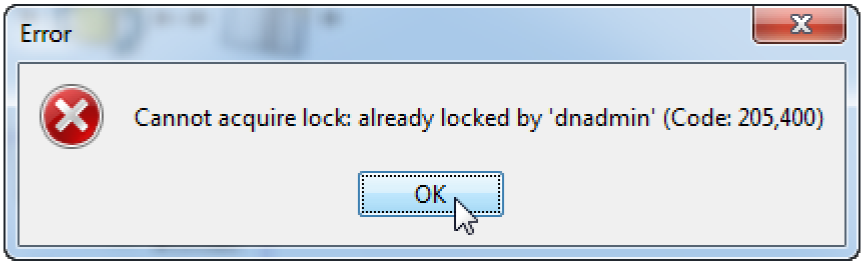
> **Note:**
>
> Options that externalized on the published processor always require Reference Data to be made available (either in the project or at system level. Options that are not externalized on the published processor can either have their Reference Data supplied with the published processor (the default behavior with this option selected) or can still require Reference Data to be made available. For example, to use a standard system-level Reference Data set.

Editing a Published Processor

Published processors can be edited in the same way as a normal processor, although they must be republished once any changes have been made.

If a Template Published processor is edited and published, only subsequent instances of that processor will be affected, as there is no actual link between the original and any instances.

If a Reference or Locked Reference Published processor is reconfigured, all instances of the process will be modified accordingly. However, if an instance of the processor is in use when the original is republished, the following dialog is displayed:



Cannot acquire lock: already locked by 'dnadmin' (Code: 205,400)

Attaching Help to Published Processors

It is possible to attach Online Help before publishing a processor, so that users of it can understand what the processor is intended to do.

The Online Help must be attached as a zip file containing an file named index.htm (or index.html), which will act as the main help page for the published processors. Other html pages, as well as images, may be included in the zip file and embedded in, or linked from, the main help page. This is designed so that a help page can be designed using any HTML editor, saved as an HTML file called index.htm and zipped up with any dependent files.

To do this, right-click the published processor and select **Attach Help**. This will open a file browsing dialog which is used to locate and select the file.

> **Note:**
>
> The **Set Help Location** option is used to specify a path to a help file or files, rather than attaching them to a processor. This option is intended for Solutions Development use only.

If a processor has help attached to it, the help can be accessed by the user by selecting the processor and pressing F1. Note that help files for published processors are not integrated with the standard EDQ Online Help that is shipped with the product, so are not listed in its index and cannot be found by search.

Publishing Processors Into Families

It is possible to publish a collection of published processors with a similar purpose into a family on the Tool Palette. For example, you may create a number of processors for working with a particular type of data and publish them all into their own family.

To do this, you must customize the family icon of each processor before publication, and select the same icon for all the processors you want to publish into the same family. When the processor is published, the family icon is displayed in the Tool Palette, and all processors that have been published and which use that family icon will appear in that family. The family will have the same name as the name given to the family icon.

For more information, see:

- *Understanding Enterprise Data Quality*

Using the Event Log

The Event Log provides a complete history of all jobs and tasks that have run on an EDQ server.

By default, the most recent completed events of all types are shown in the log. However, you can filter the events using a number of criteria to display the events that you want to see. It is also possible to tailor the Event Log by changing the columns that are displayed in the top-level view. Double-clicking on an event will display further information where it is available.

The displayed view of events by any column can be sorted as required. However, older events are not displayed by default, so a filter must be applied before sorting before they can be viewed.

About Logged Events

An event is added to the Event Log whenever a Job, Task, or System Task either starts or finishes.

Tasks are run either as part of Jobs or individually instigated using the Director UI.

The following types of Task are logged:

- Process
- Snapshot
- Export
- Results Export
- External Task
- File Download

The following types of System Task are logged:

- OFB - a System Task meaning 'Optimize for Browse' - this optimizes written results for browsing in the Results Browser by indexing the data to enable sorting and filtering of the data. The 'OFB' task will normally run immediately after a Snapshot or Process task has run, but may also be manually instigated using the EDQ client by right-clicking on a set of Staged Data and selecting **Enable Sort/ Filter**, or by a user attempting to sort or filter on a non-optimized column, and choosing to optimize it immediately.
- DASHBOARD - a System Task to publish results to the Dashboard. This runs immediately after a Process task has been run with the **Publish to Dashboard** option checked.

About Server Selection

If the Director UI is connected to multiple servers, you can switch servers using the **Server** drop-down field in the top-left hand corner.

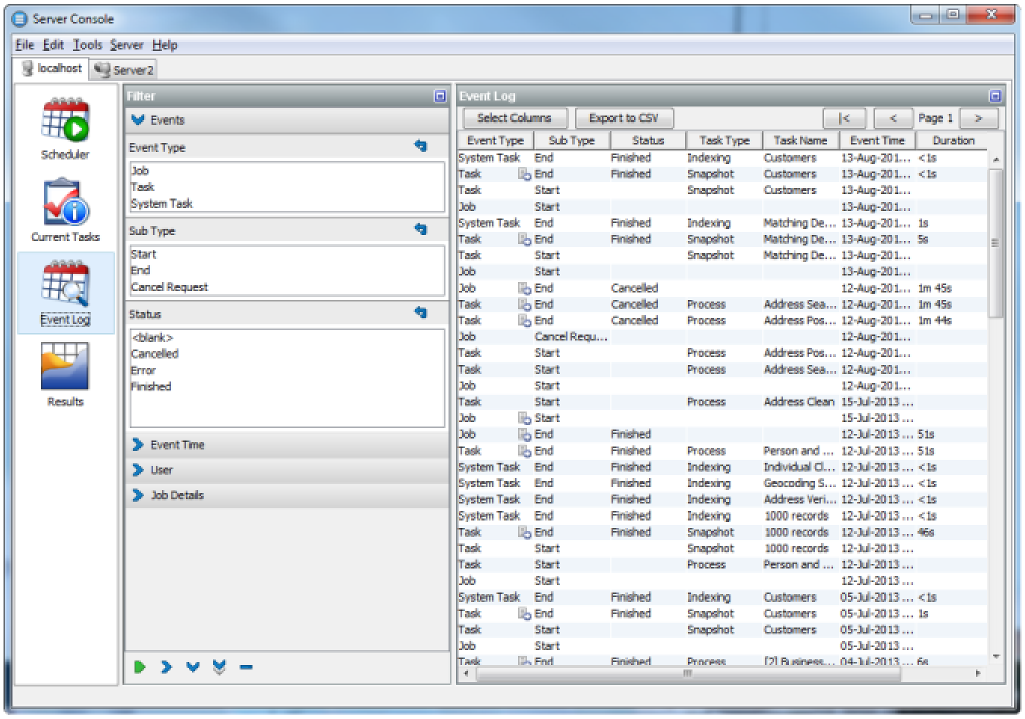
If Server Console UI is connected to multiple servers, select the required server in the tab list at the top of the window.

About Filtering Events

Quick filters

Quick filter options are made available to filter by **Event Type**, **Status** and **Task Type**. Simply select the values that you want to include in the filter (using Control - Select to select multiple items) and click on the **Run Filter** button on the bottom left of the screen to filter the events.

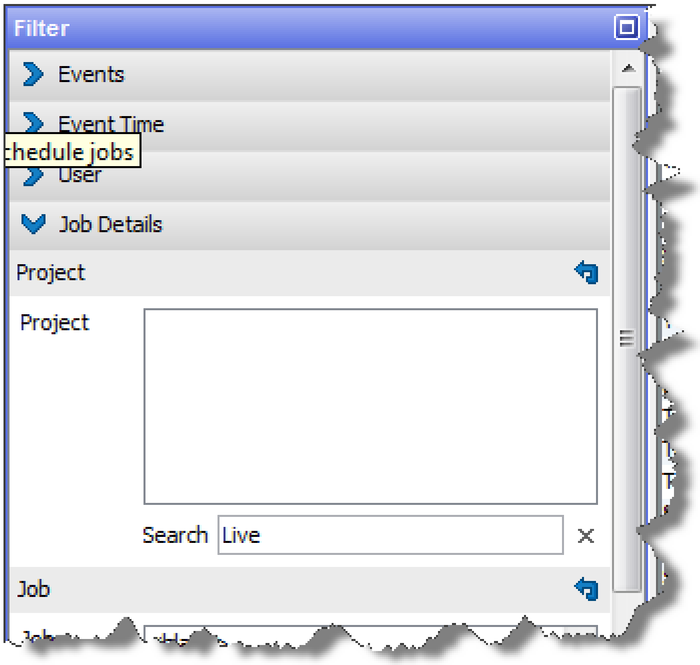
For example:



Event Type	Sub Type	Status	Task Type	Task Name	Event Time	Duration
System Task	End	Finished	Indexing	Customers	13-Aug-201...	< 1s
Task	End	Finished	Snapshot	Customers	13-Aug-201...	< 1s
Task	Start		Snapshot	Customers	13-Aug-201...	
Job	Start				13-Aug-201...	
System Task	End	Finished	Indexing	Matching De...	13-Aug-201...	1s
Task	End	Finished	Snapshot	Matching De...	13-Aug-201...	5s
Task	Start		Snapshot	Matching De...	13-Aug-201...	
Job	Start				13-Aug-201...	
Job	End	Cancelled			12-Aug-201...	1m 45s
Task	End	Cancelled	Process	Address Sea...	12-Aug-201...	1m 45s
Task	End	Cancelled	Process	Address Pos...	12-Aug-201...	1m 44s
Job	Cancel Requ...				12-Aug-201...	
Task	Start		Process	Address Pos...	12-Aug-201...	
Task	Start		Process	Address Sea...	12-Aug-201...	
Job	Start				12-Aug-201...	
Task	Start		Process	Address Clean	15-Jul-2013 ...	
Job	Start				15-Jul-2013 ...	
Job	End	Finished			12-Jul-2013 ...	51s
Task	End	Finished	Process	Person and ...	12-Jul-2013 ...	51s
System Task	End	Finished	Indexing	Individual Cl...	12-Jul-2013 ...	< 1s
System Task	End	Finished	Indexing	Geocoding S...	12-Jul-2013 ...	< 1s
System Task	End	Finished	Indexing	Address Veri...	12-Jul-2013 ...	< 1s
System Task	End	Finished	Indexing	1000 records	12-Jul-2013 ...	< 1s
Task	End	Finished	Snapshot	1000 records	12-Jul-2013 ...	46s
Task	Start		Snapshot	1000 records	12-Jul-2013 ...	
Task	Start		Process	Person and ...	12-Jul-2013 ...	
Job	Start				12-Jul-2013 ...	
System Task	End	Finished	Indexing	Customers	05-Jul-2013 ...	< 1s
Task	End	Finished	Snapshot	Customers	05-Jul-2013 ...	1s
Task	Start		Snapshot	Customers	05-Jul-2013 ...	
Job	Start				05-Jul-2013 ...	
Task	End	Finished	Process	[2] Business...	04-Jul-2013 ...	6s

Free-text filters (search ahead)

Further free-text filtering options are available to filter by **Project Name**, **Job Name**, **Task Name** and **User Name**. These are free-text so that you can enter partial names into the fields. You can enter a partial name into any of these fields - provided the object contains the partial name, it will be displayed (though note that matching is case-sensitive). For example, if you use a naming convention where all projects working on live systems have a name including the word 'Live' you can display all events for live systems as follows:



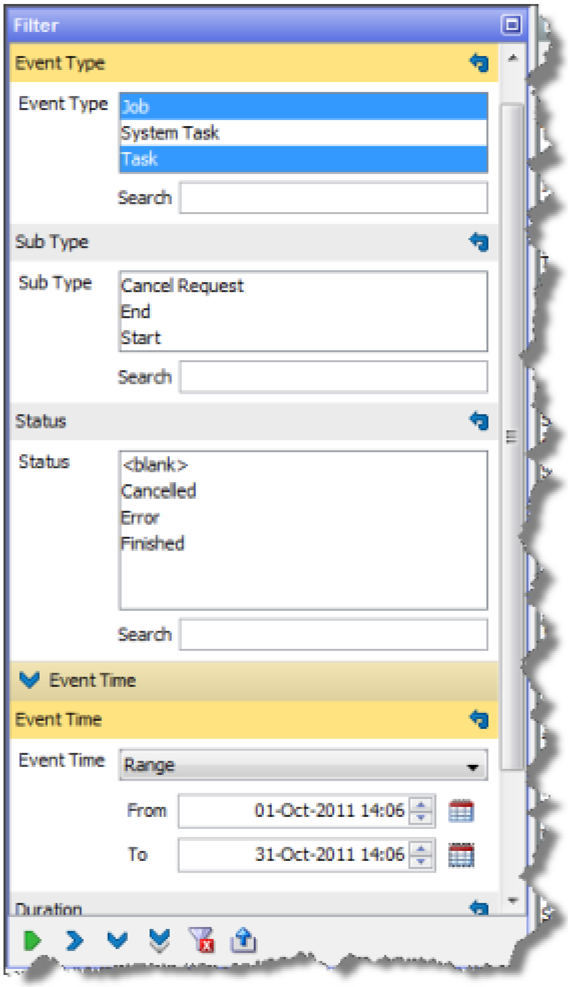
> **Note:**
>
> The Project Name column is not displayed by default. To change the view to see it, click the **Select Columns** button on the left hand side, and check the Project Name box.

Date/time filters

The final set of filters, on the right-hand side of the screen, allow you to filter the list of events by date and time. A Date picker is provided to make it easier to specify a given date. Note that although only the most recent events are shown when accessing the Event Log, it is possible to apply filters to view older events if required.

> **Note:**
>
> Events are never deleted from the history by EDQ, though they are stored in the repository and may be subject to any custom database-level archival or deletion policies that have been configured on the repository database.

Events may be filtered by their start times and/or by their end times. For example, to see all Jobs and Tasks (but not System Tasks) that completed in the month of November 2008, apply filters as follows:



Filter

Event Type

Event Type: Job, System Task, Task

Search:

Sub Type

Sub Type: Cancel Request, End, Start

Search:

Status

Status: <blank>, Cancelled, Error, Finished

Search:

Event Time

Event Time

Event Time: Range

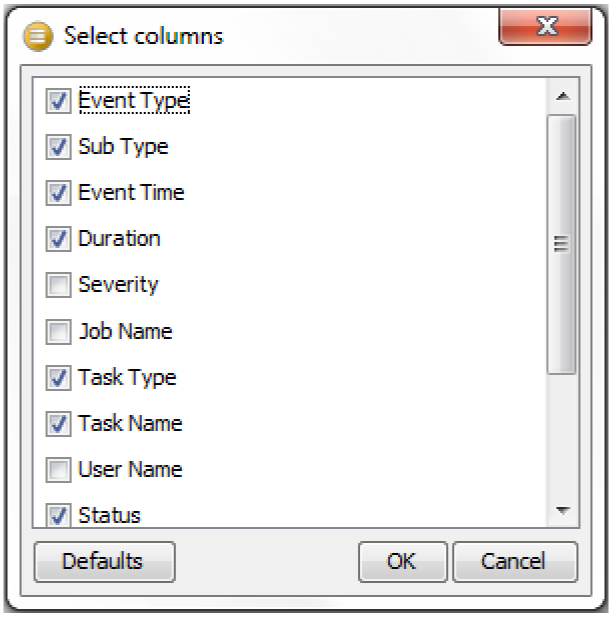
From: 01-Oct-2011 14:06

To: 31-Oct-2011 14:06

Duration

Column selection

To change the set of columns that are displayed on the Event Log, click the **Select Columns** button on the top left of the **Event Log** area. The Select Columns dialog is displayed. Select or deselect the columns as required, and click OK or save or Cancel to abandon the changes. Alternatively, click Default to restore the default settings:



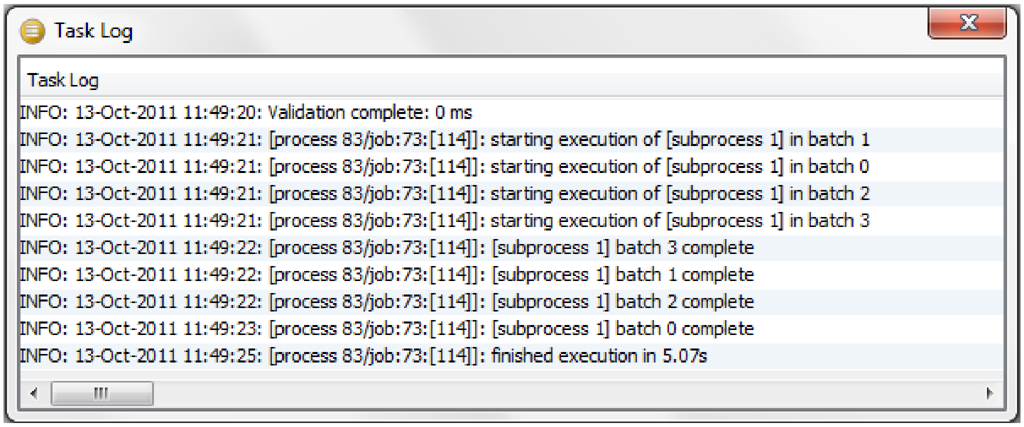
- ☑ Event Type
- ☑ Sub Type
- ☑ Event Time
- ☑ Duration
- □ Severity
- □ Job Name
- ☑ Task Type
- ☑ Task Name
- □ User Name
- ☑ Status

Note that **Severity** is a rarely used column - it is currently set to 50 for tasks or jobs that completed correctly, and 100 for tasks or jobs that raised an error or a warning.

Opening an event

Double-clicking to open an event will reveal further detail where it is available.

