

Oracle® Data Integrator

Reference Manual

10g Release 3 (10.1.3)

September 2008

Copyright © 2006, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Table Of Contents

Organization of This Manual	1
Designer	3
Introduction to Designer.....	3
Designer's Interface	3
Connection.....	4
Connecting to the Work Repository	4
Work Repository Connection Details	5
Driver Details.....	6
URL Samples	6
Projects	6
What is a Project?	6
Projects	8
Folders	8
Packages.....	9
Interface	20
Procedures.....	30
Variables	34
Sequences	35
User Functions	36
Knowledge Modules.....	37
Marker Groups	44
Models	45
What is a Model?	45
Models.....	46
Model Folders	50
Sub-models	50
Datastore.....	51
Operator	63
Introduction to Operator.....	63
Operator's Interface.....	63
Session	64
Session.....	64
Session Steps	65
Session Tasks.....	66
Session Variables	67
Scenario.....	68
Scenarios	68
Scenario Reports	69
Step Reports	70

Scheduling Information	71
Using the diagram	71
Topology Manager	73
Introduction to Topology Manager	73
Topology Manager's Interface.....	73
What is the Topology?	74
Physical Architecture.....	74
Contexts	74
Logical Architecture.....	75
Languages	75
Repositories	75
Hosts	75
Connection.....	75
Connecting to the Master Repository.....	75
Master Repository Connection Details.....	76
Driver Details.....	77
URL Samples	77
Physical Architecture	77
Technologies.....	78
Actions.....	91
Agents	93
Context.....	96
Definition	96
Agents	97
Schemas	97
Logical Architecture	97
Logical Schemas.....	97
Logical Agents.....	98
Languages	99
Languages	99
Sub-languages	100
Language Elements	101
Repositories.....	101
Master Repository	101
Work Repository.....	102
Hosts.....	103
Definition	104
Usage	104
Security Manager	105
Introduction to Security Manager.....	105

Security Manager's Interface	105
Connection.....	106
Connecting to the Master Repository.....	106
Master Repository Connection Details.....	107
Driver Details.....	108
URL Samples	108
Objects.....	108
Object.....	108
Method	109
Profiles.....	109
Profile	109
Authorization by Profile	111
Users.....	111
User.....	112
Authorization by User.....	112
Authorization by Object Instance	113
Appendices.....	115
Date Format.....	115
Available Symbols.....	115
Examples.....	116
JDBC URL Samples	116

This manual describes the details of Oracle Data Integrator graphical user interface. It is intended for advanced developers and administrators who already know Oracle Data Integrator and want to make good use of it.

Organization of This Manual

This manual contains the following:

- **Chapter 1 to 4** provide a reference for the graphical user interface of Oracle Data Integrator.

Introduction to Designer

Through the **Designer** module, you can handle:

Models: Descriptions of the data and applications structures

Projects: The developments made with Designer.

The Designer module stores this information in a work repository, while using the topology and the security information defined in the master repository.

Designer's Interface

The Designer GUI appears as follow:

The Menu

The **Menu** contains pull-down menus to access the following features:

- Import/Export
- Wizards
- Display options
- Open modules or tree views
- Change the user's password and options

The Toolbar

The **Toolbar** lets you:

- Open other modules
- Refresh the Tree views
- Open the on-online help
- Choose the default **context**

The **context** selected is used as default for all context choices to be made in the application windows. In addition, when data is consulted (right-click on data on a datastore), Designer displays the data in the context defined in the toolbar. For example, if your context is "Development", you consult the data in the "development" context. As a safety precaution, even if you are authorized for all contexts, you are asked for a confirmation password at every change of context, to avoid any inappropriate manipulation. Only the contexts you are granted appear in the menu bar.

The Tree Views

Designer objects available to the current user are organized into the following tree views: **Projects**, **Models**, **Solutions** and **Others** (User Functions, Global Variables and Sequences).

Each tree view appears in a floatable frames that may be docked to the sides of the main window. These frames can be also be stacked up. When several frames are stacked up, tabs appear at the bottom of the frame window to access each frame of the stack.

Tree view frames can be moved, docked and stacked by selecting and dragging the frame title or tab. To lock the position of views, select **Lock views** from the **Windows** menu.

If a tree view frame does not appear in the main window or has been closed, it can be opened using the **Windows > Show View** menu.

It is possible in each tree view to perform the following operations:

- Insert or import root objects to the tree view by clicking the appropriate button in the frame title
- Expand and collapse nodes by clicking on them
- Activate the methods associated to the objects (Edit, Delete, ...) through the popup menus
- Edit objects by double-clicking on them, or by drag and dropping them on the **Workbench**

The Workbench

The windows for object being edited or displayed appear in the **Workbench**.

Connection

Connecting to the Work Repository


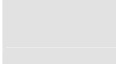
With this window, a connection to a work repository can be selected when opening the **Designer** module. The creation, modification or deletion of existing connections can also be launched.

Note: The definitions of the work repository connections are stored locally in the `/bin/snps_login_work.xml` file

General Properties

Properties	Description
Login name	List of the defined connections to the work repositories.
User	The default Oracle Data Integrator user for connecting to the selected work repository. If no default user has been specified, this must be completed.
Password	The password corresponding to the Oracle Data Integrator user for the connection. If no default password has been specified, this must be completed.

Toolbar

Button	Description
	Allows a new connection to a work repository to be created.
	Allows the connection selected in the login name field to be modified

Allows the connection selected in the **login name** field to be deleted

Work Repository Connection Details

With this window, a connection to a **work repository** can be defined or modified.

Note: Each work repository is attached to a master repository, therefore, information about the physical connection to a work repository is stored in the master repository it is attached to. Defining a connection to a work repository consists of defining a connection to a master repository, then selecting one of the work repositories attached to this master repository.

Note: The definitions of the work repository connections are stored locally in the `/bin/snps_login_work.xml` file

General Properties

Properties	Description
Oracle Data Integrator Connection	With this information group, the properties displayed in the connection window can be defined.
Login name	This is the name presented in the list when a Data Integrator module is opened.
User	The default Oracle Data Integrator user for connecting to the work repository. If no default user has been specified, the user must enter it each time this connection is used.
Password	Password corresponding to the Oracle Data Integrator user. If no default password has been specified, the user must enter it each time this connection is used.
Database Connection (Master Repository)	This information group shows the connection parameters for the relational database that hosts the master repository.
User	User allowing you to connect to the schema of the database containing the master repository that the work repository is attached to.
Password	The data server password for the database user.
Driver List	Type of technology the master repository is based on. You can obtain more information about the technology and the driver used by clicking on the button. For more information, refer to Driver details.
Driver name	Name of the JDBC driver used for connecting to the data server of the master repository. When a technology is selected, Data Integrator displays the most frequently used driver for this technology. Nevertheless, a different one can be used. For this, this driver must be correctly installed. For more information, refer to Installing JDBC / JMS drivers in the installation guide.