Opening a Task will display the Task Log, showing any messages that were generated as the task ran:



```
INFO: 13-Oct-2011 11:49:20: Validation complete: 0 ms
INFO: 13-Oct-2011 11:49:21: [process 83/job:73:[114]]: starting execution of [subprocess 1] in batch 1
INFO: 13-Oct-2011 11:49:21: [process 83/job:73:[114]]: starting execution of [subprocess 1] in batch 0
INFO: 13-Oct-2011 11:49:21: [process 83/job:73:[114]]: starting execution of [subprocess 1] in batch 2
INFO: 13-Oct-2011 11:49:21: [process 83/job:73:[114]]: starting execution of [subprocess 1] in batch 3
INFO: 13-Oct-2011 11:49:22: [process 83/job:73:[114]]: [subprocess 1] batch 3 complete
INFO: 13-Oct-2011 11:49:22: [process 83/job:73:[114]]: [subprocess 1] batch 1 complete
INFO: 13-Oct-2011 11:49:22: [process 83/job:73:[114]]: [subprocess 1] batch 2 complete
INFO: 13-Oct-2011 11:49:23: [process 83/job:73:[114]]: [subprocess 1] batch 0 complete
INFO: 13-Oct-2011 11:49:25: [process 83/job:73:[114]]: finished execution in 5.07s
```

Note:

Messages are classified as INFO, WARNING, or SEVERE. An INFO message is for information purposes and does not indicate a problem. A WARNING message is generated to indicate that there could be an issue with the process configuration (or data), but this will not cause the task to error. SEVERE messages are generated for errors in the task.

For Jobs, if a notification email was configured on the job, the notification email will be displayed in a web browser when opening the completed event for the Job. Jobs with no notifications set up hold no further information.

Exporting data from the Event Log

It is possible to export the viewable data in the Event Log to a CSV file. This may be useful if you are in contact with Oracle Support and they require details of what has run on the server.

To export the current view of events, click **Export to CSV**. This will launch a browser on the client for where to write the CSV file. Give the file a name and click **Export** to write the file.

Part IV

Reference Information

Depending on the tasks you want to perform with Data Integration Platform Cloud, you may need to perform additional configuration steps for your Agents, Connections, and your on-premises environments.

Topics:

16
Configure Remote Agents

Depending on the type of Task you want to perform in Data Integration Platform Cloud, there are additional agent configuration steps.

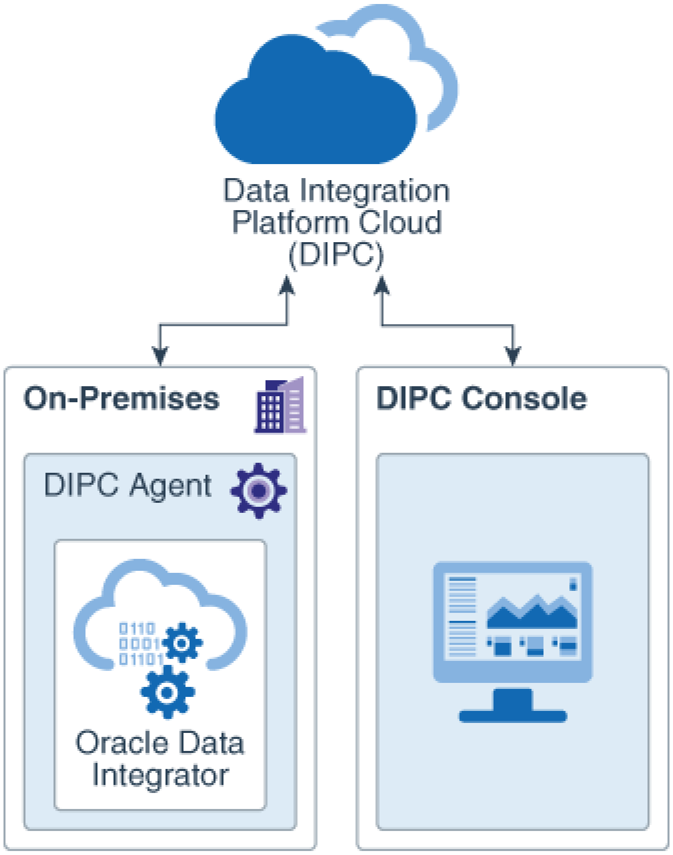
Topics:

- Set up a Remote Agent for ODI
- Set up an External Library Path for GoldenGate 12c
- Set up Remote Agent for Autonomous Data Warehouse Cloud
- Set up a Remote Agent for Big Data
- Upgrade an On-Premises Agent

Set up a Remote Agent for ODI

When you select Data Integrator (ODI) in the Download Agent Installer dialog, ODI binaries are included in your Agent package. You need these bits when performing Synchronize Data and ODI Execution Tasks.

The ODI plug-in for Agents enables you to perform data integration tasks remotely in the on-premises environment. Once downloaded and configured, the Agent starts working on the remote environment. The Agent gets registered in Data Integration Platform Cloudand appears in the Agents list.



To download and configure the Remote Agent for ODI:

1. Log in to Data Integration Platform Cloud and click Agents in the left navigation.
2. On the Agents page, click **Download Installer** and select **Data Integrator (ODI)** and **Oracle 12c (OGG)** in the Download Agent Installer dialog, and then click **Download.**
3. Create a directory called `dicloud` for installing the agent, move the agent package to that directory, and then unzip the agent package.
4. After the files are unzipped, move to the **dicloud** directory and then run the agent configuration with authentication type BASIC and the odiRemote flag.

   ```
   ./dicloudConfigureAgent.sh -authType=BASIC -odiRemote
   ```

   > **Note:**
   >
   > Port 3307 must be kept free for installing MySQL.

   For more detailed information about configuring the remote agent, see Set Your Agent Properties.
5. Follow the prompts and enter your Data Integration Platform Cloud instance name, your managed server port number, username and password. Be aware that a username confirmation is required before entering the password.

When the script completes, MySQL, your Data Integration Platform Cloud agent, GoldenGate, and the on-premises ODI are installed. You can now start the agent.

```
> cd $DIPC_AGENT_DIR/dicloud/agent/dipcagent001/bin
> ./startAgentInstance.sh
```

Set up an External Library Path for GoldenGate 12c

To configure the database library paths for a particular supported database for Oracle GoldenGate 12*c*, you must set an external library path for Oracle GoldenGate.

`ggExternalLibraryPath` is a property in agent.properties. You use it to set an external library path for Oracle GoldenGate 12*c* on Windows.

> **Note:**
>
> Make sure that you've an Oracle GoldenGate 12*c* client installed on Windows.
>
> Binaries should be the client binary of the corresponding Oracle GoldenGate 12*c* database.

To set the Oracle GoldenGate 12*c* dependent library path on Windows:

1. Edit your agent.properties file located at `<agent_unzip_loc>/dicloud/agent/<dipcagent001>/conf/agent.properties`.
2. Set `ggExternalLibraryPath=<path to oracle 12c windows lib path>;<path2>`
3. In the path or in the terminal where the agent is started, set `ORACLE_HOME` to the same value as `ggExternalLibraryPath`.
4. To download the database library binaries, go to:

To download the database library binaries:

- 12.1.0.2: https://www.oracle.com/technetwork/database/database12c-win64-download-2297732.html
- 12.2.0.1: https://www.oracle.com/technetwork/database/enterprise-edition/downloads/oracle12c-windows-3633015.html

Set up Remote Agent for Autonomous Data Warehouse Cloud

To replicate data to Oracle Autonomous Data Warehouse Cloud (ADWC) using Data Integration Platform Cloud, you must set up the remote Agent.

> **Note:**
>
> If any error message appears, you can track it from the drill-down status list of Agents.

To set up the Remote Agent for ADWC:

Perform the steps below, before you Create a Replicate Data Task for ADWC.

1. Stop the DIPC Agent using:

   ```
   <agent install location>/dicloud/agent/dipcagent001/bin/
   stopAgentInstance.sh
   ```

2. Stop/kill all running OGG processes (Manager, extract, pump, replicat, pmsrvr).
3. Download the **12.2 Instant Client** from here:

   https://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html
4. Extract the required Instant client packages listed below to disk.
5. Copy selected OCI libraries (overwrite) the files under <agent Install location>/dicloud/oci.

   From `12.2 Instant Client Package - Basic zip`, you need these files:

   `libclntsh.so.12.1`

   `libclntshcore.so.12.1`

   `libocijdbc12.so`

   `liboramysql12.so`

   `libipc1.so`

   `libnnz12.so`

   `libons.so`

   `libmql1.so`

   `libocci.so.12.1`

   From `12.2 Instant Client Package - ODBC zip`, you need this file:

   `libsqora.so.12.1`

   From `12.2 Instant Client Package - JDBC Supplement zip`, you need this file:

   `libheteroxa12.so`

From 12.2 Instant Client Package - SQL*Plus zip, you need this file:

libsqlplus.so

6. Download the client credentials zip file from ADWC console, and extract the files on disk.

   This zip file will contain cwallet.sso, ewallet.p12, keystore.jks, ojdbc.properties, sqlnet.ora, tnsnames.ora, truststore.jks files.

   - From extracted wallet, update the extracted sqlnet.ora file with correct location of the wallet.

     The sample sqlnet.ora:

     ```
     NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
     WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA =
     (DIRECTORY=<Location of extracted Wallet file>)))
     SSL_SERVER_DN_MATCH=yes
     SQLNET.USE_HTTPS_PROXY=on
     bequeath_detach=true
     SQLNET.ALLOWED_LOGON_VERSION=8
     ```

   - From extracted wallet, update the extracted tnsnames.ora with proxy and port. The use of proxy is optional. The sample tnsnames.ora:

     ```
     odidb_low = (description= (address=(https_proxy=www-proxy-
     adwc.example.com)(https_proxy_port=80)
     (protocol=tcps)(port=1522)(host=adwc.examplecloud.com))
     (connect_data=
     (service_name=@adwc_odidb_low.adwc.examplecloud.com))
     (security=(ssl_server_cert_dn=
     "CN=adwc.examplecloud.com,OU=Example BMCS
     Example,O=Example,L=Example
     City,ST=Example,C=Example"))   )
     ```

7. Copy the updated sqlnet.ora and tnsnames.ora files to <agent install location>/dicloud/agent/dipcagent001/conf directory.

8. Set environment variables. The following examples are of a correct setting sequence:

   - Set TNS_ADMIN environment variable to <agent Install location> / dicloud/agent/dipcagent001/conf.
   - Set LD_LIBRARY_PATH environment variable to <agent install location>/dicloud/oci.
   - Unset ORACLE_HOME environment variable.

9. Start the DIPC agent with <agent install location>/dicloud/agent/dipcagent001/bin/startAgentInstance.sh.

10. You may encounter this error, if the DIPC agent is manually installed:

    ```
    IO Error: IO Error Received fatal alert: handshake_failure,
    Authentication lapse 0 ms.
    ```

When the agent is deployed, it uses JAVA. But when you restart the agent , or install it manually you should specify the Java present in the machine, where agent is installed. Use a Java version JDK 1.8.0_161 or higher. If this Java version is below JDK 1.8.0_161, execute the following steps:

- Download and install Java Cryptology Extension Unlimited Strength Jurisdiction Policy Files. It contains files - `local_policy.jar`, `README.txt`, and `US_export_policy.jar` . Copy these files to `JDK_HOME\jre\lib\security` folder.

  https://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

- Open `JDK_HOME\jre\lib\security\java.security`. You can see the following entries there:

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

- Add the following line below the above-listed entries:

  ```
  security.provider.11=oracle.security.pki.OraclePKIProvider
  ```

- Restart the Agent.

Set up a Remote Agent for Big Data

When you select Big Data (OGG) in the Download Agent Installer dialog, the GoldenGate for Big Data binaries are included in your Agent package.

Oracle GoldenGate Big Data is configured the same way as the other GoldenGate Homes. You must set JAVA_HOME environment variable on the Agent’s host machine. The install directory is referred as `GG_HOME`. In the `agent.properties` file, the following properties are configured by default:

```
ggmonbigdata.ggPerfMetricsServerPort = <7830>

ggmonbigdata.agentGoldenGateHome = <path to gghomebigdata>

ggmonbigdata.ggRepoUdpPort = <9950>

ggmonbigdata.ggInstancePort = <7829>
```

You can override the default settings if you need to.

To configure Oracle GoldenGate agent for Big data:

1. After you download the Agent Installer package, extract the files.
2. Run the `dicloudConfigureAgent` script located in the `dicloud` directory.

For more information on configuring an agent, see Set Your Agent Properties

Upgrade an On-Premises Agent

You can upgrade an on-premises DI agent using certain command prompts relevant to the chosen environment.

To upgrade an on-premises agent:

1. Log in to Data Integration Platform Cloud.
2. From the left navigation pane, click **Agents**.
3. From the **Download Installer** menu, select the available agent package.
4. In the Download Agent Installer window, click **OK.**

   An agent package bundle for your platform is downloaded to your machine. You can unzip, configure, and start the Agent.
5. Unzip to <AgentTempUnzipLoc>
6. Stop the running/old agent by calling the stop agent script.
7. Launch a Command prompt (@ <OldAgentUnzipLoc>/gghome) and set following environment variables :
   a. For Linux (Oracle 12c): Set LD_LIBRARY_PATH :: <OldAgentUnzipLoc>/gghome/oci
   b. For Linux (Oracle 11g): Set LD_LIBRARY_PATH :: <OldAgentUnzipLoc>/gghome/oci11g
   c. For Windows (Oracle 12c): Set Path : <OldAgentUnzipLoc>/gghome/lib12
8. Launch GGSCI by executing:
   a. <OldAgentUnzipLoc>/gghome/ggsci for Oracle 12c
   b. <OldAgentUnzipLoc>/gghome11g/ggsci for Oracle 11g
9. Execute "INFO ALL" to see the list of running GG processes
10. Stop all the Running processes by executing "STOP <ProcessName>"
11. Exit GGSCI.
12. Run the script <AgentTempUnzipLoc>/dicloud/dicloudUpgradeAgent.bat<old_agent_oracle_home>
13. Once the agent upgrade is finished - start the agent
14. Repeat steps 7 to 10.
15. Restart the processes stopped in step 12 by using the "START <processname>" command.

Example 16-1 Syntax of the Command

```
${AgentTempUnzipLoc}/dicloudUpgradeAgent.sh <old_agent_oracle_home>   [-
debug ]
<old_agent_oracle_home>  = <old_agent_unzip_loc>/dicloud/agent/oracle
[-debug ]                = Generates debug log for this run.
[Optional]
Example :
```

```
$./scratch/MyAgentTempUnzipLoc/dicloud/dicloudUpgradeAgent.sh /scratch/
work/install/MyDIPCAgentunzipLoc/dicloud/agent/oracle
```

17 Data Source Connection Details

Prepare and create connections to your source and target data sources for your tasks in Oracle Data Integration Platform Cloud.

Topics:

Create a File Connection

Learn to create a connection to a file in Oracle Data Integration Platform Cloud.

1. From the Getting Started section of the Home page, click **Create** from the Connection tile or from the Create menu in the Catalog, select **Connection**. You can also create Connections from any Task screen.
2. Complete the fields in the General Information section. The Identifier is to identify this connection through a string with no spaces, but is not currently used. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
   - For **Agent Name**, select the agent you have created for the this connection.
   - For Type, select **File**.
3. Click **Test Connection**. If successful, click **Save**.

Create a Kafka Connect Connection

Learn to create a connection to Kafka Connect in Oracle Data Integration Platform Cloud.

Set up Format for Kafka Topics

Before you create a Kafka Connect connection in Oracle Data Integration Platform Cloud, you must choose a format for your Kafka topics in the Kafka producer properties file.

1. Save your Kafka producer properties file in the `${agent_unzip_loc}/dicloud/gghomebigdata/dirprm` directory:

   - To use the provided example file:

     ```
     cp ${agent_unzip_loc}/dicloud/gghomebigdata/AdapterExamples/big-
     data/kafka_connect/kafkaconnect.properties$ ${agent_unzip_loc}/
     dicloud/gghomebigdata/dirprm/kafkaconnect.properties
     ```

   - To use your own producer file, rename your `producer.properties` file to `kafkaconnect.properties` and copy it to `${agent_unzip_loc}/dicloud/gghomebigdata/dirprm`. Review the provided example file for reference.

   In order to see the `gghomebigdata` folder , you must have already downloaded your agent with the **Big Data (OGG)** component and **registered** it with Oracle Data Integration Platform Cloud.

2. Update the `kafkaconnect.properties` file to set the delivery format for Kafka topics to either **JSON** or **Avro**.

   - For Avro, uncomment the Avro section of the kafkaconnect.properties file and comment out the JSON section.
     Having an Avro format, by only uncommenting the JSON section sends schema information in JSON and Avro formats. To omit the JSON schema information from the messages set the following:

     ```
     key.converter.schemas.enable=false
     value.converter.schemas.enable=false
     ```

     Note:

   - For JSON leave the Avro section commented out, as is.

   Here's the example `kafkaconnect.properties` file for messages to be delivered in a JSON format:

   ```
   #JSON Converter Settings
   key.converter=org.apache.kafka.connect.json.JsonConverter
   key.converter.schemas.enable=true
   value.converter=org.apache.kafka.connect.json.JsonConverter
   value.converter.schemas.enable=true

   #Avro Converter Settings
   #key.converter=io.confluent.connect.avro.AvroConverter
   #value.converter=io.confluent.connect.avro.AvroConverter
   #key.converter.schema.registry.url=http://localhost:8081
   #value.converter.schema.registry.url=http://localhost:8081


   #Adjust for performance
   ```

```
buffer.memory=33554432
batch.size=16384
linger.ms=0
```

The `batch.size` property is the maximum number of bytes to buffer before sending data to Kafka. The `linger.ms` property is the maximum milliseconds to wait before sending data. Setting `linger.ms` to zero causes messages to be sent immediately to Kafka. If you set both properties, data is sent to Kafka according to whichever property reaches its limit first.