Url	URL for connecting to the data server of the master repository. You can select a URL model for the driver used by clicking on the button. For more information, refer to URL Samples. For more information on drivers and URLs, refer to the section JDBC URL Sample.
Work Repository	This information group shows the work repository attached to the master repository to be used for the connection.
Repository Name	Name of the work repository. If the master repository connection has been correctly specified, you can access the list for selecting of one of the work repositories attached to this master repository by using the button
Default Connection	If this box is checked, when an Oracle Data Integrator graphic module is started, the current connection will be proposed by default.

Driver Details

This window provides information about the driver used with a technology.

General Properties

Properties	Description
Name	The driver used for the selected technology.
Description	Information about the driver.

For more information on drivers and URLs, refer to the section JDBC URL Sample.

URL Samples

This window provides URL models for a driver.

General Properties

Properties	Description
Name	List of valid URL models for the selected driver.
Description	Details on the URL model.

For more information on drivers and URLs, refer to the section JDBC URL Sample.

Projects

What is a Project?

A **project** is a group of objects developed using Oracle Data Integrator.

- Managing a Project

Oracle Data Integrator Project Components

The following components are stored below the level of the project in the tree view:

Folder

Certain objects in a project are organized into **folders** and **sub-folders**.

Packages

The **package** is the largest unit of execution in Oracle Data Integrator. A package is made up of a sequence of **steps** organized into an execution diagram.

- More information about Packages
- Create a Package

Interface

An interface consists of a set of rules that define the loading of a Datastore or a temporary target structure from one or more source Datastores.

- More information about Interfaces
- Create an Interface

Procedure

A Specific Procedure is a reusable component that groups operations that do not fit in the Interface framework - loading a target datastore from one or more sources.

Examples of procedures:

- wait and unzip a file
- send a batch of files via FTP
- receive emails
- purge a database

A procedure can launch commands on logical schemas defined in the topology, but may also use OS commands or Oracle Data Integrator tools.

Variable

A variable's value is stored in Oracle Data Integrator. This value may change during the execution.

The value:

- has a default defined at creation time
- can be passed as a parameter when running a scenario using the variable,
- may change with the refresh, set and increment variable steps ,
- may be evaluated to create conditions in the packages,
- may be used in the interfaces, procedures, steps, ...

A variable can be defined outside a project (global scope), in order to be used in all projects.

Sequence

A sequence is an variable automatically incremented when used. Between two uses the value is persistent.

The sequences are usable like variable in interfaces, procedures, steps, ...

A sequence can also be defined outside a project (global scope), in order to be used in all projects.

User Functions

User functions enable to define customized functions or "functions aliases", for which you will define technology-dependant implementations. They are usable in the interfaces and procedures.

Knowledge Modules

Oracle Data Integrator uses knowledge modules to define methods related to a given technology. These modules enable processes generation for the technology, dedicated to a precise function.

Note: Default knowledge modules are provided with Oracle Data Integrator, and must be imported in the project before use.

Marker

Elements of a project may be flagged in order to reflect the methodology or organization of the developments.

Flags are defined using the markers. These markers are organized into groups, and can be applied to most objects in a project.

Scenario

When a package, interface, procedure or variable component is finished, it is compiled in a **scenario**. A scenario is the execution unit for production, that can be scheduled.

Projects

Definition

Properties	Description
Name	Name of the project, as it appears in the Oracle Data Integrator graphical interface.
Code	The project code is used as the prefix for project-type variables. Modifying it will therefore change the name of project-type variables.

Folders

A folder is a group of packages, interfaces and specific procedures. Folders and sub-folders allow these objects to be grouped and organized according to criteria specific to the project. Sub-folders can be created to an unlimited number of levels.

Note: To create a sub-folder, simply create a folder, then drag and drop it into the parent folder.

Definition

Properties	Description
Name	Name of the folder or sub-folder, as it appears in the Oracle Data Integrator graphical interface.

Packages

What is a Package ?

The **package** is the biggest execution unit in Oracle Data Integrator. A package is made of a sequence of **steps** organized in an execution diagram.

- Create a Package.

Steps

There are different type of steps. They can be grouped in families of steps:

- **Flow (Interface):** Executes an interface.
- **Procedure:** Executes a procedure.
- **Variable:** Declares, sets, refreshes or evaluates the value of a variable.
- **Oracle Data Integrator Tools:** These tools, available in the Toolbox, enable to access all Oracle Data Integrator API commands, or perform operating system calls.
- **Models, Sub-models and Datastores:** Performs journalizing, static control or reverse-engineering operations on these objects.

For example, the "Populate Sales Datamart" Package could be made up of the following jobs:

1. Procedure "System Backup"
2. Interface "Customer Group"
3. Interface "Customer"
4. Interface "Product"
5. Refresh variable "Last Invoice ID"
6. Interface "Invoice Header"
7. Interface "Invoice Lines"

Packages

Definition

Properties	Description
Name	Name of the package, as it appears in the Oracle Data Integrator graphical interface.
Description	Detailed description of the package.

Diagram

Refer to the Diagram topic for a detailed description of the package diagram.

Execution

A package can be executed directly without having to generate a scenario. By executing a package, the smooth functioning of the series of steps that make it up can be tested. Execution is launched by means of the **Execute** button. The window for selecting the execution options opens.

On the **execution** tab, the result of preceding executions is detailed in a chart containing the following elements:

Properties	Description
Date	Start date and time of execution of the package.
Context	Execution context of the package.
Agent	Name of the agent who executed the package. The note internal shows that the agent used is the one contained in Designer .
Duration	The time taken for execution of the package.
Log Level	Logging level, chosen on execution of the package.
Error	Return code of the package.

Note: A package can also be launched by selecting **Execute** in the contextual menu of the **package** (in the **Designer** module tree).

Scenarios

This tab displays in a table view the scenarios generated for this component, as well as their schedules. You may double-click a scenario or a schedule to display its properties.

It is possible to create and delete scenarios by clicking the **Generate** and **Delete** buttons.

The popup menus for scenarios and schedules are also available from this tab.

Diagrams

The **Package Diagram** tab is used to define the sequence of **Steps** within a **Package** in a graphical way.

The Diagram tab is separated in 3 panes :

- The **toolbar**,
- The package **diagram** pane which contains a graphical view of the steps and the sequence of steps,

- The **properties** pane, displaying information about the selected step.

In a package, a step is always, upon success or failure, followed either by another step (represented by a green or red link), or by the end of the package (not represented).

The Toolbar

The toolbar contains the tools in order to:

- Select a link or a step
- Define the next step upon step success
- Define the next step upon step failure
- Display/hide the properties panel
- Duplicate the selected step
- Delete the selected steps/links
- Reorganize the diagram
- Edit the linked object to a selected step
- Execute the selected step
- Execute the whole package

The **Error** button changes color when there is an error in the package organization (for instance unlinked step). Click on the button to have a detail of the errors. If it is not selectable, there are no errors in the package.

The Diagram pane

You can add steps to the package by object drag and drop (an interface, variable, specific treatment) from the tree view into the diagram pane.

In the diagram pane, each step appears with an icon specific to the step type. The sequence of steps appear as green arrows (for success), and red arrows (for failure). For **Evaluate Variable** steps, both arrows are green with marker indicating if the next step is executed in the condition is "true" or "false".

It is also possible to rearrange using either steps and links drag and drop, options of the step's contextual menu, or the **Reorganize** button to automatically rearrange the steps position.

You can open a step's **Properties** pane by clicking on its icon, and open the **Linked Object** (an Interface for a flow step, a variable for an evaluate variable step, ...) by double clicking on the Step icon.

You can **Execute, Duplicate, Edit or Delete** a step by right clicking on it and selecting the appropriate option in the contextual menu.

The Properties pane

General tab

This tab describes the information about the step. The properties on this tab depend on the selected step type.

Common properties

The following properties appearing for most step types.

Properties	Description
Name	Name of the step, as it appears in the Oracle Data Integrator graphic interface
Type	Type of the step. This field does not appear for Oracle Data Integrator tools.
Linked object	This is the name of the interface, procedure, model, sub-model, datastore or variable inserted into the package to create this step. This field does not appear for Oracle Data Integrator tools.
Path	Path to the linked object. Contains for example the project and folder containing the interface for a flow step.

Specific properties

Below are detailed the properties specific to each type of step. The steps not indicated below do not have specific properties.

Set Variable

For a **Set variable** step, the following properties appear:

Properties	Description
Assign the value	When this option is checked, the variable takes the value specified in the value field on the variable tab.
Increment the value	This option only appears for numerical-type variables. It increments the variable value with the value defined in the increment field.
Value/Increment	The value assigned to the variable (in the case of an assignation), or the increment added to the variable (in the case of increment of the numerical variable).

Evaluate Variable

For an **Evaluate variable** step, the following properties appear:

Properties	Description
Operator	The relational operator between the value of the variable and the value defined in the value field on the variable tab.
Value	The value compared to the value of the variable.

Remark: It is possible to compare two variables by typing in the **value** field the name of the variable to be compared to. The variable name must be prefixed with GLOBAL or the name of the project containing this variable.

Note: When using the IN operator, the list of values into which the variable value is searched for is a semicolon separated list such as AB;AC;AA;AE or 66;33;88;11;22;92.

Journalizing ... steps

These properties only appears for **Journalize ...** type step. The fields describe below appear or not depending on the object (datastore, model) being processed and on the type of journalizing mode (Consistent Set or Simple)

Properties	Description
Start	Starts Journalizing for the datastore, model or sub-model.
Stop	Stops Journalizing for the datastore, model or sub-model.
Add subscriber	Create the subscriptions for the list of Subscribers. You can add or remove subscribers from the list using the Add or Delete buttons.
Remove subscriber	Deletes the subscriptions indicated in the list of Subscribers.
Extend Window	Extends the consistency window for the CDC set or the datastore.
Purge Journal	Purge the journals from useless entries.
Lock Subscribers	Locks entries in the consistency windows for the given subscribers.
Unlock Subscribers	Unlocks the entries in the consistency windows for the given subscribers. Validates consumption of the changes.

Check ... steps

These properties only appears for all **Check ...** type step.

Properties	Description
Delete Errors from the Checked Table	Deletes the erroneous data from the checked datastore, model or sub-model.

Execute OS Command steps

These properties only appears for all **Execute OS command** type step.

Properties	Description
Order	OS Order or Oracle Data Integrator Tool launched by the agent. For more information about Oracle Data Integrator tools, refer to the syntax of Data Integrator Tools

Note: OS Commands and Oracle Data Integrator Tools are launched by an agent in a given environment (OS, machine, etc), and therefore, this environment features should be taken into account (file access path, OS order syntax, etc).

Additional Variables

This tab appears when using the SnpsStartScen Tool. You can specify in the tab the list of startup variables for this scenario.

Command Tab

This tab appears when using an Oracle Data Integrator Tool. It contains the command corresponding to the tool called with the parameters specified in the general tab.

Advanced tab

This tab allows the behavior of the package to be determined according to whether the step succeeds or fails.

Conditions of success or failure

- A **Flow (Interface), Procedure, Execute OS Command, or Execute Oracle Data Integrator Tool**-type step is successful, if the procedure, interface or command has been correctly executed (return code 0). If not, it has failed.
- A **Set Variable**-type step is successful, if the assignation has been correctly carried out. If the assignation is not possible (for example, assigning a chain in a numerical), the step has failed.
- A **Refresh Variable**-type step is successful, if the Select order that allows the value of the variable to be calculated has been correctly executed in the execution context, and if the value is correct for the type of variable. If not, the step has failed.
- An **Evaluate Variable**-type step is successful, if the condition `<variable> <operator> <value>` is true. If not, it has failed.
- A **Journalize, reverse or check** step is successful if the operation is correctly executed (Return code 0). If not, it has failed.

Behavior following success or failure

You can define the behavior following success or failure by specifying the following parameters:

Properties	Description
On Success	Describes the behavior to implement if the step has been successful
End	Execution of the package is stopped, and all transactions are subject to a Rollback.
Execute step	This field allows a package step to be selected. The special step <<Next Step>> allows the step following the current step in the package to be executed.
On Failure	Describes the behavior to implement if the step has failed
Number of Attempts	Defines the number of times the step will be re-launched if it has failed.
Interval between each Attempt	The time interval in seconds between two re-launches of the step.
End	Execution of the package is stopped, and all transactions are subject to a Rollback. This behavior is implemented after the retries.
Execute step	This field allows a package step to be selected. The special step <<Next Step>> allows the step following the current step in the package to be executed. This behavior is implemented after the retries.
Log Steps in the Journal	Describes how the step information is logged in the journal when its execution is finished.
Log Steps in the Journal	Before its execution and while being executed, a step appears in the execution log. This drop box defines whether the step should be kept in the journal after its execution is finished: <ul style="list-style-type: none"> • Never: the step is deleted from the journal.

- Always: the step is always kept in the journal.
- Errors: the steps is kept in the journal only if it failed. Otherwise, it is deleted.

Default behavior

The default behavior of steps in a package is as follows:

- If the step is successful, it moves on to the following step in the classification order within the package.
- If the step ends abnormally, execution of the package is stopped, and all transactions are subject to a rollback.

Some models of standard series and iterative loops are given in Examples of series.

Options tab

This tab allows to set the values of the options of a procedures or knowledge module (for journalizing, checking or reverse-engineering). It is only used for a **Procedure**-type or A **Journalize, reverse or check** step. For more information about options, refer to the options section.

Properties	Description
Name of the option	Name of the procedure option the value is assigned to.
Value	Value assigned to the option for the execution

Note: The KM option values are specified in the interfaces that use the KMs.

Execution Parameters

When a package or a step is launched, execution parameters must be specified. As this information is not saved in the Repository, you must specify it each time.

Properties	Description
Context	Specifies the context in which the session is launched. Only the contexts you are granted rights to appear here.
Agent	The agent which will execute the package or step. The package or step can also be directly executed by Designer, by selecting Local (No Agent) .
Log level	Level of logging information to retain. All session tasks with a defined log level lower than or equal to this value will be kept in the log when the session completes. However, if the package or step ends abnormally, all tasks will be kept, regardless of this setting.