Create the Connection

1. From the Home page or from the Create menu in the Catalog, select **Create Connection**.
2. Complete the fields in the **General Information** section.
   - The **Identifier** is to identify this connection through a string with no space. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
   - For **Agent**, select the agent you have created for the this connection.
   - For **Type**, select **Kafka Connect**.
3. In the **Connection Settings**, enter the location of the following files to append to the system's **classpath**.
   - For **JSON** format
     - Kafka client libraries
   - For **Avro** format
     - Kafka client libraries
     - Confluent Avro converters
     - Schema registry

   The Kafka Connect client libraries do not ship with the Big Data (OGG) component that you download and register with the Data Integration Platform Cloud remote agent. You must obtain the Kafka Connect client libraries that match the version of Kafka in Kafka Connect and point to them in the **Java Classpath** field in the Connection Settings. For a list of the required client JAR files by version, see Kafka Connect Client Libraries.

   The default location of the Kafka Connect client JARs is the `{kafka_connect_install_dir}/libs/*` directory. Use the * wildcard character to include all of the JAR files from each of the associated directories. Do not use *.jar. And make sure you don’t have spaces in the classpath. Here is an example:

   ```
   dirprm:{kafka_connect_install_dir}/share/java/kafka-serde-tools/*:
   {kafka_connect_install_dir}/share/java/kafka/*:
   {kafka_connect_install_dir}/share/java/confluent-common/*
   ```

4. In the **Properties** section, for Kafka Producer Config file, enter `kafkaconnect.properties` or the file name you created under `/gghomebigdata/dirprm` if you used your own file or changed it from kafkaconnect.properties to

something else. Either way, this file must be located in `${agent_unzip_loc}/dicloud/gghomebigdata/dirprm` for the connection to work.

5. Complete the rest of the properties listed in the **Properties** section based on the following explanations. Oracle Data Integration Platform Cloud's Kafka Connect handler will publish messages to Kafka topics based on these properties.

Table 17-1 Kafka Connect Connection Properties

Property	Default Value Shown in the Field	Explanation
Topic Mapping Template	`${fullyQualifiedTablename}`	This value will be the name of your topic. If you choose `${fullyQualifiedTablename}`, the fully qualified table name includes a period (.) delimiter between the schema and table names. For example, if there's a transaction with an insert operation in a data source with schema `ABC`, in table `X`, then the Kafka producer will create a Kafka topic named `ABC.X` and add that inserted data to the `ABC.X` topic. In case of multiple databases, such as databases with PDBs, then the `${fullyQualifiedTableName}` is `database_name.schema_name.table_name`. For example, there is a salesdb.contacts table and the contacts table has a column called `first_name`. Then the Kafka topic will be `salesdb.contacts.first_name`. You can either select one of the values from the drop-down menu or customize your topic names. This field is editable, For example, a Kafka topic name can be `${schemaName}_${tableName}` **Note:** Custom topic names are not validated. If you enter something that is not in a predefined template keywords, then the value will not be resolved. For example, `{date}_{table}` is not valid and all your data will go to one topic that is called `{date}_{table}`. For correct template names and an explanation of all your options for this field, field, see **Template Keywords for Kafka Connect Connection** table on this page. You can override the value of Topic Mapping Template later, when you set up a Replicate Data Task with a

Table 17-1 (Cont.) Kafka Connect Connection Properties

Property	Default Value Shown in the Field	Explanation
		Kafka Connect target. This field is called **Mapping Pattern** in the Replicate Data Task.

Table 17-1 (Cont.) Kafka Connect Connection Properties

Property	Default Value Shown in the Field	Explanation
Key Mapping Template	`${primaryKeys}`	When topics have keys attached to them, messages with the same key go to the same partition in a topic. Kafka offers order within a partition, but not across partitions in a topic, so having keys provides order in the partitions. Choose a key for the partitions in your topic. The default key for a topic is the primary key. That means that the data in each partition have the same primary key. You can either select the default primary key value or one of the values from the drop-down menu or customize your key and decide how you want your topics to be partitioned. This field is editable, For example, a Kafka topic key can be `${opType}` and all the inserts will go in one partition and all the updates in another one, for each topic. **Note:** Custom topic names are not validated. If you enter something that is not in a predefined template keywords, then the value will not be resolved. For example, `{date}_{table}` is not valid and all your data will go to one topic that is called `{date}_{table}`. Custom topic names are not validated. If you enter something that is not in a predefined template keywords, then the value will not be resolved. For example, `{date}_{table}` is not valid and all your data will go to one topic that is called `{date}_{table}`. For correct template names and an explanation of all your options for this field, see **Template Keywords for Kafka Connect Connection** table on this page. You can override the value of Key Mapping Template later, when you set up a Replicate Data Task with a

Table 17-1 (Cont.) Kafka Connect Connection Properties

Property	Default Value Shown in the Field	Explanation
		Kafka Connect target. This field is called **Key Mapping Pattern** in the Replicate Data Task.
Include Table Name	(Selected)	If you want the fully qualified table name to be included in the messages that are delivered to your topics, then select this option. Your messages will then include a field called `table` for which the value is the fully qualified table name.
Include Op Type	(Selected)	If you want the type of source operation to be included in the messages that are delivered to your topics, select this option. Your messages will then include a field called `op_type` with the value `I` for insert, `U` for update, `D` for delete and `T` for truncate. You can change the letters that represent these operations to other letters or words in the following fields available in the Properties section of the Kafka Connect Connection dialog: `insertOpKey`, `updateOpKey`, `deleteOpKey` and `truncateOpKey`.
Include Op Timestamp	(Selected)	If you want a time stamp for when the source operation was committed to be included with your messages, then select this option. Your messages will then include a field called `op_ts` for which the value is the operation timestamp (commit timestamp) from the source transaction.
Include Current Timestamp	(Selected)	If you want a time stamp for when the message was delivered to your topic, then select this option. Your messages will then include a field called `current_ts` for which the value is the current timestamp of when the handler delivered the message.

Table 17-1 (Cont.) Kafka Connect Connection Properties

Property	Default Value Shown in the Field	Explanation
Include Position	(Selected)	When a Replicate Data Task starts, Oracle Data Integration Platform Cloud creates a starting position with a unique sequence number in a file on the source called a trail file. Then with every transaction, it moves its position to the position of last record read in the data source. Select this option to create a field in the output messages called `pos` for which the value is the position (sequence number + offset) of the transactions from the data source trail file.
Include Primary Keys	(Deselected)	Select this option to include a field in your messages called `primary_keys`. The value of this field is an array of names of the primary key columns.
Include Tokens	(Deselected)	Select to include a map field in your messages called maps. The key is tokens and the value is a map where the keys and values are the token keys and values from the trail file described in the **IncludePosition** property. Each change record written by Oracle Data Integration Platform Cloud to a trail file includes a header area, a data area, and possibly a user token area. The record header contains information about the transaction environment, and the data area contains the actual data values that were extracted. The token area contains information that is specified by users for use in column mapping and conversion.

Table 17-1 (Cont.) Kafka Connect Connection Properties

Property	Default Value Shown in the Field	Explanation
Message Formatting	`row`	Select row for the output messages to be modeled as a row in each topic. Set to op for the output messages to be modeled with an operation format. Row applies to both JSON and Avro. Op only applies to Avro. If you have chosen JSON for the delivery format, then op will be ignored, even if you select it. **Note:** If you choose `op` for message formatting, then you can't include the operation type (`insert, abend, update, delete`) in your messages.
Insert Op Key	`I`	`I` is for `insert`. Keep this default, or enter a value to be used for the `insert` operations. This value will be included with your message if the operation on the delivered data is `insert`. Ensure that **includeOpType** property is already selected. This value will be included in a field called `op_type`. Leave this field blank if you don't want `insert` operations included in the Kafka topic
Update Op Key	`U`	`U` is for `update`. Keep this default, or enter a value to be used for the `update` operations. This value will be included with your message if the operation on the delivered data is `update`. Ensure that **includeOpType** property is already selected. This value will be included in a field called `op_type`. Leave this field blank if you don't want `update` operations included in the Kafka topic

Table 17-1 (Cont.) Kafka Connect Connection Properties

Property	Default Value Shown in the Field	Explanation
Delete Op Key	`D`	`D` is for `delete`. Keep this default, or enter a value to be used for the `delete` operations. This value will be included with your message if the operation on the delivered data is `delete`. Ensure that **includeOpType** property is already selected. This value will be included in a field called `op_type`. Leave this field blank if you don't want `delete` operations included in the Kafka topic
Truncate Op Key	`T`	`T` is for `truncate`. Keep this default, or enter a value to be used for the `truncate` operations. This value will be included with your message if the operation on the delivered data is `truncate`. Ensure that **includeOpType** property is already selected. This value will be included in a field called `op_type`. Leave this field blank if you don't want `truncate` operations included in the Kafka topic
Treat All Columns As Strings	(Deselected)	Select to treat all output fields as strings. If you don't select this field, then the handler will map the data type for the source data that is being delivered to the best corresponding Kafka Connect data type.
Map Large Numbers As Strings	(Deselected)	Large numbers are delivered to Kafka topics with a double-precision floating point format, which only has 64 bits. You may lose precision on your large numbers with this 64 bit limitation. You can select this field to save your large numbers as strings to preserve their precision.
ISO8601 Format	(Deselected)	Select to output the current date in the ISO8601 format. An example of this format is `2018-05-25T14:33:46-07:00`, whereas a Unix format would be `1527284026` and an RFC 2822 format would be `Fri, 25 May 2018 14:33:46 -07:0`

Table 17-1 (Cont.) Kafka Connect Connection Properties

Property	Default Value Shown in the Field	Explanation
PK Update Handling	`insert`	This option is for handling Primary Key (PK) updates and only applies for the row message formatting. Choose `insert`, if you want insert for the updated key. For example, if the row with the old key is not found in the target, the change data will be considered an insert. Otherwise, you can choose `abend` (abnormal end), `update` or `delete`. **Note:** If you have selected `op` (operation) for the Message Formatting field, then this option will be ignored.

Table 17-2 Template Keywords for Kafka Connect Connection

Keyword	Explanation
`${catalogName}`	Resolves to the catalog name.
`${columnValue[]}`	Resolves to a column value where the key is the fully-qualified table name and the value is the column name to be resolved. For example: `${staticMap[dbo.table1=col1,dbo.table2=col2]}`
`${currentTimestamp}` or `${currentTimestamp[]}`	Resolves to the current timestamp. You can control the format of the current timestamp using the Java based formatting for the `SimpleDateFormatclass`. For example: `${currentDate}` or `${currentDate[yyyy-mm-dd hh:MM:ss.SSS]}`
`${custom[]}`	It is possible to write a custom value resolver. If required, contact Oracle Support.
`${emptyString}`	Resolves to “”.
`${fullyQualifiedTableName}`	Resolves to the fully qualified table name including the period (.) delimiter between the catalog, schema, and table names. For example: `test.dbo.table1`.

Table 17-2 (Cont.) Template Keywords for Kafka Connect Connection

Keyword	Explanation
`${groupName}`	Resolves to the name of the Replicat process. If using coordinated delivery, it resolves to the name of the Replicat process with the Replicate thread number appended.
`${null}`	Resolves to a NULL string.
`${opTimestamp}`	The operation timestamp from the source trail file.
`${opType}`	Resolves to the type of the operation: `insert`, `update`, `delete` or `truncate`.
`${position}`	The sequence number of the source trail file followed by the offset (RBA).
`${primaryKeys}`	Resolves to the concatenated primary key values delimited by an underscore (_) character.
`${schemaName}`	Resolves to the schema name.
`${staticMap[]}`	Resolves to a static value where the key is the fully-qualified table name. The keys and values are designated inside of the square brace in the following format: `${staticMap[dbo.table1=value1,dbo.table2=value2]}`
`${tableName}`	Resolves to the short table name.

Table 17-3 Example Templates

Example Template	Resolved Value
`${currentDate[yyyy-mm-dd hh:MM:ss.SSS]}`	`2018-10-17 11:45:34.254`
`${groupName}_{fullyQualfiedTableName}`	`KAFKA001_dbo.table1`
`prefix_${schemaName}_${tableName}_suffix`	`prefix_dbo_table1_suffix`

6. Click **Test Connection**. If successful, click **Save**.

Create a Microsoft SQL Server Connection

Learn to create a connection to Microsoft SQL Server in Oracle Data Integration Platform Cloud

1. From the Get Started with Integration section of the Home page or from the Create menu in the Catalog, select **Connection**. You can also create Connections from any Create Task screen.
2. Complete the fields in the **General Information** section. The Identifier is to identify this connection through a string with no spaces, but is not currently used. If you

want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.

- For Agent Name, select the agent you're running for the Microsoft SQl connection from the list of available agents.
- For Type, select **Microsoft SQL Server**

.

3. In the **Connection Settings** complete the following fields:
   - **Hostname:** Enter the Host/IP, where SQL server is running.
   - **Service:** Either port or Instance Name is mandatory. Both cannot be given together.
   - **Port:** The port where SQL Server is listening.
   - **Username:** The database user
   - **Password:** The database password
4. Click the **Database** drop-down, and select a **Database**. If not specified, it will populate the default database.
5. Click the **Schema Name** drop-down, and select a **Schema Name**. If not specified, then username will be populated as schema.
6. In the **Properties** section, add **JDBC** properties to establish SQL server connection.
7. Click **Test Connection**. If successful, click **Save**.

Create a MySQL Connection

Learn to register a connection to MySQL in Oracle Data Integration Platform Cloud

1. From the Get Started with Integration section of the Home page or from the Create menu in the Catalog, select **Connection**. You can also create Connections from any Create Task screen.
2. Complete the fields in the **General Information** section. The Identifier is to identify this connection through a string with no spaces, but is not currently used. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
   - For Agent Name, select the agent you're running for MySQL, from the list of available agents.
   - For Type, select **MySQL**.
3. In the **Connection Settings**, complete the following fields:
   - **Hostname:** Enter the DNS name/IP.
   - **Port:**Enter the port number on the host specified for accessing the database.
   - **Username:**The username to connect to MySQL.
   - **Password:** The database password
   - After entering these details, click the **Database** drop-down, and select a **Database**. If not specified, it will populate the default database.

- Click the **Schema Name** drop-down, and select a**Schema**. If not specified, then username will be populated as schema.
- In the **Properties** section, three default Properties are available. You can add more **JDBC** properties.

  For more information, see the MySQL JDBC driver documentation at https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-reference-configuration-properties.html

4. Click **Test Connection**. If successful, click **Save**.

Create an Autonomous Data Warehouse Cloud (ADWC) Connection

Autonomous Data Warehouse Cloud (ADWC) as a connection type. Oracle ADWC provides an easy-to-use, fully autonomous database that scales elastically and delivers fast query performances.

Before you begin

Before you create an ADWC Connection, do the following:

- Create an ADWC instance. For more information, see Provisioning Autonomous Data Warehouse Cloud.
- Download the client credentials zip file from ADWC console. This zip file will contain cwallet.sso, ewallet.p12, keystore.jks, ojdbc.properties, sqlnet.ora, tnsnames.ora, truststore.jks files.

Create the ADWC Connection

1. In the Getting Started section, click **Create**. You can also create a Connection from any Create Task screen.
2. Complete the fields in the **General Information** section.
   - For **Type**, select **Oracle Autonomous Data Warehouse Cloud**.
   - For **Agent**, select the Agent you've downloaded and registered.

     See Set up an Agent.
3. Complete the **Connection Settings**.
   a. ADWC has a pre-existing user created for Oracle GoldenGate On Premises called ggadmin that DIPC will be using as well. The ggadmin user has been granted the right set of privileges for Oracle GoldenGate On Premises Replicat to work. By default, this user is locked. To unlock the ggadmin user, connect to your Oracle Autonomous Data Warehouse Cloud database as the ADMIN user using any SQL client tool. Run the alter user command to unlock the ggadmin user and set the password for it:

   ```
   alter user ggadmin identified by password account unlock;
   ```

   For **Username**, enter `ggadmin`.

   For **Password**, enter the password for `ggadmin`.

   b. For **Credentials**, select the downloaded client credential zip file.

c. For **Service**, select the required service.
d. For **JDBC Setting**, a URL will automatically populate after selecting the service. You need to update the proxy details in this URL, if needed.
e. For **Schema Name**, select the primary schema name.
f. Click **Test Connection** and then **Save** if successful.

Create an Oracle BI Cloud Connector Connection

Harvest ERP Cloud objects using Oracle BI Cloud Connector to view in Data Integration Platform Cloud's Catalog as Data Entities.

To create an Oracle BI Cloud Connection Connection:

1. Click **Create** in the Connection tile of the Getting Started section of the Home page, or from the Create menu in the Catalog, select **Connection**.
2. In the Connection dialog, complete the General Information fields.
   - For Identifier, enter a name that includes only capital letters, numbers, and underscores (_) with no space, or leave it as is. This auto-generated value is used to identify this Connection through a string with no spaces.
   - For Agent, select the appropriate agent.
   - For Type, select **BI Cloud Connector**.
3. Complete the Connection Settings fields.
   - For Service URL, enter your BI Cloud Connector service URL.
   - For Username, enter the user name associated with your BI Cloud Connector Service URL.
   - For Password, enter the password associated with the user name for your BI Cloud Connector account.
   - For Schema, select from a list of available schemas associated with the Service URL and account information for BI Cloud Connector.
4. Click **Test Connection**. If successful, click **Save**, otherwise, review your Connection details and try again.

Create a PDB Connection

If you want to use a Pluggable Database as a Connection for a Replicate Data Task or a Synchronize Data Task with the Replication advanced option in Data Integration Platform Cloud, you must create Container Database Connection also.

Topics:

- Handle Pluggable Multitenant Databases
- Create an Oracle CDB Connection
- Set up PDB as the Source and Target

Handle Pluggable Multitenant Databases

Oracle Data Integration Platform Cloud enables you to replicate Pluggable Databases (PDBs) with its extensive multitenant architecture.