Examples

Definition of an iterative loop

To create a loop that repeats ten times, simply create a numerical variable, `snp_increment`, that contains the increment, then insert the following three steps:

- Step 1 (Initializing loop): A **Set Variable** -type step which assigns '0' to `snp_increment`.
- Step 2 (Increment loop): A **Set Variable** -type step which increments `snp_increment` by '1'.
- Step 3 to n: ...actions to execute in a loop n times.... The `snp_increment` variable can be used in these procedures.
- Step n+1 (loop end test): An **Evaluate Variable** -type step which tests `snp_increment<=10`. If successful, **execute the task:** "step 2", if failed, **execute the task:** "<<next step>>"
- Step n+2: next actions

Scenario

Scenarios

A scenario is designed to put a source component (interface, package, procedure, variable) into production. A scenario results from the generation of code (SQL, shell, etc) for this component.

Note: Once generated, the scenario is stored inside the work repository. The scenario can be exported then imported to another repository (remote or not) and used in different contexts.

Note: 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.

The scenario properties are displayed on the **Definition** tab in the **Scenario** window in the **Designer** module, or via the **Operator** module. Operation of a scenario takes place from the same window or from an operating system command window. In the Designer module, scenarios are grouped under their respective source components.

Properties

Properties	Description
Package/Interface/Variable/Procedure	Name of the component used to create the scenario
Name	Name of the scenario
Version	Version of the scenario
Description	Detailed description of the scenario

Execution

This tab allows you, by using the **Execute** button, to launch the scenario. The window for selecting the execution options opens.

Variables

This tab allows you to select the scenario variables. They will be displayed as parameters when starting the scenario using Metadata Navigator or when creating a schedule. This selection of parameter variables lets you hide from the user variables that cannot be parameterized.

If the **Use All** value is selected, then all variables are considered as parameters. If the **Selective Use** value is selected, you can select the variables to use as parameters.

Operating a scenario

A scenario is operated from the work repository it is stored in. The same scenario can be executed in several contexts from the same work repository.

A scenario can be:

- launched from Designer or Operator
- Scheduled with Data Integrator scheduler
- launched from an Operating System command line interface, using a Web Service or an HTTP URL.
- Scheduled with any Scheduler
- exported to another work repository where it can be similarly launched or scheduled. See Operating in another repository.
- Import a scenario in production.

Schedules

If you have the **Scheduler** option, you will be able to schedule execution of your scenarios. With the **Scheduler** option, the agent will automatically activate the scenarios according to a predefined scheduler.

A schedule concerns only one scenario, but a scenario can be scheduled in several ways, and can have several schedules. 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.

A schedule is created by using the **Insert Schedule** option in the contextual menu of a scenario.

General parameters

Properties	Description
Context	Execution context of the scheduled scenario
Agent	Agent responsible for executing the scenario
Log Level	Logging level. All the tasks whose logging level is lower than or equal to this value will be saved in the log at the end of execution.
State	Defines activation of the schedule
Active	The scheduling will be active when the agent is restarted or when the scheduling of the physical agent is updated. An active schedule appears with the following icon in the tree view.
Inactive	The schedule cannot be active. An inactive schedule appears with the following icon in the tree view.
Active for the	Activity range of the schedule. A schedule active for a period appears with the

period	following icon in the tree view.
Execution	Defines the frequency of execution of each execution cycle
Execution	Frequency of execution option (annual, monthly, ... simple) This option is completed by a set of options that depend on this main option

Execution cycle

The execution cycle shows the repeat mode of the scenario.

Properties	Description
One time	The scenario is executed once only.
Many times	The scenario is repeated several times.
Maximum number of repetitions	The maximum number of times the scenario is repeated during the cycle.
Maximum Cycle Duration	As soon as the maximum time is reached, the scenario is no longer restarted, and the cycle stops.
Interval between repetitions	The downtime between each scenario execution.
Constraints	Allows limitations to be placed on the cycle, in the event of a problem during execution.
Number of attempts on failure	Maximum number of consecutive launchings returning an error
Stop Execution after	Maximum cycle time. If this time is reached, the scenario is automatically stopped.

Note: After an unexpected agent scheduler stop, the schedules in progress (those in the middle of an execution cycle) and that can be restarted will be automatically recovered when the agent restarts.

Variables

When you create a schedule for a scenario, you can define the values taken by the parameter variables for the scheduled executions.

Properties	Description
Name	Name of the variable. It is prefixed by GLOBAL for a global variable, or by the code of the project containing the variable.
Default	The variable takes its last value (or its default value if it has never been set) if this box is checked. If its is unchecked, the variable takes the value indicated in the value field.
Value	Value taken by the variable for the execution.

Scenario Reports

The scenario execution report gives information on the general execution of a scenario.

Definition

Properties	Description
Execution agent	Agent responsible for executing the scenario.
State	State of the scenario (Done, Error, Running, Waiting, Warning).
Execution context	Execution context of the scenario.
Start	Start date and time of execution of the scenario.
End	End date and time of execution of the scenario.
Duration	The time taken for execution of the scenario.
Return code	Return code of the scenario.
Message	Scenario execution error message, if needed.

Details

Properties	Description
No. of inserts	Number of rows inserted during the scenario.
No. of Deletes	Number of rows deleted during the scenario.
No. of Updates	Number of rows updated during the scenario.
No. of Errors	Number of rows in error in the scenario.
No. of Rows	Total number of rows processed by the scenario. Caution, this value is not the sum of the above values.

Step Reports

The step execution report is linked to the scenario execution report.

Definition

Properties	Description
Name	Name of the step executed.

Order number	The order number linked to the step for this session.
--------------	---

Execution

Properties	Description
Number of executions	In the event of successive executions of the step (loops), the execution number.
Execution	Gives the detailed result of execution of the step
Start	Start date and time of execution of the step.
End	End date and time of execution of the step.
Duration	The time taken for execution of the step.
State	State of the step (Done, Error, Running, Waiting or Warning).
Return code	Return code for the step.
Message	Step execution error message, if needed.
No. of Inserts	Number of rows inserted during the step.
No. of Deletes	Number of rows deleted during the step.
No. of Updates	Number of rows updated during the step.
No. of Errors	Number of rows in error in the step.

Interface

What is an Interface?

An interface consists of a set of rules that define the loading of a datastore or a temporary target structure from one or more source datastores.

- Create an Interface

Components of an Interface

Target Datastore

The **target datastore** is the element that will be loaded by the interface. This datastore may be permanent (defined in a model) or temporary (created by the interface in the staging area).

Source Datastores

The **source datastores** contain data used to load the target datastore. Two types of datastores can be used as a source of an interface: The datastores from the models, and temporary datastores target of an interface.

The source datastores of an interface can be filtered during the loading process, and must be put in relation through joins. **Joins** and **filters** can be recovered from the models definitions, and can also be defined for the interface.