About Multitenant Databases

The multitenancy option in Oracle Database 12*c* enables you to create a *virtual* database for each schema. Each virtual database behaves like an independent database; but runs on top of a real, physical database, which may be hidden from end users. These virtual databases are called **Containers.** The physical database that houses these containers is in effect, a database of containers and is known as a **Container Database** (CDB). You can pull out (or “unplug”) a container from one CDB and place it (or “plug” it) into another CDB. This is why a container is also known as a **Pluggable Database** (PDB). For all practical purposes, from the perspective of clients, PDBs are just regular databases. With the release of Oracle Database 12*c*, comes a radically new multitenant architecture. With the release of Oracle GoldenGate 12*c*, you now have a tool to replicate PDBs.

Oracle Data Pump with PDBs

You can use Data Pump to migrate all, or portions of, a database from a non-CDB into a PDB, between PDBs within the same or different CDBs, and from a PDB into a non-CDB. In general, using Data Pump with PDBs is identical to using Data Pump with a non-CDB.

In Oracle Database 12c Release 1 (12.1), Data Pump does not support any CDB-wide operations. Data Pump produces the following warning if you’re connected to the root or seed database of a CDB:

```
ORA-39357: WARNING: Oracle Data Pump operations are not typically needed
when connected to the root or seed of a container database.
```

Oracle GoldenGate with PDBs

In most ways, Oracle GoldenGate operates in a multitenant container database the same way it operates in a regular Oracle database.

The following are the special requirements that apply to replication to and from multitenant container databases.

- The different pluggable databases in the multitenant container database can have different character sets. Oracle GoldenGate captures data from any multitenant database with different character sets into one trail file and replicates the data without corruption due to using different character sets.
- Extract must operate in integrated capture mode.
- Extract must connect to the root container (`cdb$root`) as a common user in order to interact with the logmining server. To specify the root container, use the appropriate SQL*Net connect string for the database user that you specify with the `USERID` or `USERIDALIAS` parameter. For example: `C##GGADMIN@FINANCE`.
- To support source CDB 12.2, Extract must specify the trail format as release 12.3.
- The `dbms_goldengate_auth.grant_admin_privilege` package grants the appropriate privileges for capture and apply within a multitenant container

database. This includes the `container` parameter, which must be set to `ALL`, as shown in the following example:

```
dbms_goldengate_auth.grant_admin_privilege('C##GGADMIN',container=>'all')
```

REGISTER EXTRACT Command

```
REGISTER EXTRACT sales DATABASE CONTAINER (sales, finance, hr)
```

```
CONTAINER (container[, ...]) here
```

Applies the registration to a list of one or more pluggable databases (containers) of a multitenant CDB. Specify one or more PDBs as a comma-delimited list within parentheses, for example, `CONTAINER (pdb1, pdb2, pdb3)`. All of the PDBs must exist in the database, and all names must be explicit, not wildcarded.

Create User ID Alias

As mentioned, Extract must connect to the root container (`cdb$root`) as a common user in order to interact with the logmining server.

```
ALTER CREDENTIALSTORE ADD USER CDBUSER@HOST:PORT/CDB_SERVICE_NAME PASSWORD *** ALIAS
GGALIASSRC
```

DBLOGIN Command

```
DBLOGIN USERIDALIAS GGALIASSRC
OR
DBLOGIN CDBUSER@HOST:PORT/CDB_SERVICE_NAME PASSWORD ***
```

Privileges Required for CDB User for GoldenGate

```
CREATE EVALUATION CONTEXT
INSERT ANY TABLE
CREATE RULE
DEQUEUE ANY QUEUE
SELECT ANY TABLE
EXECUTE ANY RULE SET
CREATE TABLE
BECOME USER
CREATE RULE SET
FLASHBACK ANY TABLE
UNLIMITED TABLESPACE
ALTER SESSION
CREATE JOB
SET CONTAINER
CREATE SESSION
```

To assign these privileges, you need to execute the following grant commands while logged in as SYSDBA:

```
exec
dbms_goldengate_auth.grant_admin_privileges('cdbuser','capture',container=>
'all');
grant create session to cdbuser;
grant alter session to cdbuser;
grant set container to cdbuser;
```

Oracle Data Integration Platform Cloud with PDB

For Oracle Data Integration Platform Cloud, only CDB user and User ID Alias should be created using the information above. All others are generated or executed by Oracle Data Integration Platform Cloud.

Create an Oracle CDB Connection

You need an Container Database (CDB) connection in Oracle Data Integration Platform Cloud only if you're using an Oracle PDB as a source for a Synchronize Data or Replicate Data task.

First, create an Oracle CDB connection and then create an Oracle Database Connection and select the CDB connection in the connection settings. If you're using the PDB as a target data source, then skip these instructions and create an Oracle Database Connection.

Create an Oracle Database Connection.

To create a connection to a cloud or on-premises Oracle CDB, you must first have your agent ready and running. When you download your agent, ensure that you download the Oracle 12*c* component with it and follow agent instructions to set it up.

For more information see Set up an Agent. After your agent is running, then:

1. From the Getting Started section of the Home page, click **Create** in the Connection tile or from the Create menu in the Catalog, select **Connection**. You can also create Connections from any Create Task screen.
2. Complete the fields in the General Information section. The Identifier is to identify this connection through a string with no spaces, but is not currently used. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
   - For **Agent**, select the agent for your Oracle CDB from the list of available agents.
   - For **Type**, select **Oracle CDB**

     .
3. In the **Connection Settings** complete the following fields:
   - **Hostname:** The DNS name or IP address of your database. For example, example.com or 192.0.2.1.
   - **Port:** The port number on the host specified for accessing the database.
   - **Username:**

     > **Note:**
     >
     > The Container Database (CDB) username is different from the Pluggable Database (PDB) username. Use the CDB common username which has the prefix C##, such as C##GGSRC.
     >
     > For more information, see Create Users for Oracle Database Connections.
   - **Password:** The password associated with the username.

  - **Service Name:** The service name for your PDB database. There is no SID option here.

4. Click **Test Connection**. If successful, click **Save**.

If you want to use your PDB connection as a source for a Synchronize Data or Replicate Data task, then must create another connection and select the type as Oracle. Then select this CDB Connection in the CDB dropdown menu in the Connection Settings.

See Create an Oracle Database Connection

Set up PDB as the Source and Target

Perform the following steps to set up PDB as a source or target for your Synchronize Data Task. The steps are similar to setting up a classic database as a source, but there are requirements for using the `CDBROOT` as `ggaliassrc`.

To set up PDB as a source or target:

1. Log in to Database Cloud Service 12.1 as `sys`:

   ```
   sql> conn sys/password as sysdba
   ```

2. Enable GoldenGate replication:

   ```
   sql> ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION=TRUE SCOPE=BOTH;
   ```

3. Turn on `ARCHIVELOG` mode:

   a. Check whether archivelog is on:

   ```
   sqlplus> archive log list
   ```

   b. Enable archivelog mode:

   ```
   sqlplus> shutdown immediate
   sqlplus> startup mount
   sqlplus> alter database archivelog;
   sqlplus> alter database open;
   ```

   c. Turn on logging

      i. Check whether logging is enabled:

      ```
      sqlplus> SELECT supplemental_log_data_min, force_logging FROM
      v$database;
      ```

      ii. Enable logging:

      ```
      sqlplus> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
      sqlplus> ALTER DATABASE FORCE LOGGING;
      ```

   iii. Disable logging (for clean up later):

   ```
   sqlplus> ALTER DATABASE DROP SUPPLEMENTAL LOG DATA;
   sqlplus> ALTER DATABASE NO FORCE LOGGING;
   ```

4. For the source CDB, enable Open Transaction:

   ```
   sqlplus > exec DBMS_CAPTURE_ADM.BUILD;
   ```

5. Create the common GoldenGate user with the following grants (logged in to CDB):

   ```
   create user C##GGSRC identified by password;
   grant connect, resource, create  session, select any table, create ANY
   table, CREATE ANY  view,  create ANY procedure, CREATE database link,
   SELECT ANY dictionary, exp_full_database, imp_full_database to C##GGSRC;
   grant read, write on directory DATA_PUMP_DIR to C##GGSRC;
   GRANT CREATE SESSION TO C##GGSRC CONTAINER=ALL;
   exec
   dbms_goldengate_auth.grant_admin_privilege('C##GGSRC',container=>'ALL');
   exec dbms_goldengate_auth.grant_admin_privilege('C##GGSRC','capture');
   GRANT alter session to C##GGSRC;
   GRANT set container to C##GGSRC;
   GRANT dba to C##GGSRC CONTAINER=ALL;
   ```

6. Create a source user in PDB with the following grants (first connect to PDB: sys/*password*#@pdb1 as sysdba):

   ```
   create user dipc_src identified by password;
   grant connect, resource, create  session, select any table, create ANY
   table, CREATE ANY view, create ANY procedure, CREATE database link,
   SELECT ANY dictionary, exp_full_database, imp_full_database to dipc_src;
   grant read, write on directory DATA_PUMP_DIR to dipc_src;
   GRANT UNLIMITED TABLESPACE TO dipc_src;
   grant select on v_$database to dipc_src;
   grant select on v_$transaction to dipc_src;
   grant select on v_$archived_log to dipc_src;
   exec DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('dipc_src');
   ```

7. Create a target user in PDB with the following grants:

   ```
   create user dipc_tgt identified by password;
   grant connect, resource, create session, select any table, create ANY
   table, CREATE ANY view, create ANY procedure, CREATE database link,
   SELECT ANY dictionary, exp_full_database, imp_full_database to dipc_tgt;
   grant read, write on directory DATA_PUMP_DIR to dipc_tgt;
   GRANT UNLIMITED TABLESPACE to dipc_tgt;
   exec DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('dipc_tgt');
   ```

8. Test the login. Log in as source to PDB and create a table, and then push some data into it and commit.

9. Log in to Data Integration Platform Cloud and create Connections for the CDB root user (C##GGuser), PDB source and target schemas.

> **Note:**
>
> The source Connection does not use `C##GGSRC` as the Connection username or schema. It is only for `ggaliassrc`. Use a PDB source user or schema.

a. Create an Oracle CDB Connection with C##GGrootuser as the user and CDB service name.

b. Create an Oracle Database Connection for srcSchema user and PDB service.

c. Create an Oracle Database Connection for tgtSchema user.

10. Create a Synchronize Data Task between the source and target Connections.

> **Note:**
>
> If you’re using PDB as a source connection, you must also have an associated CDB connection to support replication.

11. Run the Job.
12. Log in as source to PDB and perform a few inserts and updates. Every operation is updated on the Jobs page. A snapshot is taken every 30 seconds.

Related Topics

- Oracle GoldenGate Multitenant White Paper
- Oracle Fusion Middleware Release Notes for Oracle GoldenGate

Create an Oracle Database Connection

Learn to register your Oracle Database connection in Oracle Data Integration Platform Cloud.

Perform Prerequisites:

1. Set up your users and schemas on your Oracle data source.

   See Create Users for Oracle Database Connections.

2. Download Data Integration Platform Cloud's Agent Installer bundled with Oracle 11g or Oracle 12c.

   > **Note:**
   >
   > If this agent is not installed on the same machine with Oracle 11g or 12c, then the agent must be on a machine that has access to the data source that you are connecting to.

   See Set up an Agent.

Create a Connection

1. From the Getting Started section of the Home page, click Create from the Connection tile or from the Create menu in the Catalog, select **Connection**. You can also create Connections from any Create Task screen.
2. Complete the fields in the General Information section. The Identifier is to identify this connection through a string with no spaces, but is not currently used. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
    - For Agent Name, select the agent for your Oracle CDB from the list of available agents.
    - For Type, select **Oracle Database**.
3. In the **Connection Settings** complete the following fields:
    - **Hostname:** Enter the DNS name or IP address of your database. For example, example.com or 192.0.2.1.
    - **Port:** Enter the port number on the host specified for accessing the database.

      > **Note:**
      >
      > For Oracle Database Cloud Service, port number 1521 should be open or on your system to enable metadata discovery.

    - **Username:**
        - For Synchronize or Replicate Data task's source connection, enter the Data Integration Platform Cloud admin username you created for the setup tables. For example, `dipc_admin`.
        - For Synchronize or Replicate Data task's target connection, enter the Data Integration Platform Cloud username you created for the target. For example, `dipc_tgt`.
        - For other tasks, select a username to connect to this data source.
    - **Password:** Enter the password associated with the username.
    - **Service:** Choose one of the following options for your Database Cloud Service deployment or Oracle Database:
        - Service Name
        - SID Name
    - **Schema Name:**
        - For Synchronize or Replicate Data task's source connection, enter the schema you created for the source user. For example, `dipc_src`.
        - For Synchronize or Replicate Data task's target connection, enter the schema for the target. For example, `dipc_tgt`.
    - **CDB Connection:** If you want to use a PDB as a source, you must first create a CDB connection to that source, and then select that CDB connection here to support your Synchronize Data or Replicate Data task. If the PDB is used for target, then just leave the CDB Connection field blank.
      See Create a PDB Connection.
4. Click **Test Connection**. If successful, click **Save**.

Create Users for Oracle Database Connections

When both Initial Load and Replication are selected for a Synchronize Data task, you may discover more records are processed during Initial Load than you have in your source Connection. This occurs if you don't have a separate user created for the setup process.

Oracle Data Integration Platform Cloud uses a plugin to capture data from an Oracle source. This plugin creates tables in the data source that's defined in the source connection. When the Initial Load process starts, the tables created by the plugin are copied to the target. If you create some transactions in the source, then those changes will also be copied to the target. As a result, there's a discrepancy with the number of records processed. For example, Initial Load action in the job detail will show the number four, when you only have one row in your table. This issue only happens when run your Synchronize Data task with both Initial Load and Replication selected.

To solve this issue, create a separate user for the plugin setup on the source. For example, call it `dipc_admin`. You can then use this admin as the username in the source connection parameters. After that, Data Integration Platform Cloud plugin creates the setup tables in this user's default schema. The user schema specified in the connection details is used for the Synchronize Data source schema. No separate user is required for the target Connection. First follow the instructions here to set up the users and then set up your connections. The link to connections is at the end of this article.

Set up Users on Oracle Source without PDB

Create schemas and grant privileges. You can use existing schemas, just make sure you grant the users the following privileges:

```
SQL > create user dipc_admin identified by password default tablespace
USERS temporary TABLESPACE temp;
User created.
SQL > create user dipc_src identified by password default tablespace USERS
temporary TABLESPACE temp;
User created.
```

Then, grant privileges to these source users:

```
SQL > grant connect, resource, create session, select any table, create
ANY table, CREATE ANY view, create ANY procedure, CREATE database link,
SELECT ANY dictionary, exp_full_database, imp_full_database to dipc_admin;
Grant succeeded.

SQL > grant connect, resource to dipc_src;
Grant succeeded.

SQL > exec
dbms_goldengate_auth.grant_admin_privilege('DIPC_ADMIN','CAPTURE');
```

Grant DIPC plugin privileges to dipc_admin.

```
SQL > grant select on v_$database to dipc_src;
Grant succeeded
```

Set up User on Oracle Target without PDB

Create schemas and grant privileges. You can use existing schemas, just make sure you grant the users the following privileges:

```
SQL > create user dipc_tgt identified by password default tablespace USERS
temporary TABLESPACE temp;
User created.

SQL > exec dbms_goldengate_auth.grant_admin_privilege('DIPC_TGT','APPLY');
```

Then, grant privileges to the target user:

```
SQL > grant connect, resource, create session, select any table, create
ANY table, CREATE ANY view, create ANY procedure, CREATE database link,
SELECT ANY dictionary, exp_full_database, imp_full_database to dipc_tgt;
Grant succeeded.
```

Now, to assign these users in the connection parameters, see Create an Oracle Database Connection

Set up the On-Premises Source and Target

This section describes prerequisite on-premises source and target database configurations that you need to complete before you run your Synchronize Data Task.

Now that you have your Connections created for your source and target databases, you need to set up both the source and target databases before creating and executing your synchronize task. To read from the database transaction log, you may need to enable certain settings in your database and provide replication users with proper privileges. You’ll perform the steps outlined here for both the source and target.

> **Note:**
>
> Database Cloud Service port (1521 in this case) is not accessible from outside by default.
>
> When you're performing a Test Connection through a remote Agent to a Database Cloud Service instance, you are essentially trying to access a Database Cloud Service port (1521 in this case) from outside, which is not accessible.
>
> For this to work:
>
> - You must create an access rule in the Database Cloud Service instance, for the specific port (1521 in this case).
> - Make sure to mention the SOURCE IP (remote agent host), instead of PUBLIC_INTERNET.

Set up the source database

To capture open transactions in your Synchronize Data Task, your database administrator should run the command:

```
DBMS_CAPTURE_ADM.BUILD
```

This creates and dumps a dictionary build in the archived log so that the GoldenGate Capture process can understand the metadata of tables. You also have to run the following commands:

```
grant select on v_$transaction to dipc_source
grant select on v_$archived_log to dipc_source
```

If you run your Synchronize Data Task with Initial Load and Replication separately, then you have to make sure your data is in sync between your source and target. This means that any transaction that starts before Initial Load that ends during the Initial Load process, or before you start the Replication process, is not replicated from source to target.