Mapping

The **mapping** defines the transformation rules on the sources enabling to generate data to load the target.

Flow

The **flow** is the set of loading and integration strategies for mapped data, based on **knowledge modules**.

Control Strategy

The **flow control strategy** lets you define the method used to check the flow before the insertion in the target. The control strategy is defined by a **Check Knowledge Module (CKM)**.

The interfaces use the following components:

- **Datastores** defined in the models as sources and target or the loading process.
- **Knowledge Modules** to generate the appropriate processes.
- **Variables** and **Sequence** to store values or counters in the expressions.
- **Users functions** to ease the transformation rule coding.

Definition Tab

This tab defines the general properties of an interface.

Properties	Description
Name	Name of the interface that appears in the graphical interface.
Optimization context	<p>This is the context which will be used by Designer during design of the interface to determine the necessary data to load, to check the syntax of expressions and to display the data.</p> <p>The execution context of the interface (chosen when the interface is launched) can be different from the optimization context, as long as there is compatibility between the physical architectures referenced through the different contexts.</p>
Staging Area different from Target	<p>As a general rule, if the target data server has join and transformation capabilities (classic SQL capabilities), it is advised to use the engine of this data server for the transformations.</p> <p>Temporary objects will be created in a schema on this server called the Staging Area. In the case of an interface with a temporary target table, the temporary table will be created in this space.</p> <p>If this is not the case (if the target is a MOM or a file, for example), you can choose another transformation engine, which can be another data server, or the internal engine working in memory.</p> <p>Caution: Only by choosing a transformation engine on the target can the</p>

data flow consistency be checked automatically.

Description Detailed description of the interface.

Diagram

Diagram Tab

The sources, the target, and the transformation and mapping rules are defined in the **Diagram** tab of the **Interface** window.

Use this tab to define:

- A source data model in the form of an "entity-relation" diagram (like in an SQL query editor). It appears in the **composition panel** on the left side of the window. A data model is made up of source datastores (permanent or temporary), joins and filters.
- A target datastore.
- A mapping, the set of correspondence rules between source and target data.

Diagram options

The options, which are displayed above the diagram, are as follows:

Properties	Description
Columns	Display column-to-column joins (as black lines).
Datastores	Display only the joins between tables (blue lines), which can be useful when there are many joins.
Sets	Draw green arrows to represent the construction order of the sets resulting from the joins. The order number that appears in the join properties allows the set construction order to be modified. This display will be modified accordingly. (A join with a order number near 1 will be constructed first).
Errors	Allow errors detected on the interface to be displayed. When an error is detected, the text of the button turns red. It may also appear yellow when there are some warnings.
Properties	Displays the properties of joins or tables (bottom panel).

Source Datastores

An interface can use two types of datastore as source:

- Permanent datastores, resulting from models.
- Temporary datastores, created by another interface.

The same datastore can be used several times as a source, provided it has a different alias.

To edit the properties of a source datastore, simply click once only on the entity representing it within the diagram, and the properties will appear at the bottom of the composition panel.

Properties	Description
------------	-------------

Journalized Data only	<p>Indicates that only Journalized (modified) lines of data will be taken as sources for this interface.</p> <p>When this box is checked, a filter is automatically created on the source datastore in order to define the lines to take.</p>
Alias	The alias allows a unique name to be given to the datastores participating in the interface. This name should be concise, as it is used before every column name in all expressions.
Order	This is the order in which the datastore will appear when procedures are generated in FROM clauses of SQL queries. This order is ignored in sequenced joins (ISO).
Context	<p>Context in which the datastore must be systematically looked for.</p> <p>By default, Oracle Data Integrator will choose the optimization context (during design) and the execution context (during interface execution). A context needs to be selected only if the interface is Multi-context.</p> <p>For example: To load the "Test" context tables from the "Production" context tables.</p>

Visual indicators

Certain properties of source datastore columns are displayed with visual indicators.

Indicator	Description
	Column is part of this datastore's primary key.
n	Column is used in a mapping, join or filter. The letter is generally the first letter of the column type. n is numeric, s is string and so forth.
n	Column not used in any mapping, join or filter.
*	Mandatory column as defined on the model.

Joins

If your interface includes more than one source datastore, they must be linked with one another by a join. The number of joins on a source datastore cannot be lower than the number of source datastores minus one.

To edit the properties of a join, click on the relation representing it within the diagram.

Properties	Description
Active Clause	This checkbox indicates that the Join must be taken in account when executing the interface.
Implementation	<p>A join expression is usually expressed in SQL language. For sequenced joins (ISO), this is the text of the "ON" clause.</p> <p>For example: REG.REGION_ID=CIT.REGION_ID</p>
Technical	This field cannot be edited. It provides a version of the join with all column

description	names replaced by their descriptions.
Business Rule	This field enables you to enter a business description and/or to provide notes and comments for the join.
Execute on:	Transformation engine constructing the join. <ul style="list-style-type: none"> • Source: This is the data server hosting the two source datastores. This server must have join capacities. • Staging Area: Transformation engine selected on the Definition tab for the interface. By default, this is the Target engine.
Operator	<p>According to the relational model, data sets can be joined with different operators. For two joined tables, one with 10 rows and one with 20 rows:</p> <ul style="list-style-type: none"> • CROSS: product set systematically returning 200 rows (10 * 20). • INNER JOIN: retains in the two joined tables only the rows respecting the join clause, that is, between 0 and 20 rows. • LEFT/RIGHT OUTER: join known as external, which returns only the rows of the table situated on the left or right of the join operator. According to the case, it will return 10 or 20 rows. • FULL: is equivalent to an external join on each side. According to the case, this operator will return between 20 and 30 result rows. <p>On some technologies, not all these operators are valid, in which case, the join symbol is crossed out in red.</p>
Ordered Join (ISO)	Check this box if the join is to be sequenced (also called ISO syntax).
Order number	<p>For sequenced joins, this determines the order in which joins are resolved.</p> <p>For example, if there is a join between A and B, and another between A and C, you need to know if A and B are to be joined first, then the result with C, or rather, A and C, then the result with B. The order number allows this to be determined. A low order number shows the join will be resolved first.</p>
Compute datastores	This button force Oracle Data Integrator to recalculate the join. It evaluates the code of the join, and specifies the datastores that are used. Errors may occur in specific cases (missing datastores, revoked privileges).

Filters

Filters can be placed on the source datastores of interfaces to restrict the data to be taken as source by the interface.

To edit the properties of a filter, click on the filter symbol in the diagram.

Properties	Description
Active Filter	This checkbox indicates that the Filter must be taken in account when executing the interface.
Implementation	A filter expression is usually expressed in SQL language.
Technical description	This field cannot be edited. It provides a version of the filter with all column names replaced by their descriptions.

Business Rule	This field enables you to enter a business description and/or to provide notes and comments for the filter.
Execute on	Transformation engine that carries out the filter. <ul style="list-style-type: none"> • Source: This is the data server that hosts the source datastore. This server must have sufficient capacity to carry out a filter operation. • Staging Area: Transformation engine selected on the Definition tab on the interface. By default, this is the Target engine.