1. Enable GoldenGate Replication:

   ```
   ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION=TRUE SCOPE=BOTH;
   ```

2. Enable `ARCHIVELOG` mode:
   1. Check whether `archivelog` is enabled:

      ```
      sqlplus > archive log list
      ```

   2. Enable `archivelog` mode:

      ```
      sqlplus > shutdowm immediate
      sqlplus > startup mount
      ```

```
sqlplus > alter database archivelog;
sqlplus > alter database open;
```

c. Disable `archivelog` mode (for clean up later)

```
sqlplus > shutdown immediate
sqlplus > startup mount
sqlplus > alter database noarchivelog;
sqlplus > alter database open;
```

3. Enable logging:
   a. Check if logging is enabled:

```
sqlplus > SELECT supplemental_log_data_min, force_logging FROM
v$database;
```

   b. Enable logging:

```
sqlplus > ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
sqlplus > ALTER DATABASE FORCE LOGGING;
```

   c. Disable logging (for cleanup later)

```
sqlplus > ALTER DATABASE DROP SUPPLEMENTAL LOG DATA;
sqlplus > ALTER DATABASE NO FORCE LOGGING;
```

Set up the target database

Enable GoldenGate replication:

```
sqlplus > ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION=TRUE SCOPE=BOTH;
```

Configure source and target schemas

1. Create schemas and grant privileges. You can use existing schemas, just make sure you grant privileges:
   a. First, create schemas if you're not using existing ones:

```
sqlplus > create user dipc_src identified by dipc_src;
sqlplus > create user dipc_tgt identified by dipc_tgt;
```

   b. Then, grant privileges:

```
sqlplus > grant connect, resource, create session, select any
table, create ANY table, CREATE ANY view, create ANY procedure,
CREATE database link, SELECT ANY dictionary, exp_full_database,
imp_full_database to dipc_src;
sqlplus > grant connect, resource, create session, select any
table, create ANY table, CREATE ANY view, create ANY procedure,
CREATE database link, SELECT ANY dictionary, exp_full_database,
imp_full_database to dipc_tgt;
```

```
sqlplus > grant read, write on directory DATA_PUMP_DIR to dipc_src;
sqlplus > grant read, write on directory DATA_PUMP_DIR to dipc_tgt;
```

2. Grant GoldenGate privileges to source and target schemas:

```
sqlplus > exec DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('dipc_src');
sqlplus > exec DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE('dipc_tgt');
```

3. Grant privileges to source schema:

```
sqlplus > grant select on v_$database to dipc_src;
sqlplus > grant select on v_$transaction to dipc_src;
sqlplus > grant select on v_$archived_log to dipc_src;
sqlplus > grant select on v_$instance to dipc_src;
```

4. Grant tablespace on USERS. For example:

```
sqlplus > Alter user dipc_src quota 50g on USERS;
sqlplus > Alter user dipc_tgt quota 50g on USERS;
```

> **Note:**
>
> Apply a sufficient tablespace amount to avoid errors in Data Pump. See Insufficient tablespace quota leads to error in Data Pump.

Create an Oracle Object Storage Connection

Create a Connection to Oracle Object Storage.

To create an Oracle Object Storage Connection:

1. Obtain the keys and OCIDs.

   You can follow the steps in this tutorial the RSA key pair in PEM format, public key fingerprint, Tenancy's OCID, and user's OCID: Required Keys and OCIDs.

2. Set up a proxy for the agent in its `agent.properties` file, and then restart the agent:

   For example:

```
agentUseProxy=true
proxyHost=www-proxy.example.com
proxyPort=80
```

3. Create and configure a default OCS Connection.
   - For information on how to create a default OCS Connection, see Create an Oracle Object Storage Classic Connection.
   - For information about how to configure a default OCS Connection, see Set up Default Connections.
4. Click **Create** from the Connection tile in the Getting Started section of the Home page, or select **Connection** from the Create menu in the Catalog.

5. Complete the General Information fields. Be sure to select **Oracle Object Storage** for Type.
6. Complete the Connection Settings fields.
7. Click **Test Connection**. If successful, click **Save**, otherwise review your connection details and try again.

Create an Oracle Object Storage Classic Connection

Learn to create a connection to Oracle Object Storage Classic in Oracle Data Integration Platform Cloud.

If required, you can set proxy details in `agent.properties` to connect to Object Storage Connection.

- Go to dicloud/agent/dipcagent/conf/agent.properties.
- Add proxy details in this format:
  - `agentUseProxy=`
  - `proxyHost=`
  - `proxyPort=`

> **Note:**
>
> The Agent ports should be configured as unique port of the system, especially when there are more than one agent on the same operating system.
> Make sure to set gghome in the path to access ggsci prompt :
>
> ```
> PATH=%12CGGHOME%;%12CGGHOME%\crypto;%PATH%
> ```
>
> For Windows , you must set it as
>
> ```
> PATH=%12CGGHOME%;%12CGGHOME%\crypto;%PATH%
> ```

To create an Oracle Object Storage Classic connection:

After setting the parameters, and successful testing, you can connect to Oracle Object Storage Classic.

1. From the Getting Started section of the Home page, click **Create** from the Connection tile or from the Create menu in the Catalog, select **Connection**. You can also create Connections from any Create Task screen.
2. Complete the fields in the **General Information** section. The Identifier is to identify this connection through a string with no spaces, but is not currently used. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
   - For Agent Name, select the agent you’re running for the Oracle Object Storage Classic from the list of available agents.

- For Type, select **Oracle Object Storage Classic**.

3. In the **Connection Settings** complete the following fields:
   - **Domain**: Enter the domain of your Object Storage Classic container. For example, `Storage-example00001234`
   - **Service URL**: Enter the Service URL of your Object Storage Classic container. For example, `https://example00001234.storage.oraclecloud.com`.
   - **Container**: Enter your Object Storage Classic container name. For example, `DIPC_EXPORT_TEST`.
   - **Username**: Enter the username used to connect to your Object Storage Classic container.
   - **Password**: Enter the password associated with the username used to connect to your Object Storage Classic container.
4. Click **Test Connection**. If successful, click **Save**.

Create an Amazon S3 Connection

Harvest data from your S3 Cloud Storage Service for use with Data Integration Platform Cloud.

To create an S3 Connection:

1. Click **Create** from the Connection tile in the Getting Started section of the Home page, or select **Connection** from the Create menu in the Catalog.
2. In the Connection dialog, complete the General Information fields. Be sure to select **S3** for Type.
3. Complete the Connection Settings fields.

   For information on how to obtain your Access Key ID and Secret Access Key, see Access Keys (Access Key ID and Secret Access Key)

   .
4. Click **Test Connection**. If successful, click **Save**, otherwise review your connection details and try again.

Create a Salesforce Connection

Learn to create a connection to Salesforce in Data Integration Platform Cloud

To create a Salesforce connection:

1. From the Getting Started section of the Home page click **Create** from the Connection tile, or from the Create menu in the Catalog, select **Connection**. You can also create Connections from any Task screen.
2. Complete the fields in the General Information section. The Identifier is to identify this connection through a string with no spaces, but is not currently used. If you want to edit this auto-generated field, then you must only include capital letters, numbers, and underscores (_) with no space.
   - For Agent Name, select the agent you're running for Salesforce from the list of available agents.

- For Type, select **Salesforce**.

3. In the Connection Settings, complete the following fields:
    - Hostname: Salesforce server hostname.
    - Username: The username to connect to Salesforce.
    - Password: The database password.
    - After entering these details, click the Database drop-down, and select the **Salesforce** database name. If not specified, it will populate the default database.
    - In the Properties section, the default Properties are available. You can add more JDBC properties.
4. Click **Test Connection**. If successful, click **Save**.

18
Configure GoldenGate

Depending on your source or target environment setup, you may have to manually configure GoldenGate in order to perform certain tasks in Data Integration Platform Cloud.

Topics:

Set up Your GoldenGate Environment

Data Integration Platform provides four variants of GoldenGate bits, Oracle 12*c*, 11*g*, Big Data, and MySQL. Each variant has a setup file for the environment setup.

When you select Data Integration Platform Cloud Enterprise Edition, you can set up your GoldenGate environment as follows:

> **Note:**
>
> Make sure to set gghome in the path to access ggsci prompt :
>
> `PATH=%12CGGHOME%;%12CGGHOME%\lib12;%12CGGHOME%\crypto;%PATH%`

1. For Oracle 12*c*, enter `# source ~/.ggsetup` in a Terminal window on your VM.

   GG for Oracle 12*c* installation: `/u01/app/oracle/suite/gghome` (Version: 12.3.0.1.2 OGGCORE_12.3.0.1.0_PLATFORMS_171208.0005_FBO)

   GG for Oracle 12*c* data: `/u01/data/domains/jlsData/ggdata`

   Agent logs: `/u01/data/domains/jlsData/dipcagent001/logs`

2. For Oracle 11*g*, enter `# source ~/.ggsetup11g` in a Terminal window on your VM.

   GG for Oracle 11g installation: `/u01/app/oracle/suite/gghome11g` (Version: 12.3.0.1.2 OGGCORE_12.3.0.1.0_PLATFORMS_171208.0005_FBO)

   GG for Oracle 11g data : `/u01/data/domains/jlsData/ggdata/ggora11g`

3. For BigData, enter `# source ~/.ggbigdata` in a Terminal window on your VM.

GG bigData installation: `/u01/app/oracle/suite/ggbigdata` (Version: 12.3.0.1.0 OGGCORE_OGGADP.12.3.0.1.0GA_PLATFORMS_170828.1608)

GG bigData data: `/u01/data/domains/jlsData/ggdata/ggbigdata/`

4. For MySQL, enter `# source ~/.ggmysql` in a Terminal window on your VM.

   GG mysql installation: `/u01/app/oracle/suite/ggmysql` (Version: 12.3.0.1.1 OGGCORE_12.3.0.1.0_PLATFORMS_170804.2007)

   GG mysql data: `/u01/data/domains/jlsData/ggdata/ggmysqlData/`

5. Locate the logs:

   LCM logs: (logs in `/tmp/` are lost when rebooted or patched)

   ```
   /tmp/dics_install.log /tmp/idcs_patching.log
   /u01/app/oracle/suite/dics/logs/dics_domain.log
   /u01/app/oracle/suite/dics/logs/dics_rcu.log
   ```

   SOLR logs: `/u01/data/domains/jlsData/solr/logs/`

   GoldenGate logs:

   ```
   /u01/data/domains/jlsData/ggdata/ggserr.log
   /u01/data/domains/jlsData/ggdata/ggora11g/ggserr.log
   /u01/data/domains/jlsData/ggdata/ggbigdata/ggserr.log
   /u01/data/domains/jlsData/ggdata/ggmysqlData/ggserr.log
   ```

   Patch and rollback logs: `/tmp/upgrade.log /tmp/rollback.log`

Configure GoldenGate 11*g* for Database Cloud Service

Once the basic connectivity is established by the agent to GoldenGate, the following procedure can be used to configure the agent to work with 11*g* GoldenGateHome:.

1. Stop the agent if it's already running: `ps -ef | grep mgr`.
2. Stop any manager running in the system. This takes care of port conflicts, if there are any.
3. To stop the manager, launch the GGSCI console: `# {GGHOME}/ggsci`.
4. Stop the manager.
5. Set `agentGoldenGateHome=${gghome11g}` in the agent.properties file located under `/u01/data/domains/jlsData/${agent_instance_home}/conf`
6. Start the agent.

Configure GoldenGate 12*c* for Database Cloud Service

Before you can use an Oracle Database Cloud Service database deployment as a replication target in Oracle GoldenGate Cloud Service, you must configure its database as a valid replication database.

You can configure the database during database deployment creation by selecting **Enable Oracle GoldenGate** on the Service Details page of the provisioning wizard. If you don’t, you can configure it manually after the database deployment is created by using the dbaascli utility.

To configure the database manually after the database deployment is created:

1. Connect as the **oracle** user to the database deployment’s compute node.

   For detailed instructions, see Connecting to a Compute Node Through Secure Shell (SSH) in *Administering Oracle Database Cloud Service*.

2. Confirm that the database is not yet configured as a valid replication database:

   ```
   $ dbaascli gg status
   DBAAS CLI version 1.0.0
   Executing command gg status

   Golden Gate status: disabled.
   ```

   If the status is listed as disabled, you need to configure the database; if it is listed as enabled, you do not.

3. Configure the database as a valid replication database by using the dbaascli gg setup command:

   ```
   $ dbaascli gg setup
   DBAAS CLI version 1.0.0
   Executing command gg setup
   Enter Golden Gate admin username: admin-username
   Enter Golden Gate admin password: admin-password
   Re-enter Golden Gate admin password: admin-password
   Setting up Golden Gate
   Updating the registry
   Successfully setup GG
   ```

   Where:

   - *admin-username* is the database user name for Oracle GoldenGate Cloud Service access to the database:
     – For Oracle Database 11g, specify **ggadmin**.
     – For Oracle Database 12c, specify **c##ggadmin**.
   - *admin-password* is the password to use for the database user. You can use the administrator password provided when the database deployment was created, or you can use a different password that conforms to password requirements for Oracle Database users.

4. Close your connection to the compute node.

Configure GoldenGate Manually to Work between On-Premises and Cloud

This chapter includes the following sections:

- Configure Extracts and Data Pumps
- Configure Replication

Configure Extracts and Data Pumps

Oracle Data Integration Platform Cloud replication requires you to configure Extract and data pump process on source.

1. To connect to the replication node, use one of the following options:
   - Local trail (`ExtTrail`) on the local system
   - Remote trail (`RmtTrail`) on the remote system

   > **Note:**
   >
   > Oracle Data Integration Platform Cloud trails support the continuous extraction and replication of database (on-premises or cloud) changes, storing these changes temporarily on cloud. A trail can reside on any platform that Oracle Data Integration Platform Cloud supports. (Oracle, MySQL, and Big Data databases are supported.)
   >
   > You can configure one Replication node to process a trail for target databases. After all the data has been consumed, Replicat can then purge the data using the `MinKeepDays` parameter. As long as Replicat remains current, your temporary storage requirements for trails can be low.

2. Format the Trail:

   •By default, trails are formatted in canonical format, allowing them to be exchanged rapidly and accurately among databases.

   Each trail file contains the following:

   - ***Record header area***: Stored at the beginning of the file and contains information about the trail file itself.

     Trail File Information

     – Compatibility level

     – Character set (globalization function with version 11.2.1 and later)

     – Creation time

     – File sequence number

     – File size

     First and Last Record Information

     – Timestamp

     – Commit Sequence Number (CSN)

     Extract Information

     – Oracle Data Integration Platform Cloud version

     – Group name

     – Host name and Hardware type

     – Operating system type and version

  - DB type, version, and character set
- ***Record data area***: Contains a header area as well as a data area.
- ***Checkpoints:*** Both Extract and Replicat maintain checkpoints into the trails. Checkpoints provide persistent processing whenever a failure occurs. Each process resumes where the last checkpoint was saved, guaranteeing that no data is lost. One Extract can write to one or many trails. One or many Replicat processes are involved in processing each trail.

> **Note:**
>
> Instead of the default canonical format, you can use alternative formats to output data.
>
> This feature is beneficial if database load utilities or other programs are used that require different input format.
>
> These alternative formats include:
>
> - Logical Change Records (LCRs)
> - `FormatASCII`
> - `FormatSQL`
> - `FormatXML`

3. Set Up a View

Objective	Command
To view the trail file header:	`Logdump 1> fileheader on`
To view the record header with the data:	`Logdump 2> ghdr on`
To add column information:	`Logdump 3> detail on`
To add hexadecimal and ASCII data values to the column list:	`Logdump 4> detail data`
To control how much record data is displayed:	`Logdump 5> reclen 280`

4. Keep a log of your session

Objective	Command
To start and stop the logging of a logdump session, use the **Log** option:	`Logdump> Log to MySession.txt`
When enabled, logging remains in effect for all sessions of Logdump until it's disabled with the `Log Stop`command:	`Logdump> Log Stop`

Supported Scenarios

This table describes different scenarios considering that integrated extract and integrated delivery are not supported on any of the non-Oracle databases.

Source	Target	Extract	Replicat
Oracle 11.x	Oracle Database Cloud Service	Integrated Extract is supported	Integrated and Coordinated Delivery supported
Oracle 12.1	Oracle Database Cloud Service	Integrated Extract is supported	Integrated and Coordinated Delivery supported
MySQL	Oracle Database Cloud Service	Only Classic Extract is supported	Integrated and Coordinated Delivery supported

> **Note:**
>
> With Oracle 12.1, when not using multitenancy, you can still use Classic Extract, however, it can’t be used when container/pluggable databases are used.

You can review detailed steps in the tutorial, Replicate On-Premises Data to Cloud with Oracle GoldenGate Cloud Service.

Configure Replication

Oracle Data Integration Platform Cloud replication requires connecting to and configuring database support for the replication node.

Configure a General Replication Process

1. **Connect to the node defined in the Manager parameter file mgr.prm** located at /u01/app/oracle/gghome/dirprm
2. Avoid using root user to run Data Integration Platform Cloud processes, otherwise some operations can fail to read using 'oracle' user.

In the following table, you can review the parameters and descriptions necessary to configure a replication process.

Parameter	Description
`Port:`	Establishes the TCP/IP Port Number on Which Manager Listens For Requests
`DynamicPortList:`	Specifies the ports that Manager can dynamically allocate
`Autostart:`	Specifies the processes to be restarted after abnormal termination
`LagReportHours:`	Sets the interval, in hours, at which Manager checks the lag for Extract and Replicat processing. Alternatively, this interval can be set in minutes.
`LagInfoMinutes:`	Specifies the interval at which Extract and Replicat send an informational message to the event log. Alternatively, this interval can be set in seconds or hours.
`LagCriticalMinutes:`	Specifies the interval at which Extract and Replicat send a critical message to the event log. Alternatively, this interval can be set in seconds or hours.
`PurgeOldExtracts:`	Purges the Data Integration Platform Cloud trails that are no longer needed, based on option settings.

Note:

If you copy and paste text into parameter files, then beware of editors that try to turn a double-minus into em—dashes.

Managing Trail Files Use the `PurgeOldExtracts` parameter in the Manager parameter file to purge trail files when Data Integration Platform Cloud has finished processing them.

Note:

Trail files, if not managed properly, can consume a significant amount of disk space!

Configure Replication process, using the DMZ (Demilitarized Zone) Server

A DMZ server is a public-facing computer host placed on a separate or isolated network segment. The intention of this server is to provide an addition layer of network security between servers in the trusted network and servers in the public network.

Follow the four high-level steps to configuring a replication from a non-cloud database to cloud.

1. Start the SSH Proxy Server on the DMZ Server.
2. Configure and start the Online Change Capture Process (Extract) on the on-premise server.
3. Configure and start the Data pump Extract on the on-premise Server (SOCKS PROXY pointing to DMZ Server).
4. Configure and start the Online Change Delivery Process (Replicat) on the GGCS server.