Target Datastore

An interface can have only one target datastore. There are two possibilities for this datastore:

- A **permanent datastore**, corresponding to a datastore that already exists in a model.
- A **temporary datastore**, that does not yet exist in a model, and which will thus be created dynamically by the interface, in either the work or data schema of the logical schema of the **Staging Area** specified on the **Definition** tab of the interface.

The target datastore, with the mapping for each column, is displayed on the right of the **Diagram** tab.

To edit the properties of the target datastore, click on the datastore title. The properties panel will appear at the bottom of the window.

Properties	Description
Name	Name of the target table. This field can only be modified for a temporary target datastore.
Update key	A set of columns which identifies which rows to update. It is used primarily in two situations: <ul style="list-style-type: none"> • In "incremental update" IKMs, the update key identifies the row in the target datastore to update for each row in the source datastore. • For flow control, it identifies which erroneous rows to delete from the flow, by matching them against rows in the errors table. <p>This field only appears for permanent datastores. To specify a custom update key, choose <Undefined> then check the Key checkbox for each individual column.</p>
Temporary Datastore Location	This field only appears for temporary datastores. The temporary datastore is always created on the staging area. More precisely, it is created on the physical schema corresponding to the staging area in the context specified for this temporary datastore. <ul style="list-style-type: none"> • Data Schema: The temporary datastore will be created in the normal (data) schema of this physical schema. • Work schema: The temporary datastore will be created in the work schema of this physical schema.
Context	If specified, overrides the execution context for the datastore. That is, it specifies the context in which the datastore will be sought (for permanent datastores) or created (for temporary datastores). <p>By default, Oracle Data Integrator chooses the optimization context (at design time) and the execution context during execution of the interface.</p>

Visual indicators

Certain properties of target datastore columns are displayed with visual indicators.

Indicator	Description
	Column is part of the update key.
n	Data type: the letter is generally the first letter of the column type. n is numeric, s is string and so forth.
	Mapping executed on the source.
	Mapping executed on the staging area.
	Mapping executed on the target.
	Warning associated with a mapping. Place the mouse cursor over the warning to see the problem.
	Error associated with a mapping. You cannot execute an interface with an error. Place the mouse cursor over the error to see the problem.
×	Mandatory column defined on the model, but Check Not Null is not set.
+	Mandatory column not defined on the model, but Check Not Null is set.
*	Mandatory column defined on the model, and Check Not Null is not set.

Mappings

Mapping is the set of the target datastore columns and the transformation rules linking them to the source datastore columns.

To edit the properties of a target column and its mapping, click on the target column name in the target datastore (to the right of the diagram tab), and the column and mapping properties will appear at the bottom of the composition panel.

Properties	Description
Active Mapping	This checkbox indicates that the Mapping must be taken into account when executing the interface.
Implementation	Text of the mapping. This is the transformation rule for determining the value to insert in the mapped column. It is usually written in SQL language. Columns of the entity-relation model of the composition panel can be added directly in this field. Note that from this field, you can also call up the expression editor.
Technical description	This field cannot be edited. It provides a version of the mapping with all column names replaced by their descriptions.
Business Rule	This field enables you to enter a business description and/or to provide notes and comments for the mapping.
Execute on	The transformation engine that performs the transformation. This engine must

	<p>have sufficient capacity (functions, SQL, etc) to effect these transformations.</p> <ul style="list-style-type: none"> • Source: This is the data server hosting the source datastore or datastores. For a source execution of a transformation that could be executed on several sources, a combo box allows you to choose the source on which the mapping is to be executed. For example, if, with two source tables on two Oracle servers, you enter the <code>SYSDATE</code> mapping, with execution on the source, the choice will allow you to state if you want the date of one Oracle server or the other. • Staging Area: Transformation engine selected on the Definition tab on the interface. By default, this is the Target engine. • Target: this is the data server hosting the target datastore. Mappings on the target cannot reference source columns, nor contain aggregate functions.
Update	<p>Defines insertion and update options, and the user specific options (User define) for the knowledge module.</p> <ul style="list-style-type: none"> • Insert: Shows that this mapping column participates in the insertion operations in the target table • Update: Shows that this mapping column participates in the update operations in the target table • UD1 - UD5: Shows that this mapping column participates in the operations linked to User Defined 1 to 5. Refer to the KM in use for more information.
Target column	This section describes the target column (only for a temporary target datastore).
Name	Name of the target column (modifiable for a temporary target datastore)
Datatype	Type of target column (modifiable for a temporary target datastore)
Length	Length of the target column (modifiable for a temporary target datastore)
Scale	Number of decimals in the target column (modifiable for a temporary target datastore)
key	Shows that the column participates in the update key for the interface. If the datastore is permanent, the update key must be left not selected on the properties level of the target datastore.
Check not Null	Shows that during the flow control, all the "not null" on this column must be verified.

Flow Tab

The **Flow** tab of an interface specifies the knowledge modules used to generate code from the interface.

On the Flow tab are displayed the different spaces (sources, staging area and target) on which the processing will take place, the datastores they contain, as well as the temporary tables that will be created there.

These spaces represent the physical servers of the information system.

A space on which a KM must be specified is shown with a red cross in the upper left corner.

You can specify the appropriate KM by clicking on the title of the window representing the physical server.

Properties of a LKM

Properties	Description
Name	Name of the selected source space.
Loading KM	The Loading Knowledge Module (LKM) used to extract the data from the selected space towards the staging area or target space. It is possible to select only from the LKM imported into the project that are appropriate for both technologies.
Options	Lists the LKM options with their names and values in a table. Options have a default value specified during creation of the KM.
Description of the LKM	Detailed description of the KM, written by the KM developer.

Properties of a IKM

Properties	Description
Distinct Rows	Allows you to specify that integration eliminates potential duplicates before inserting them in the target.
IKM	The Integration Knowledge Module (IKM) integrates the data in the selected space. Only IKM imported into the project and appropriate for this technology can be selected.
Options	Lists the LKM options with their names and values in a table. Options have a default value specified during creation of the KM.
Description of the IKM	Detailed description of the KM, written by the KM developer.

Controls Tab

The interface flow control strategy is defined on the **Controls** tab. This strategy is centered around a CKM (Check Knowledge Module).

Note: If it is not possible to perform a flow control for the interface (for instance, when the target is a temporary datastore), this tab only contains the fields related to the maximum number of errors allowed.

Properties	Description
Maximum number of errors allowed	This option applies to errors detected while extracting data from source file datastores, and to errors detected by the flow control process. When this number or percentage of errors is exceeded, the interface is goes into error state. If the % box is unchecked, the interface goes into error when one of the

following condition is true:

- The number of errors detected while extracting data from one file exceeds the number of errors allowed.
- The number of errors detected during the flow control exceeds the number of errors allowed.

If the % box is checked, the interface is in error when one of the following condition is true:

- The number of errors detected while extracting data from one file exceeds the percentage of errors allowed. The percentage is computed out of the number of lines extracted from the file.
- The number of errors detected during the flow control exceeds the percentage of errors allowed. This percentage is computed out of the number of records integrated to the target datastore (Insert + Update).