1. Start the SSH Proxy Tunnel Setup on the DMZ Server

- Start the SSH SOCKS Proxy Server on the DMZ Server

  Command Syntax:

  ```
  ssh –i <private_key file> -v –N –f –D <listening IP
  Address>:<listening IP port><GGCS Oracle User>@<GGCS IP
  Address>><socksproxy output file>2>&1
  ```

Parameter	Description
–i	Private Key file.
-v	Verbose Mode.
–N	Don’t execute remote command (used for port forwarding).
–f	Run SSH process in the background.
–D	Specifies to run as local dynamic application level forwarding; act as a SOCKS proxy server on a specified interface and port.
2>&1	Redirect Stdout and Stderr to the output filelistening.

Parameter	Description
`IP Address`	DMZ Server IP Address.
`listening IP port`	TCP/IP Port Number.

- Verify that the SSH SOCKS Proxy server has started successfully.

  Check the socks proxy output file using the `cat` Unix command.

  Look for the:

  `Local connections to <dmz-server:port>` and

  `Local forwarding listening on <ip_address> port <port #>`.

  This information helps you to make sure you're pointing to the right DMZ server address and port.

2. Configure and Start the Online Change Capture Process (Extract) on the On-premises Server

On the source server, create the online change capture (Extract) process, using the following commands:

```
GGCSI> add extract etpcadb, tranlog, begin now
GGSCI> add exttrail ./dirdat/ea, extract etpcadb, megabytes 100
GGSCI> start extract etpcadb
GGSCI> info extract etpcadb detail
```

3. Configure and Start the Data pump Extract on the On-premise Server

On the source server, create the datapump (Extract) process, using the following commands:

```
GGCSI> add extract ptpcadb, exttrailsource ./dirdat/ea
GGSCI> add rmttrail ./dirdat/pa, extract ptpcadb, megabytes 100
GGSCI> start extract ptpcadb
GGSCI> info extract ptpcadb detail
```

4. Configure and Start the Online Change Delivery Process (Replicat) on the Cloud Server.

On the Data Integration Platform Cloud server, create the Change Delivery process (Replicat) using the following commands:

```
GGCSI> dblogin useridalias dipcuser_alias
GGSCI> add replicat rtpcadb integrated, exttrail ./dirdat/pa
GGSCI> start replicat rtpcadb
GGSCI> info replicat rtpcadb detail
```

You can review this detailed steps by following the tutorial Replicate On-Premises Data to Cloud with Oracle GoldenGate Cloud Service.

Configure GoldenGate for MySQL Server Targets

This chapter provides instructions for preparing your system for running Oracle GoldenGate. It contains the following sections:

- Ensure Data Availability
- Set Logging Parameters
- Add Host Names
- Set the Session Character Set
- Configure Bi-directional Replication
- Other Oracle GoldenGate Parameters for MySQL
- Prepare Tables for Processing
- Position Extract to a Specific Start Point
- Change the Log-Bin Location

Ensure Data Availability

Retain enough binary log data so that if you stop Extract or there is an unplanned outage, Extract can start again from its checkpoints. Extract must have access to the binary log that contains the start of the oldest uncommitted unit of work, and all binary logs thereafter. The recommended retention period is at least 24 hours worth of transaction data, including both active and archived information. You might need to do some testing to determine the best retention time given your data volume and business requirements.

If data that Extract needs during processing was not retained, either in active or backup logs, one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which binary log data is available (and accept possible data loss on the target).
- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

To determine where the Extract checkpoints are, use the `INFO EXTRACT` command. For more information, see GGSCI Command Interface in *Reference for Oracle GoldenGate*.

Set Logging Parameters

To capture from the MySQL transaction logs, the Oracle GoldenGate Extract process must be able to find the index file. index file in turn contains the paths of all binary log files.

> **Note:**
>
> Extract expects that all of the table columns are in the binary log. As a result, only `binlog_row_image` set as `full` is supported and this is the default. Other values of `binlog_row_image` are not supported.

Extract checks the following parameter settings to get this index file path:

1. Extract `TRANLOGOPTIONS` parameter with the `ALTLOGDEST` option: If this parameter specifies a location for the log index file, Extract accepts this location over any default that is specified in the MySQL Server configuration file. When `ALTLOGDEST` is used, the binary log index file must also be stored in the specified directory. This parameter should be used if the MySQL configuration file does not specify the full index file path name, specifies an incorrect location, or if there are multiple installations of MySQL on the same machine

   To specify the index file path with `TRANLOGICOPTIONS` with `ALTLOGDEST`, use the following command format on Windows:

   ```
   TRANLOGOPTIONS ALTLOGDEST "C:\\Program Files\\MySQL\\logs\\binlog.index"
   ```

   On Linux, use this format:

   ```
   TRANLOGOPTIONS ALTLOGDEST "/mnt/rdbms/mysql/data/logs/binlog.index"
   ```

2. The MySQL Server configuration file: The configuration file stores default startup options for the MySQL server and clients. On Windows, the name of the configuration file is `my.ini`. On other platforms, it is `my.cnf`. In the absence of `TRANLOGOPTIONS` with `ALTLOGDEST`, Extract gets information about the location of the log files from the configuration file; however, even with `ALTLOGDEST`, these Extract parameters must be set correctly:

   - `binlog-ignore-db=oggddl`: This prevents DDL logging history table entries in the `binlog` and is set in the `my.cnf` or `my.ini` file.
   - `log-bin`: This parameter is used to enable binary logging. This parameter also specifies the location of the binary log index file and is a required parameter for Oracle GoldenGate, even if `ALTLOGDEST` is used. If log-bin is not specified, binary logging will be disabled and Extract returns an error.
   - `log-bin-index`: This parameter specifies the location of the binary log index. If it is not used, Extract assumes that the index file is in the same location as the log files. If this parameter is used and specifies a different directory from the one that contains the binary logs, the binary logs must not be moved once Extract is started.
   - `max_binlog_size`: This parameter specifies the size, in bytes, of the binary log file.

   > **Note:**
   >
   > The server creates a new binary log file automatically when the size of the current log reaches the `max_binlog_size` value, unless it must finish recording a transaction before rolling over to a new file.

- binlog_format: This parameter sets the format of the logs. It must be set to the value of ROW, which directs the database to log DML statements in binary format. Any other log format (MIXED or STATEMENT) causes Extract to abend.

> **Note:**
>
> MySQL binary logging does not allow logging to be enabled or disabled for specific tables. It applies globally to all tables in the database.

To locate the configuration file, Extract checks the MYSQL_HOME environment variable: If MYSQL_HOME is set, Extract uses the configuration file in the specified directory. If MYSQL_HOME is not set, Extract queries the information_schema.global_variables table to determine the MySQL installation directory. If a configuration file exists in that directory, Extract uses it.

Add Host Names

Oracle GoldenGate gets the name of the database it is supposed to connect to from the SOURCEDB parameter. A successful connection depends on the localhost entry being properly configured in the system host file. To avoid issues that arise from improper local host configuration, you can use SOURCEDB in the following format:

```
SOURCEDB database_name@host_name
```

Where: *database_name* is the name of the MySQL instance, and *host_name* is the name or IP address of the local host. If using an unqualified host name, that name must be properly configured in the DNS database. Otherwise, use the fully qualified host name, for example myhost.company.com.

Set the Session Character Set

The GGSCI, Extract and Replicat processes use a session character set when connecting to the database. For MySQL, the session character set is taken from the SESSIONCHARSET option of SOURCEDB and TARGETDB. Make certain you specify a session character set in one of these ways when you configure Oracle GoldenGate.

Configure Bi-directional Replication

In a bi-directional configuration, there are Extract and Replicat processes on both the source and target systems to support the replication of transactional changes on each system to the other system. To support this configuration, each Extract must be able to filter the transactions applied by the local Replicat, so that they are not recaptured and sent back to their source in a continuous loop. Additionally, AUTO_INCREMENT columns must be set so that there is no conflict between the values on each system.

1. Configure Oracle GoldenGate for high availability or active-active replication according to the instructions in the Overview of Replicat in *Administering Oracle GoldenGate*.
2. To filter out Replicat operations in a bi-directional configuration so that the applied operations are not captured and looped back to the source again, take the following steps on each MySQL database:

- Configure each Replicat process to use a checkpoint table. Replicat writes a checkpoint to this table at the end of each transaction. You can use one global checkpoint table or one per Replicat process See Overview of Replicat in *Administering Oracle GoldenGate*.
- Specify the name of the checkpoint table with the `FILTERTABLE` option of the `TRANLOGOPTIONS` parameter in the Extract parameter file. The Extract process will ignore transactions that end with an operation to the specified table, which should only be those of Replicat.

> **Note:**
>
> Although optional for other supported databases as a means of enhancing recovery, the use of a checkpoint table is required for MySQL when using bi-directional replication (and likewise, will enhance recovery).

3. Edit the MySQL server configuration file to set the `auto_increment_increment` and `auto_increment_offset` parameters to avoid discrepancies that could be caused by the bi-directional operations. The following illustrates these parameters, assuming two servers: **ServerA** and **ServerB**.

   ServerA:

   ```
   auto-increment-increment = 2
   auto-increment-offset = 1
   ```

   ServerB:

   ```
   auto-increment-increment = 2
   auto-increment-offset = 2
   ```

Other Oracle GoldenGate Parameters for MySQL

The following parameters may be of use in MySQL installations, and might be required if non-default settings are used for the MySQL database. Other Oracle GoldenGate parameters will be required in addition to these, depending on your intended business use and configuration.

Table 18-1 Other Parameters for Oracle GoldenGate for MySQL

Parameter	Description
`DBOPTIONS` with `CONNECTIONPORT` *`port_number`*	Required to specify to the VAM the TCP/IP connection port number of the MySQL instance to which an Oracle GoldenGate process must connect if MySQL is not running on the default of 3306. `DBOPTIONS CONNECTIONPORT 3307`
`DBOPTIONS` with `HOST` *`host_id`*	Specifies the DNS name or IP address of the system hosting MySQL to which Replicat must connect.
`DBOPTIONS` with `ALLOWLOBDATATRUNCATE`	Prevents Replicat from abending when replicated LOB data is too large for a target MySQL `CHAR`, `VARCHAR`, `BINARY` or `VARBINARY` column.

Table 18-1 (Cont.) Other Parameters for Oracle GoldenGate for MySQL

Parameter	Description

`SOURCEDB` with `USERID` and `PASSWORD`	Specifies database connection information consisting of the database, user name and password to use by an Oracle GoldenGate process that connects to a MySQL database. If MySQL is not running on the default port of 3306, you must specify a complete connection string that includes the port number: `SOURCEDB` *`dbname@hostname:port`*, `USERID` *`user`*, `PASSWORD` *`password`*.Example:
	`SOURCEDB mydb@mymachine:3307, USERID myuser, PASSWORD mypassword`
	If you are not running the MySQL database on port 3306, you must also specify the connection port of the MySQL database in the `DBLOGIN` command when issuing commands that affect the database through GGSCI:
	`DBLOGIN SOURCEDB` *`dbname@hostname`*`:`*`port`*`, USERID` *`user`*`, PASSWORD` *`password`*
	For example:
	`GGSCI> DBLOGIN SOURCEDB mydb@mymachine:3307, USERID myuser, PASSWORD mypassword`

`SQLEXEC`	To enable Replicat to bypass the MySQL connection timeout, configure the following command in a `SQLEXEC` statement in the Replicat parameter file. `SQLEXEC "select CURRENT_TIME();" EVERY` *`n`* `MINUTES` **Where**: *n* is the maximum interval after which you want Replicat to reconnect. The recommended connection timeout 31536000 seconds (365 days).

See Oracle GoldenGate Parameters in *Reference for Oracle GoldenGate*.

See Introduction to Oracle GoldenGate in *Administering Oracle GoldenGate*.

Prepare Tables for Processing

This section describes how to prepare the tables for processing. Table preparation requires these tasks:

Position Extract to a Specific Start Point

You can position the `ADD EXTRACT` and `ALTER EXTRACT` commands to a specific start point in the transaction logs with the following command.

```
{ADD | ALTER EXTRACT} group, VAM, LOGNUM log_num, LOGPOS log_pos
```

- *`group`* is the name of the Oracle GoldenGate Extract group for which the start position is required.
- *`log_num`* is the log file number. For example, if the required log file name is `test.000034`, this value is 34. Extract will search for this log file.

- *log_pos* is an event offset value within the log file that identifies a specific transaction record. Event offset values are stored in the header section of a log record. To position at the beginning of a binlog file, set the log_pos as 4. The log_pos 0 or 1 are not valid offsets to start reading and processing.

In MySQL logs, an event offset value can be unique only within a given binary file. The combination of the position value and a log number will uniquely identify a transaction record and cannot exceed a length of 37. Transactional records available after this position within the specified log will be captured by Extract. In addition, you can position an Extract using a timestamp.

Change the Log-Bin Location

Modifying the binary log location by using the log-bin variable in the MySQL configuration file might result in two different path entries inside the index file, which could result in errors. To avoid any potential errors, change the log-bin location by doing the following:

1. Stop any new DML operations.
2. Let the extract finish processing all of the existing binary logs. You can verify this by noting when the checkpoint position reaches the offset of the last log.
3. After Extract finishes processing the data, stop the Extract group and, if necessary, back up the binary logs.
4. Stop the MySQL database.
5. Modify the log-bin path for the new location.
6. Start the MySQL database.
7. To clean the old log name entries from index file, use flush master or reset master (based on your MySQL version).
8. Start Extract.

Configure GoldenGate for Big Data Targets

This procedure helps you to configure and run Oracle GoldenGate for Big Data to extend the capabilities of Oracle GoldenGate instances. Oracle GoldenGate for Big Data supports specific Big Data handler configurations. It details how each of the Oracle GoldenGate for Big Data Handlers are compatible with the various data collections including distributions, database releases, and drivers.

See the following table for more information.

Topic	Description
Introduction to GoldenGate for Big Data	This chapter provides an introduction to Oracle GoldenGate for Big Data concepts and features. It includes how to verify and set up the environment, use it with Replicat, logging data, and other configuration details.
Using the Cassandra Handler	This chapter explains the Cassandra Handler and includes examples so that you can understand this functionality.

Topic	Description
Using the Elasticsearch Handler	This chapter explains the Elasticsearch Handler and includes examples so that you can understand this functionality.
Using the Flume Handler	This chapter explains the Flume Handler and includes examples so that you can understand this functionality.
Using the HBase Handler	The HBase Handler allows you to populate HBase tables from existing Oracle GoldenGate supported sources.
Using the HDFS Handler	This chapter explains the HDFS Handler, which is designed to stream change capture data into the Hadoop Distributed File System (HDFS).
Using the Java Database Connectivity Handler	The Generic Java Database Connectivity (JDBC) is a handler that lets you replicate source transactional data to a target system or database. This chapter explains the Java Database Connectivity (JDBC) Handler and includes examples so that you can understand this functionality.
Using the Kafka Handler	This chapter explains the Kafka Handler and includes examples so that you can understand this functionality.
Using the Kafka Connect Handler	This chapter explains the Kafka Connect Handler and includes examples so that you can understand this functionality.
Using the Kinesis Streams Handler	This chapter explains the Kinesis Streams Handler and includes examples so that you can understand this functionality.
Using the MongoDB Handler	This chapter explains the MongoDB Handler and includes examples so that you can understand this functionality.
Using the Metadata Provider	This chapter explains the Metadata Provider functionality, different types of Metadata Providers, and examples that can be used to understand the functionality.
Using the Pluggable Formatters	Formatters provide the functionality to convert operations from the Oracle GoldenGate trail file info formatted messages that can then be sent to Big Data targets by one of the Oracle GoldenGate for Big Data Handlers.
Cassandra Handler Client Dependencies	This appendix lists the Cassandra client dependencies for Apache Cassandra.
Elasticsearch Handler Client Dependencies	This appendix lists the Elasticsearch transport client dependencies.
Flume Handler Client Dependencies	This appendix lists the Flume client dependencies for Apache Flume.
HBase Handler Client Dependencies	This appendix lists the HBase client dependencies for Apache HBase. The hbase-client-x.x.x.jar is not distributed with Apache HBase nor is it mandatory to be in the classpath. The hbase-client-x.x.x.jar is an empty maven project with the purpose of aggregating all of the HBase client dependencies.

Topic	Description
HDFS Handler Client Dependencies	This appendix lists the HDFS client dependencies for Apache Hadoop. The hadoop-client-x.x.x.jar is not distributed with Apache Hadoop nor is it mandatory to be in the classpath. The hadoop-client-x.x.x.jar is an empty maven project with the purpose of aggregating all of the Hadoop client dependencies.
Kafka Handler Client Dependencies	This appendix lists the Kafka client dependencies for Apache Kafka.
Kafka Connect Handler Client Dependencies	This appendix lists the Kafka Connect client dependencies for Apache Kafka.
MongoDB Handler Client Dependencies	Oracle GoldenGate recommends that you use the 3.2.2 MongoDB Java Driver integration with MongoDB using mongo-java-driver-3.2.2.jar. You can download this driver from: http://mongodb.github.io/mongo-java-driver/
Understanding the Java Adapter and Oracle GoldenGate for Big Data	The Oracle GoldenGate Java Adapter is the overall framework. It allows you to implement custom code to handle Oracle GoldenGate trail records, according to their specific requirements. It comes built-in with Oracle GoldenGate File Writer module that can be used for flat file integration purposes.

Part V
Appendix

Topics

A Terminology

Your Data Integration Platform Cloud service uses specific terms that are related to the product use.