Note: When this field is left empty, then an infinite number of errors is allowed.

Note: The total number of errors can exceed the number of rows in the interface, as a line can infringe several constraints or contain several errors. In this case, the errors add up.

CKM **Defining the flow control strategy**

CKM The check knowledge module (CKM) used for checking the data. Only the project CKM appropriate for the technology (staging area) into which the check takes place can be selected.

Options Lists the CKM options with their names and values in a table.

Description of the CKM Detailed description of the KM, written by the KM developer.

Constraints **Showing the constraints to check**

Name of the constraint Name of the constraint concerned.

Value Shows if the given constraint is to be checked (yes/no)

Execution Parameters

When a interface is launched, execution parameters must be specified. As this information is not saved in the Repository, you must specify it each time.

Properties	Description
Context	Specifies the context in which the session is launched. Only the contexts you are granted rights to appear here.
Agent	The agent which will execute the interface. The interface can also be directly executed by Designer, by selecting Local (No Agent) .
Log level	Level of logging information to retain. All session tasks with a defined log level lower

than or equal to this value will be kept in the log when the session completes. However, if the interface ends abnormally, all tasks will be kept, regardless of this setting.

Scenarios Tab

This tab on an interface displays in a table view the scenarios generated for this component, as well as their schedules. You may double-click a scenario or a schedule to display its properties.

It is possible to create and delete scenarios by clicking the **Generate** and **Delete** buttons.

The popup menus for scenarios and schedules are also available from this tab.

Procedures

Procedures

A procedure is a set of commands that can be executed by an agent. These commands concern all technologies accessible by Oracle Data Integrator (OS, JDBC, JMS commands, etc).

The procedure properties are displayed in the **Definition** tab of the **Procedure** window of the **Designer** module. Execution for testing is carried out from the same window.

It is possible to encrypt a procedure to make it unreadable by its users, protecting a valuable development.

Procedures are located in a folder within a project. They are not accessible from another project.

Definition

Properties	Description
Name	Name of the procedure, as it appears in the graphical interface.
Description	Detailed description of the procedure.
Multi-connections	Shows if the procedure accesses paired data servers. If this box is checked, it becomes possible to exchange data loading commands between two data servers, of different technologies if necessary. For example, on procedure commands, you would be able to specify a command "Select ..." in one database and a command "Insert" into another database to transfer the whole result.
Source Technology	The source technology used by default on procedure commands. This information is only available if the procedure is a "Multi-connections"-type
Target Technology	The default technology to which the commands of this procedure will be addressed.

Procedure details

A procedure is made up of several commands. The detail tab draws up a list of these commands, which you can create, delete and arrange using the buttons below and beside the grid. To edit an existing command row, double-click on the row to be edited.

Options

A procedure may be parameterized when executed, by means of options. The **options** tab allows all options to be displayed, as well as their default values. The options values specified in this tab will only be used during execution of the procedure.

Execution of the procedure

The execution tab allows the procedure to be tested by executing it on a context and an agent.

Scenarios

This tab displays in a table view the scenarios generated for this component, as well as their schedules. You may double-click a scenario or a schedule to display its properties.

It is possible to create and delete scenarios by clicking the **Generate** and **Delete** buttons.

The popup menus for scenarios and schedules are also available from this tab.

Procedure Commands

A procedure command is the unit of execution of a procedure.

Definition

This tab allows the command row to be defined.

Properties	Description
Name	Name of the command, as it appears in the graphical interface.
Log Counter	Shows which counter (Insert, Update, Delete or Errors) will record the number of rows processed by this command.
Log level	Level of importance of the command, for consulting the execution log according to this level.
Ignore Errors	Shows that the procedure is not interrupted in case of an invalid return code. If this box is checked, the procedure command will go into "warning" instead of "error", and the procedure will not be stopped.
Command	The command launched on the data server. Two similar tabs (Command Source and Command Destination) appear in the case of a multi-connection procedure.
Technology	Technology on which the command will be executed. Note: To use Oracle Data Integrator Tools (commands) in KM procedure commands or procedures commands, you must set the technology to <code>Sunopsis API</code> . To use OS commands or to execute external programs, you must set the

	technology to Operating System.
Transaction Isolation	<p>The transaction isolation level for Select orders. The isolation levels shown are those of the SQL ISO. They are not supported by all data servers.</p> <p>The levels of isolation are:</p> <ul style="list-style-type: none">• Default: The transaction keeps the connection's isolation level.• Read Uncommitted: The transaction can read data not committed by another transaction.• Read Committed: The transaction can only read data committed by other transactions (in general, this is the default mode of many data servers).• Repeatable Read: The transaction is certain to read the same information if it executes the same SQL query several times, even if the rows have been modified and committed by another transaction in the meantime.• Serializable: The transaction is certain to read the same information if it executes the same SQL query several times, even if the rows have been modified, deleted or created and committed by another transaction in the meantime.
Context	Execution context of the query. If "Execution context" is left, the command will be executed in the execution context chosen on launching or on the step.
Schema	Logical schema for execution. This logical schema, linked to the context, allows the physical execution schema to be defined.
Transaction	You can execute commands on several concurrent transactions (numbered from 0 to 9) or work off-transaction by choosing the option "Autocommit". The latter may be more efficient.
Commit	<p>If your command is being executed in a transaction (numbered 0 to 9), you can decide to continue or to commit the current transaction according to the following modes:</p> <ul style="list-style-type: none">• No Commit: The transaction is not committed. In this case, it can be committed in a later command. If a session ends normally, all transactions are committed by default.• Commit: The transaction is committed.• Commit 1000 rows: Oracle Data Integrator commits every 1000 records processed. This choice is only possible on a loading procedure command, that is, one containing a source command returning a result set and a destination command that carries out Inserts.
Command	<p>The text of the command expressed in a native language or in a standard language (such as SQL, PL/SQL, Transact-SQL, shell, etc). You can use the expression editor, accessible through the button to the right of the text.</p> <p>Warning: If the command seems to be made of random characters, then the procedure is encrypted. For more details about encryption and decryption, please read Encrypt a KM or a Procedure. Contact also the procedure's provider.</p>

Options

This tab allows you to specify whether the procedure command is to be executed or not, according to the value of a procedure option.

Properties	Description
Always execute	The procedure command is executed regardless of the options values.
Options	This chart shows all options. The checkbox preceding each option shows that the procedure command is executed if the (checkbox-type) option has the value "yes".

Options

An option is attached to either a Procedure or a Knowledge Module (KM), and has two functions:

- Defining textual variables (**text** type or **value** type) that are internal to the procedure or to the KM, and which will be specified when it is used.
- Defining **checkbox** type options, to determine whether or not a procedure or a KM task should be executed.

For a KM, for example, the options will be "UPDATE" (to determine if the actions relating to the records update should be taken into account), "TRUNCATE" (should the target data be unloaded or not?), or "INSERT" (should the data be inserted into the target?).