Term	Description
Agent	Allows Data Integration Platform Cloud to exchange heart beats with source and target resources.
Connection	Describes a physical connection to a database. It includes all the details necessary to establish a connection to a resource.
Connection Type	Defines the set of properties for a connection.
Data Asset	A repository, such as a Data Lake.
Data Entity	A collection of data such as table, file, dimension, or view.
Execution Environment	
Job	An execution of a Task.
Notification	System messages alerting you to specific job conditions.
Policy	A rule that specifies the condition(s) for which an alert notification is produced.
Schema	An owner of data entities. A schema has a default Connection to provide a default setting used in operations like generate or execute for tasks.
Task	Different types of data integration activities you can perform in Data Integration Platform Cloud.

B

What's Certified for Hosted VM Tasks

This topic only applies to Data Integration Platform Cloud Classic.

Oracle Data Integration Platform Cloud has four applications in its VM: Oracle Data Integrator (ODI), Oracle GoldenGate (OGG), Enterprise Data Quality (EDQ) and Oracle Stream Analytics (OSA). You can access these applications to create tailored data integration solutions.

Topics

See what’s certified when you access the VM of Oracle Data Integration Platform Cloud to create your own customized tasks:

> **Note:**
>
> - You can only access the VM of **user-managed** Oracle Data Integration Platform Cloud instances.
> - Oracle Data Integration Platform Cloud offers an EDQ console for instances with the Governance edition. Therefore, the EDQ certification matrix applies to user-managed and Oracle-managed Data Integration Platform Cloud instances. The certifications for the other three applications here only apply to user-managed instances.

- Browser Certifications
- Java Version Certifications
- Oracle GoldenGate (OGG) Certifications
- Data Integrator (ODI) Certifications
- Enterprise Data Quality (EDQ) Certifications
- Stream Analytics (OSA) Certifications

Browser Certifications

Use this certification matrix to find out which web browsers Oracle Data Integration Platform Cloud is certified for.

Data Integration Platform Cloud supports the following browsers:

Browser	Versions
Microsoft Internet Explorer	11.x+
Mozilla Firefox	ESR 38+

Browser	Versions
Google Chome	42+
Apple Safari	7.x, 8.x
Microsoft Edge	For Windows 10 (without Java)

Java Version Certifications

Oracle Data Integration Platform Cloud is certified for Java 1.8 and above.

Oracle GoldenGate (OGG) Certifications

Here's a list of data sources, operating systems and platforms that you can use with the Oracle GoldenGate application that's accessible in the user-managed version of Oracle Data Integration Platform Cloud.

> **Note:**
>
> - The installed Oracle GoldenGate application version is 12.3.
> - All data sources must have x86_64, the 64 bit version of x86 operating systems, with the latest upgrade.
> - Some of the data source connections have to be configured manually with additional steps.
> - For Exadata, Exadata Cloud Machine and Exadata Cloud Service, only remote capture with remote delivery is supported.

Table B-1 Oracle GoldenGate Certifications for Oracle Data Integration Platform Cloud

Data Source	Data Source Version	Data Source OS version	Source	Target	Remote Agent Location
Oracle Database Cloud Classic	11.2, 12.1, 12.2	OEL 6.x	yes	yes	OCI, OCI-Classic, on-premises
Oracle Database	11.2, 12.1, 12.2	OEL 6.x, RHEL 6.x, 7.x, SLES 11, 12	yes	yes	OCI, OCI-Classic, on-premises
Oracle Database	11.2, 12.1, 12.2	Windows 2012, 2016	yes	yes	on-premises
Exadata and Exadata Cloud Machine	11.2, 12.1, 12.2	n/a	yes	yes	OCI, OCI-Classic, on-premises
Exadata Service Cloud	11.2, 12.1, 12.2	n/a	yes	yes	OCI, OCI-Classic, on-premises
Oracle MySQL Cloud Service	MySQL 5.7	OEL 6.x	yes	yes	OCI-Classic

Table B-1 (Cont.) Oracle GoldenGate Certifications for Oracle Data Integration Platform Cloud

Data Source	Data Source Version	Data Source OS version	Source	Target	Remote Agent Location
SQL Server (source only)	2008, 2008 R2, 2012, 2014, 2016 Enterprise Editions 2016 SP1 (or greater) Standard Edition	Windows 2012, 2016	yes	no	on-premises
SQL Server (target only)	2008, 2008 R2, 2012, 2014, 2016 Standard and Enterprise Editions	Windows 2012, 2016	no	yes	on-premises
Big Data: HDFS/ Hive, Flume, Hbase, Kafka Pub/Sub, Kafka REST Proxy, MongoDB, Cassandra, ElasticSearch, AWS Kinesis, AWS S3, Azure Data Lake, Flat File, JMS	See **GoldenGate for Big Data** tab in Oracle GoldenGate Certification Matrix for 12.3.	OEL 6.x, RHEL 6.x, 7.x, SLES 11, 12, Windows 2012, 2016	yes for JMS, Cassandra	yes for all	OCI, OCI-Classic, on-premises
Autonomous Data Warehouse Cloud (ADWC)	n/a	OEL 6.x	no	yes	OCI, OCI-Classic, on-premises

Manual Configuration Details

For Oracle database manual configuration steps, see:

- Set up the On-Premises Source and Target
- Set up PDB as the Source and Target

Data Integrator (ODI) Certifications

Here’s a list of data sources, operating systems and platforms that you can use as source and targets for the Oracle Data Integrator application that’s available in the user-managed version of Oracle Data Integration Platform Cloud. The ODI version is 12.2.1.3.1

Refer to the information in the **ODI Source— Target** tab of the Oracle Fusion Middleware 12c (12.2.1.3.0) Certification Matrix.

Enterprise Data Quality (EDQ) Certifications

Here’s a list of data sources and their versions that you can use as data stores in the EDQ console available for Oracle Data Integration Platform Cloud. EDQ is only available in the Governance edition. The EDQ version is 12.2.1.3.1.

Refer to the information in the **EDQ Data Store** tab in the Oracle Fusion Middleware 12c (12.2.1.3.0) Certification Matrix for a list of certified data stores.

> **Note:**
>
> To access EDQ applications, you must use Oracle Identity Cloud Service and link to a certified **LDAP system** mentioned in the **ID & Access** tab of the Oracle Fusion Middleware 12c (12.2.1.3.0) Certification Matrix.

EDQ supports browsers the following browsers that are capable of running Java Web-Start:

- Mozilla Firefox, version ESR 38+ (latest is 52)
- Google Chrome version 42+ (latest is 58)
- Apple Safari version 7.x and 8.x
- Microsoft Edge for Windows version 10 (without Java)

Data Integration Platform Cloud server supports Java version 1.8+ ; The latest is update 141. And EDQ supports Java version 1.8.77+.

Stream Analytics (OSA) Certifications

Data Integration Platform Cloud (user-managed) certifies the following products for OSA.

OSA supports Java Development Kit version 8 update 131+.

OSA supports Spark version 2.2.1 with Hadoop 2.7 or higher.

OSA supports Google Chrome version 60+.

Refer to the information in the Oracle Fusion Middleware 12c Certification Matrix for more information

C

Frequently Asked Questions for Data Integration Platform

Get answers to frequently asked questions for Oracle Data Integration Platform Cloud.

General Questions about Oracle Data Integration Platform Cloud

What is Oracle Data Integration Platform Cloud?

Data Integration Platform Cloud (DIPC) is an Oracle Cloud service that offers a single platform to connect to hundreds of on-premises and cloud data sources. From this platform you extract, load and transform (ELT) data entities of any shape and format, synchronize or replicate selected data sources, integrate with big data technologies, perform data analytics and maintain data quality.

What use cases does Data Integration Platform Cloud solve?

The most relevant use cases are:

- Data warehouses, data integration and migrations (Standard edition)
- Big Data integration, data synchronization, zero downtime migration, real-time data warehouses and active-active data sources (Enterprise edition)
- Data profiling and validation, match and merge, creating glossaries, data lineage and metadata management (Governance edition)

What are the benefits of using Oracle Data Integration Platform Cloud?

You connect to many heterogenous on-premises and cloud data sources from a single platform and do all sorts of integration on your data entities. Data in any shape and format can be loaded and transformed on this platform. Data integration and synchronization happen in real time. Data will not get lost and data sources are highly available, even with large volumes of data. You have a choice for the type and number of sources and targets for data replication. All your unstructured data is handled well and data from this platform can be streamed into big data technologies for further ingestion and analytics. Because this is an Oracle Cloud product, you can easily integrate other Oracle cloud and non-cloud products such as Storage, Database, Exadata and Big Data Cloud with it.

Is Big Data Cloud Service certified to work with Oracle Data Integration Platform Cloud?

Yes, you can deliver data entities from DIPC to Oracle Big Data Cloud or other Big Data technologies. However, capturing data from Oracle Big Data Cloud or other Big Data technologies is not currently supported.

Service Editions

What editions does Oracle Data Integration Platform Cloud offer?

There are three editions: *Standard, Enterprise,* and *Governance.*

What can I do with each edition?

With Standard edition, you perform bulk data movement, ELT transformation, pushdown data processing and basic profiling of your data sources. Use this edition for data warehouses, data integration and migrations.

With Enterprise edition, you access all Big Data technologies, and real time data replication and streaming capabilities. Use this edition for Big Data integration, data synchronization, zero downtime migration, real-time data warehouses and active-active data sources. All Standard edition features are also included.

With Governance Edition, you profile, cleanse and govern your data sources with customized dashboards in this most advanced edition. Perform data health checks, enterprise data cleansing, and data lineage. Use this edition for data profiling and validation, match and merge, creating glossaries, data lineage and metadata management. All Enterprise edition features are also included.

How can I find out what edition I have for my service instance?

Find your DIPC service instance's name in the Services page of the DIPC console where all your service instances are listed. Click the service name to go to its detail page. The edition is displayed in the Service Overview section with acronyms of SE for Standard Edition, EE for Enterprise Edition and GE for Governance Edition.

Can I upgrade/downgrade the edition of a service instance?

No, whenever you create a DIPC service instance, you use the Create Service wizard which requires you to select an edition for your service instance. Based on your selections, one or more virtual machines (VMs) are created with the applications specific to your edition. To get a different edition, you must create a new service instance with your desired edition.

Oracle Data Integration Platform Cloud User Roles

What are all user roles provided by my cloud service instance? Can I create my own user roles?

You shouldn't create your own roles. In Oracle Identity Cloud, there are delegated administration roles that by default come with Identity Cloud, and then, there are application roles, set for you by the Data Integration Platform Cloud (DIPC) application. Use Oracle Identity Cloud to assign the following six DIPC application roles to your users.

Role	Description	Is Available for Granting?
Administrator	Service application administrator role	Yes
Developer	Service application developer role	Yes
Deployer	Service application deployer role.	Yes
Manager	Suite administrator role	Yes
Monitor	Suite monitor role	Yes
User	Service application user role	Yes

See About Oracle Identity Cloud Service.

How can I assign specific roles to my user?

Log in to your Identity Cloud Service. Find your service instance’s name in the Applications tab. Click your service instance name and then follow these instructions.

See Assigning Users to Oracle Applications

How do I add/remove users?

Add or remove users by following these instructions.

See Creating User Accounts

How can I restrict a normal user to cloud service instance deletion?

Assign that user with a "User" role. This role is not admin and can't delete any instances.

To assign a role to a user, log in to your Identity Cloud Service. Find your service instance’s name in the Application tab. Click the service instance name and then follow these instructions.

See Assigning Users to Oracle Applications.

> **Note:**
>
> All roles have full access to DIPC. Role-based Access Control (RBAC) is not implemented in the current version and will be added in a subsequent release.

Create an Instance (Provision)

This topic applies only to Oracle user-managed services.

What are other cloud services needed to be provisioned for my cloud service instance to work?

Before you create a DIPC instance, you should create an Oracle Database Cloud deployment (create an instance). This database is used to store the schemas for the DIPC server(s). One managed server if your cluster size is one and additional admin servers if your cluster is greater than one. You also need Oracle Storage Cloud for backups. You don't need to create the storage containers in advance. You can have the DIPC Create Instance wizard, create a Storage container for you. You'll provide your Cloud Storage username and password to either create one or use an existing container.

While provisioning a service instance, can I choose the data center of my choice?

No, at this moment, there is no option to select a data center of your choice.

Can I bring my own licenses for DIPC?

No, at this moment, you can't use your own license for DIPC. However, you can use your own Oracle Data Integrator or Oracle GoldenGate licenses to use with your on-premises data sources that are connecting to DIPC.

What compute options do I have when I create a a user-managed instance?

DIPC is offered in two compute shapes: `OC1m` and `OC3m`.

Can I upgrade to higher compute option later?

Yes, you can scale in/out/up/down for user-managed instances.

Data source information

What databases, platforms, or technologies can I use with Oracle Data Integration Platform Cloud?

See what's certified for each of the Data Integration Platform Cloud tasks:

- What’s Certified for Synchronize Data?
- What’s Certified for ODI Execution?

- What’s Certified for Replicate Data?
- What’s Certified for Add Data to Data Lake?
- What’s Certified for Data Preparation
- What's Certified for Hosted VM Tasks

Service Details and Native Applications on the VMs

How can I see various information about my cloud service instances such as the edition information or the number of nodes that I have in my service instances?

For details about any service instance, go to the instance detail page: From My Services dashboard that displays all your Oracle Cloud services, click the menu option for Data Integration. Click Open Service Console. Then, click the name of your service listed in the Services page.

Where are my native applications installed on a cloud service instance?

This topic applies only to Oracle user-managed services.

On the virtual machine of your cloud service, your applications are installed under /u01/app/oracle/suite/ with the following subfolders:

- `dics` (for some of the log files)
- `dicsTools` (provisioning tools)
- `ggbigdata` (GoldenGate for Big Data)
- `gghome` (GoldenGate for Oracle 12*c*)
- `gghome11g` (GoldenGate for Oracle 11*g*)
- `ggmysql` (GoldenGate for MySQL)
- `lost+found`
- `mwInventory`
- `oci` (Oracle Call Interface (OCI) for Oracle 12*c*)
- `oci11g` (Oracle Call Interface (OCI) for Oracle 11*g*)
- `suite_mw` (suite middleware for Oracle Data Integrator (ODI) and Oracle Enterprise Data Quality (EDQ) binaries)

> **Note:**
>
> There’s a `/u01/data/backup` folder, which is only accessible to the VM's administrator.

Can I change the default behavior of the native applications that are installed on the VMs of my cloud service instances? Can I alter the default configuration files like the cloud service parameter files, configuration files, and so on?

This topic applies only to Oracle user-managed services.

Yes, log in to the virtual machine of your service instance with the username opc and by using your instance's private key. Then use sudo to get the privileges of the user *oracle* with the following command:

```
[opc@YourInstance-wls-1 ~]$ sudo su - oracle
```

After that, you can update the parameter or configuration files on your VM to suit your data integration requirements. Use the VM, the same way you would if you had the applications on your computer.

> **Caution:**
>
> Do not change the installation or cloud service related files, because after all, you're working on a cloud service where the backups, updates, and patches are planned for the directory structure that the VM originally came.

What is the pre-requisite to run a workload in the virtual machine of my cloud service instance?

This topic applies only to Oracle user-managed services.

Run your data integration workloads as the *oracle* user. Use the command `sudo su - oracle` after you log into the VM of your service.

Can I uninstall the native applications on my cloud service’s virtual machines?

This topic applies only to Oracle user-managed services.

No, you should not uninstall the native applications of your virtual machines. With cloud services, a lot of programs may be running in the background to ensure a great data integration experience. You may break your cloud service by uninstalling its native applications. If you want less features, create a new service instance with a lower edition. Also, all upgrades and patches for a service instance should be done through the Administration tile of the DIPC service console.

Load Balancers

What are load balancers for cloud services?

Load balancers control the traffic to the applications that come with your service instances. They receive the client requests to the applications for a service and then, they distribute them among the VMs in a service instance. The overhead of encrypting and decrypting HTTPS traffic is also transferred to the load balancer. Load balancers can temporarily suspend traffic access to the service instance in order to perform routine maintenance. Each load balancer node is assigned a separate public IP address.

Do my service instances come with load balancers?

Yes, load balancers are included with Data Integration Platform Cloud (DIPC) service instances and they control traffic to the following applications. The users accessing these consoles can't see the IP address that's associated with these consoles and instead see an encrypted link. These consoles can be accessed through the menu option of each service instance in the Services page.

- Oracle Data Integrator (ODI) console
- Enterprise Data Quality (EDQ) console

Where can I find details about the load balancer that comes with my service instance?

Load balancer details can be found in each instance's detail page, in the Load Balancer section.

Security

Is my data secured inside my cloud service instance?

Yes, your data that's either in an Oracle Database Cloud deployment or your information, such as settings in the DIPC VM and your backups in Oracle Storage Cloud are all safe, as they're implemented with Oracle standards for keeping the VMs secure. Only you can access them with your private key. If you're using non-Oracle products as your data sources to connect to Data Integration Platform Cloud (DIPC), then obtain the security features for those data sources from their vendors.

Is my data secured while it's transmitted over the net from on-premises to cloud?

Yes. For your data integration needs, such as transferring, replicating, or loading data from on-premises or cloud data sources to other data sources, or to display your on-premises data entities in the DIPC platform, you either use SOCKS5 proxy or VPN. Both options are secure. For another level of security, data that's replicated from one data source to another uses the powerful and proven Oracle GoldenGate application's encrypted trail files to send data over the net, so it's completely secure.

Agents

Why does my agent download stop at 1 GB?

On Autonomous Data Integration Platform Cloud instances, when downloading the remote agent package, you may find that your download stops at 1GB, when the expected file size is greater than 1GB.

If you encounter this issue, you must update the nginx configuration and increase the proxy_max_temp_file_size to 5GB for each Data Integration Platform Cloud instance.

Data Synchronization

Does Data Integration Platform Cloud support on-premises to on-premises synchronization without moving data into the Cloud?

Yes, starting with release 17.4.5 with the following caveats:

- An on-premises Data Integration Platform Cloud remote agent (only available for Linux) is required. You need to download and install the agent to the appropriate source/target locations.
- A VPN connection between the on-premises systems and Oracle Cloud is required.
- This is limited to Oracle to Oracle replication processes.
- The Data Integration Platform Cloud instance in Oracle Cloud has to be running at all times while the GoldenGate processes are running, otherwise you won't be billed properly.

If your use case can be satisfied with all the conditions mentioned above, then you can create your Oracle GoldenGate artifacts — extract, data pump, and replicat, directly from within your on-premises Data Integration Platform Cloud remote agent installations and the data is routed accordingly. Connections must be defined within the Data Integration Platform Cloud Catalog for the replication source and targets.

> **Note:**
>
> The Data Integration Platform Cloud application in Oracle Cloud is not aware of these custom artifacts, as they are monitored outside of Data Integration Platform Cloud. Future releases will add support for monitoring these custom processes within Data Integration Platform Cloud.

Deliver Data to Oracle Autonomous Data Warehouse Cloud

How do I deliver data to Oracle Autonomous Data Warehouse?

To deliver data to Autonomous Data Warehouse from the Data Integration Platform Cloud console, see Replicate Data.

To deliver data to Autonomous Data Warehouse from a hosted VM, see Replicate Data to Oracle Autonomous Data Warehouse Cloud.

Maintain Your Service Instances

How can I release my cloud service instance?

All your service instances are listed in the Services page of the Data Integration Platform Cloud (DIPC) service console, each with a menu option that contains **Delete**. Use **Delete** when you no longer need your service instance and release the memory allocation and the OCPUs related to this service instance, back to your overall service credits. If you're not sure if it's OK to lose the information on your service instance, backup your instance before deleting it.

If I release my cloud service instance, then what happens to my data that was my cloud service instance?

To release a cloud service instance, so you can reuse the OCPUs and memory allocated to that service, you delete that service instance. If you choose to delete and instance, Oracle securely deletes all your information related to that service instance and no one can access it after it is deleted. You should backup your services if you need the data for later use.

How can I get my data back, after I release my cloud service instance?

When you release a service instance, you delete that service instance and all its backups. Therefore, a released instance cannot be recovered.

Can I scale in/out my service instances?

You can only delete the VMs of a cluster at this moment, but you can't add VMs to any cluster yet.

Can I scale up/down my service instances?

Not at this time. You can change the number of OCPUs for a service instance through the scale up/down menu option in the service instance's detail page whenever it becomes available.

How is patching done for cloud service instances?

Whenever a patch for a service instances becomes available, a message appears next to its name in the Services page. Click the service name to go to the detail page and then click the Administration tile. The Administration page has a tab called patching. In this page, you'll find a list of all available patches. Each patch has a **Precheck** and a **Patch** menu option. First click **Precheck** to ensure that your service needs a patch. Then click **Patch**, to apply the patch to your service. Each patch also has a Readme file for your information.

Backup and Recovery

This topic applies only to Oracle user-managed services.

Can I backup my data? If so, how? What is the data that’s getting backed up?

Yes. When you create your service instance, you associate an Oracle Storage Cloud container to your service for backups. Then, any time you'd like a backup, you go to your service instance's detail page, click the Administration tile and from there use the Manage Backups menu option of Backup Now or configure scheduled backups. A full backup contains these items:

- The `$DOMAIN_HOME` volume of each virtual machine (`u01/data/domains`). This includes any Managed Server persistent stores that are not stored in the database. These persistent stores are used for transaction logs and Java Message Service (JMS) providers.

- Domain configuration files in the `$MW_HOME` volume of the Administration Server virtual machine (`/u01/app/oracle/middleware`)
- Oracle Traffic Director configuration for the load balancer.

> **Note:**
>
> A service doesn't back up any software, including Oracle Fusion Middleware software installed on the `$MW_HOME` volume . You're responsible for ensuring that you can re-install any custom software that you've installed on a service instance's VMs if necessary.
>
> You cannot use the console to configure backups for a service instance if its Backup Destination was set to `None` when the instance was originally created.
>
> You cannot configure backups for a service instance while it is currently being backed up.

Where are my backups stored?

By default, backups are stored in the Oracle Storage Cloud Service container that was provided when the service instance was created, but you can enter a different existing container name when you configure your scheduled backups. Because you provide the username and password to the storage container used for backups, ensure that if you change the credentials of the cloud user that accesses this Oracle Storage Cloud Service container, you must also update the credentials used by Oracle Data Integration Platform Cloud to access this storage container for continuation of scheduled backups.

How long are my backups retained for?

Full scheduled backups are retained until their last linked incremental backup is no longer available. The additional retention period for full scheduled backups is fixed and you cannot change it. For On-Demand backups you choose to keep them forever, or assign a retention period.

Can I restore my data back to a new cloud service instance?

No, by backing up your service instances, you preserve them in a particular state. If you later make configuration changes to a service that you don't want, you undo them by restoring the service instance's configuration data from a backup and changes that occurred after this backup will be lost. You can also restore the applications on the virtual machine to their most current official patch set update (PSU) level. However, you restore your data to the same cloud service instance (same VM) that it originally resided on, and not a new one.

All the Oracle Storage Cloud containers are available through Storage Cloud's service console page, for you to copy and share.

Storage Administration

How can I increase the size of the storage that’s associated to an existing cloud service instance?

You don't need to increase the size of your storage. Backups are saved in Oracle Storage Cloud containers. All containers can grow in size as long as the sum of their sizes doesn't reach the maximum quota of the Storage Cloud Service itself. so it doesn't matter if you continue to add your backups to the same container, or to a new one within the same Storage. If you finish all your universal credits for all your Oracle Cloud Services, then you need to add more credits. Each VM that you create comes with 231 GB of Storage.

How can I attach my storage cloud service to a specific cloud service instance?

When you create a service instance, you can provide the Storage Container of your choice to be associated with the service instance. Then, by default, all on-demand and scheduled backups will be stored in this container. If you want to associate another container to your service instance, then you can assign this new container when you setup your scheduled backups. If you chose no backup for your service instance, when you provisioned it, you can still enable backups through the administration tile of the service instance and then associate a Storage Cloud container to your instance.

D
Kafka Connect Client Libraries

Find out the proper Kafka Connect client libraries to add to your classpath when you create a Kafka Connect connection in Oracle Data Integration Platform Cloud.

The maven central repository artifacts for Kafka Connect data sources are:

- **Maven groupId**: `org.apache.kafka`
- **Maven artifactId**: `kafka_2.11 & connect-json`
- **Maven version**: the Kafka Connect version numbers listed for each section

The following versions are compatible and certified for Oracle Data Integration Platform Cloud:

- Apache Kafka 1.1.x & Confluent Platform 4.1.x
- Apache Kafka 1.0.x & Confluent Platform 4.0.x
- Apache Kafka 0.10.2.x & Confluent Platform 3.2.x
- Apache Kafka 0.10.1.x & Confluent Platform 3.1.x
- Apache Kafka 0.10.0.x & Confluent Platform 3.0.x
- Apache Kafka 0.9.0.x & Confluent Platform 2.0.x

Refer to the Confluent sections to add additional libraries for the **Avro Converter** and its **Schema Registry**.

Kafka 0.11.0.0

```
connect-api-0.11.0.0.jar
connect-json-0.11.0.0.jar
jackson-annotations-2.8.0.jar
jackson-core-2.8.5.jar
jackson-databind-2.8.5.jar
jopt-simple-5.0.3.jar
kafka_2.11-0.11.0.0.jar
kafka-clients-0.11.0.0.jar
log4j-1.2.17.jar
lz4-1.3.0.jar
metrics-core-2.2.0.jar
scala-library-2.11.11.jar
scala-parser-combinators_2.11-1.0.4.jar
slf4j-api-1.7.25.jar
slf4j-log4j12-1.7.25.jar
snappy-java-1.1.2.6.jar
zkclient-0.10.jar
zookeeper-3.4.10.jar
```

Kafka 0.10.2.0

```
connect-api-0.10.2.0.jar
connect-json-0.10.2.0.jar
jackson-annotations-2.8.0.jar
jackson-core-2.8.5.jar
jackson-databind-2.8.5.jar
jopt-simple-5.0.3.jar
kafka_2.11-0.10.2.0.jar
kafka-clients-0.10.2.0.jar
log4j-1.2.17.jar
lz4-1.3.0.jar
metrics-core-2.2.0.jar
scala-library-2.11.8.jar
scala-parser-combinators_2.11-1.0.4.jar
slf4j-api-1.7.21.jar
slf4j-log4j12-1.7.21.jar
snappy-java-1.1.2.6.jar
zkclient-0.10.jar
zookeeper-3.4.9.jar
```

Kafka 0.10.2.0

```
connect-api-0.10.1.1.jar
connect-json-0.10.1.1.jar
jackson-annotations-2.6.0.jar
jackson-core-2.6.3.jar
jackson-databind-2.6.3.jar
jline-0.9.94.jar
jopt-simple-4.9.jar
kafka_2.11-0.10.1.1.jar
kafka-clients-0.10.1.1.jar
log4j-1.2.17.jar
lz4-1.3.0.jar
metrics-core-2.2.0.jar
netty-3.7.0.Final.jar
scala-library-2.11.8.jar
scala-parser-combinators_2.11-1.0.4.jar
slf4j-api-1.7.21.jar
slf4j-log4j12-1.7.21.jar
snappy-java-1.1.2.6.jar
zkclient-0.9.jar
zookeeper-3.4.8.jar
```

Kafka 0.10.0.0

```
activation-1.1.jar
connect-api-0.10.0.0.jar
connect-json-0.10.0.0.jar
jackson-annotations-2.6.0.jar
jackson-core-2.6.3.jar
jackson-databind-2.6.3.jar
jline-0.9.94.jar
jopt-simple-4.9.jar
junit-3.8.1.jar
kafka_2.11-0.10.0.0.jar
kafka-clients-0.10.0.0.jar
```

```
log4j-1.2.15.jar
lz4-1.3.0.jar
mail-1.4.jar
metrics-core-2.2.0.jar
netty-3.7.0.Final.jar
scala-library-2.11.8.jar
scala-parser-combinators_2.11-1.0.4.jar
slf4j-api-1.7.21.jar
slf4j-log4j12-1.7.21.jar
snappy-java-1.1.2.4.jar
zkclient-0.8.jar
zookeeper-3.4.6.jar
```

Confluent 4.1.2

```
avro-1.8.1.jar
common-config-4.1.2.jar
commons-compress-1.8.1.jar
common-utils-4.1.2.jar
jackson-annotations-2.9.0.jar
jackson-core-2.9.6.jar
jackson-core-asl-1.9.13.jar
jackson-databind-2.9.6.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
kafka-avro-serializer-4.1.2.jar
kafka-clients-1.1.1-cp1.jar
kafka-schema-registry-client-4.1.2.jar
log4j-1.2.16.jar
lz4-java-1.4.1.jar
netty-3.10.5.Final.jar
paranamer-2.7.jar
slf4j-api-1.7.25.jar
slf4j-log4j12-1.6.1.jar
snappy-java-1.1.7.1.jar
xz-1.5.jar
zkclient-0.10.jar
zookeeper-3.4.10.jar
```

Kafka 0.9.0.1

```
activation-1.1.jar
connect-api-0.9.0.1.jar
connect-json-0.9.0.1.jar
jackson-annotations-2.5.0.jar
jackson-core-2.5.4.jar
jackson-databind-2.5.4.jar
jline-0.9.94.jar
jopt-simple-3.2.jar
junit-3.8.1.jar
kafka_2.11-0.9.0.1.jar
kafka-clients-0.9.0.1.jar
log4j-1.2.15.jar
lz4-1.2.0.jar
mail-1.4.jar
metrics-core-2.2.0.jar
netty-3.7.0.Final.jar
scala-library-2.11.7.jar
scala-parser-combinators_2.11-1.0.4.jar
```

```
scala-xml_2.11-1.0.4.jar
slf4j-api-1.7.6.jar
slf4j-log4j12-1.7.6.jar
snappy-java-1.1.1.7.jar
zkclient-0.7.jar
zookeeper-3.4.6.jar
```

Confluent 4.0.0

```
avro-1.8.2.jar
common-config-4.0.0.jar
commons-compress-1.8.1.jar
common-utils-4.0.0.jar
jackson-annotations-2.9.0.jar
jackson-core-2.9.1.jar
jackson-core-asl-1.9.13.jar
jackson-databind-2.9.1.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
kafka-avro-serializer-4.0.0.jar
kafka-schema-registry-client-4.0.0.jar
log4j-1.2.16.jar
netty-3.10.5.Final.jar
paranamer-2.7.jar
slf4j-api-1.7.7.jar
slf4j-log4j12-1.6.1.jar
snappy-java-1.1.1.3.jar
xz-1.5.jar
zkclient-0.10.jar
zookeeper-3.4.10.jar
```

Confluent 3.2.1

```
avro-1.7.7.jar
common-config-3.2.1.jar
commons-compress-1.4.1.jar
common-utils-3.2.1.jar
jackson-annotations-2.5.0.jar
jackson-core-2.5.4.jar
jackson-core-asl-1.9.13.jar
jackson-databind-2.5.4.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
kafka-avro-serializer-3.2.1.jar
kafka-schema-registry-client-3.2.1.jar
log4j-1.2.17.jar
netty-3.7.0.Final.jar
paranamer-2.3.jar
slf4j-api-1.6.4.jar
slf4j-log4j12-1.7.6.jar
snappy-java-1.0.5.jar
xz-1.0.jar
zkclient-0.10.jar
zookeeper-3.4.8.jar
```

Confluent 3.2.0

```
avro-1.7.7.jar
common-config-3.2.0.jar
commons-compress-1.4.1.jar
common-utils-3.2.0.jar
jackson-annotations-2.5.0.jar
jackson-core-2.5.4.jar
jackson-core-asl-1.9.13.jar
jackson-databind-2.5.4.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
kafka-avro-serializer-3.2.0.jar
kafka-schema-registry-client-3.2.0.jar
log4j-1.2.17.jar
netty-3.7.0.Final.jar
paranamer-2.3.jar
slf4j-api-1.6.4.jar
slf4j-log4j12-1.7.6.jar
snappy-java-1.0.5.jar
xz-1.0.jar
zkclient-0.10.jar
zookeeper-3.4.8.jar
```

Confluent 3.2.1

```
avro-1.7.7.jar
common-config-3.1.2.jar
commons-compress-1.4.1.jar
common-utils-3.1.2.jar
jackson-annotations-2.5.0.jar
jackson-core-2.5.4.jar
jackson-core-asl-1.9.13.jar
jackson-databind-2.5.4.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
kafka-avro-serializer-3.1.2.jar
kafka-schema-registry-client-3.1.2.jar
log4j-1.2.17.jar
netty-3.7.0.Final.jar
paranamer-2.3.jar
slf4j-api-1.6.4.jar
slf4j-log4j12-1.7.6.jar
snappy-java-1.0.5.jar
xz-1.0.jar
zkclient-0.9.jar
zookeeper-3.4.8.jar
```

Confluent 3.1.1

```
avro-1.7.7.jar
common-config-3.1.1.jar
commons-compress-1.4.1.jar
common-utils-3.1.1.jar
jackson-annotations-2.5.0.jar
jackson-core-2.5.4.jar
jackson-core-asl-1.9.13.jar
```

```
jackson-databind-2.5.4.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
kafka-avro-serializer-3.1.1.jar
kafka-schema-registry-client-3.1.1.jar
log4j-1.2.17.jar
netty-3.7.0.Final.jar
paranamer-2.3.jar
slf4j-api-1.6.4.jar
slf4j-log4j12-1.7.6.jar
snappy-java-1.0.5.jar
xz-1.0.jar
zkclient-0.9.jar
zookeeper-3.4.8.jar
```

Confluent 3.0.1

```
avro-1.7.7.jar
common-config-3.0.1.jar
commons-compress-1.4.1.jar
common-utils-3.0.1.jar
jackson-annotations-2.5.0.jar
jackson-core-2.5.4.jar
jackson-core-asl-1.9.13.jar
jackson-databind-2.5.4.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
junit-3.8.1.jar
kafka-avro-serializer-3.0.1.jar
kafka-schema-registry-client-3.0.1.jar
log4j-1.2.17.jar
netty-3.2.2.Final.jar
paranamer-2.3.jar
slf4j-api-1.6.4.jar
slf4j-log4j12-1.7.6.jar
snappy-java-1.0.5.jar
xz-1.0.jar
zkclient-0.5.jar
zookeeper-3.4.3.jar
```

Confluent 2.0.1

```
avro-1.7.7.jar
common-config-2.0.1.jar
commons-compress-1.4.1.jar
common-utils-2.0.1.jar
jackson-annotations-2.5.0.jar
jackson-core-2.5.4.jar
jackson-core-asl-1.9.13.jar
jackson-databind-2.5.4.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
junit-3.8.1.jar
kafka-avro-serializer-2.0.1.jar
kafka-schema-registry-client-2.0.1.jar
log4j-1.2.17.jar
netty-3.2.2.Final.jar
paranamer-2.3.jar
slf4j-api-1.6.4.jar
```

```
slf4j-log4j12-1.7.6.jar
snappy-java-1.0.5.jar
xz-1.0.jar
zkclient-0.5.jar
zookeeper-3.4.3.jar
```

Confluent 2.0.1

```
avro-1.7.7.jar
common-config-2.0.0.jar
commons-compress-1.4.1.jar
common-utils-2.0.0.jar
jackson-annotations-2.5.0.jar
jackson-core-2.5.4.jar
jackson-core-asl-1.9.13.jar
jackson-databind-2.5.4.jar
jackson-mapper-asl-1.9.13.jar
jline-0.9.94.jar
junit-3.8.1.jar
kafka-avro-serializer-2.0.0.jar
kafka-schema-registry-client-2.0.0.jar
log4j-1.2.17.jar
netty-3.2.2.Final.jar
paranamer-2.3.jar
slf4j-api-1.6.4.jar
slf4j-log4j12-1.7.6.jar
snappy-java-1.0.5.jar
xz-1.0.jar
zkclient-0.5.jar
zookeeper-3.4.3.jar
```