Definition

Properties	Description
Name	Name of the option as it appears in the graphical interface.
Type	Type of option. There are three types: <ul style="list-style-type: none"> • Check Box: The option is Boolean-type (Yes = 1/No = 0). These are the only options used for procedures and KMs to determine if such tasks should be executed or not. • Default Value: It is an alphanumerical-type option. Its maximum size is 250 characters. • Text: It is an alphanumerical-type option. Its size is not limited. Accessing this type of option is slower than for a value-type option.
Description	Short description of the option. For Check Box options, this is displayed on the Command window where you select which options will trigger the execution of the command.
Position	Determines the order of appearance of the option when the procedure or KM options list is displayed.
Help	Descriptive help on the option. For KMs, is displayed in the properties pane when the KM is selected in an interface.
Default value	Value that the option will take, if no value has been specified by the user of the procedure or the KM.

Variables

Variables can be used in any expression (SQL or others), as well as within the metadata of the repository. A variable is resolved when the command containing it is executed by the agent or the graphical interface.

The properties of the variable are displayed on the **Definition** and **Refresh** tabs in the **Variable** window of the **Designer** module. The global variables are located in the tree on the same level as the projects, while the project variables are located within their project.

Definition

Properties	Description
Name	Name of the variable, in the form it will be used. This name should not contain characters that could be interpreted as word separators (blanks, etc) by the technologies the variable will be used on. Variable names are case-sensitive. That is, "YEAR" and "year" are considered to be two different variables.
Variable scope	The validity range of the variable, either "Global" or "Project". A variable can be valid for all projects (a global variable), or for the current project only. Oracle Data Integrator examines "Project"-level variables first before evaluating "Global"-level variables. Thus, if a variable YEAR exists at both project level and global level, only the project-level value can be evaluated.
Description	Detailed description of the variable
Datatype	Type of variable: Alphanumeric (255 characters), Date, Numeric (Maximum 10 digits) or Text (unlimited length).
Action	This parameter shows the length of time the value of a variable is kept for: <ul style="list-style-type: none"> • Non-persistent: The value of the variable is kept in memory for a whole session. • Last value: Oracle Data Integrator stores in its repository the latest value held by the variable. • Historize: Oracle Data Integrator keeps and history of all the values held by this variable.
Default Value	The value assigned to the variable by default.

Refresh

Refresh a variable allows a new value based on the result of an SQL-type query to be assigned to it. All expressions using this variable will subsequently be executed with the new value until a new refresh is done.

Properties	Description
Schema	Logical schema, for identifying the connection on which the SQL query will be executed.
Select	Select-type SQL query or any command (execution of a stored procedure) allowing

query	<p>a result array (Resultset) to be returned for a row and a column.</p> <p>For example: <code>Select max(order_no) from crm.order</code></p> <p>Caution: Table names should be specified in full (catalog, schema, etc), as connection parameters do not allow positioning on the right physical schema.</p>
Refresh	<p>Click on the refresh button to calculate the last value of the variable while executing the query entered above. A window is displayed for entering the context and the execution agent. You check the execution in the log.</p> <p>Note: If this variable is not persistent, you will not be able to display the value calculated.</p>

History

This window displays the history of the values of the variable with the context for variables with the "log" action, or the last value for a variable with the "last value" action.

Note: Each value is attached to the context in which the session was executed.

- The button allows to delete a line of the history.
- The **Context Filter** field lets you filter the entries for a given context.

Scenarios

This tab displays in a table view the scenarios generated for this component, as well as their schedules. You may double-click a scenario or a schedule to display its properties.

It is possible to create and delete scenarios by clicking the **Generate** and **Delete** buttons.

The popup menus for scenarios and schedules are also available from this tab.

Note: Scenarios generated for variables contain a single step performing a refresh operation for the variable.

Sequences

A sequence is a variable that increments itself each time it is used. Between two uses, the value can be stored in the repository or managed within an external RDBMS table.

Oracle Data Integrator supports two types of sequences:

- **Standard sequences**, whose last value is stored in the Repository.
- **Specific sequences**, whose last value is stored in an RDBMS table cell. Oracle Data Integrator undertakes to read the value, to lock the row (for concurrent updates) and to update the row after the last increment.

General Properties

Properties	Description
Name	Name of the sequence, in the form it will be used. This name should not contain characters that could be interpreted as word separators (blanks, etc) by the

	technologies the sequence will be used on. Upper and lower case are not distinguished during interpretation. For example: ORDER_NO
Scope	The validity range of the sequence, either "Global" or "Project". A sequence can be valid for all projects (a global sequence), or only for the current project. Oracle Data Integrator examines "Project"-level sequences first, before evaluating "Global"-level sequences. Thus, if a sequence ORDER_NO exists at both project level and global level, only the project-level sequence value can be evaluated.
Increment	Value of the increment. This value can be positive or negative.
Standard sequence	
Position	Last value allocated for a standard sequence.
Specific sequence	
Schema	Logical schema containing the sequences table
Table	Table containing the sequence value
Column	Name of the column containing the sequence value.
Filter to retrieve a single row	When the latter table contains more than one row, a filter should be completed in order to locate this row in the table. This filter picks up the SQL syntax of the data server. For example: CODE_TAB = '3'

User Functions

User functions enable to define customized functions that can be used in interfaces or procedures. These functions are implemented in one or more technologies.

Users Function Window

Definition

This tab lets you declare the generic properties for the user function

Properties	Description
Name	Name of the user function. Example: NullValue
Group	Group of the user function. If you type a group name that does not exist, a new group will be created with this group name when the function is saved.
Syntax	Syntax of the user function that will appear in the expression editor; The arguments of the function must be specified in this syntax. The function's syntax is expressed in one of the following ways: <code>function_name(\$ (arg_name) arg_type, \$ (arg_name) arg_type,</code>

```

... )
function_name($(arg_name), $(arg_name), ... )
function_name()
function_name

```

`arg_type` is used to give a type to the arguments and is not mandatory. It may be equal to `s` (string), `n` (numeric) or `d` (date).

For instance, to declare the function `NullValue`, we will use:

```
NullValue($(variable), $(default))
```

Description Detailed description of the function.

Implementations

This tab lets you define the implementations of the function for each technology, or for groups of technologies.

Properties	Description
Implementations	Technologies or technologies groups for which the function is implemented.
Add - Edit - Delete	These buttons enable to delete an implementation, to create a new implementation, or to change an existing one, in the Implementation window.
Technologies with no implementations	Technologies for which the function is not implemented.

Implementation Window

this window appears if you change or create an implementation.

Properties	Description
Implementation syntax	<p>Syntax of the function, in the language of the selected technologie(s). The arguments of the functions are used with the syntax <code>\$(arg_name)</code>.</p> <p>For instance, the two implementations of the function <code>NullValue(\$(variable), \$(default))</code> will be:</p> <p>Oracle : <code>nvl(\$(variable), \$(default))</code></p> <p>Microsoft SQL Server : <code>case when \$(variable) is null then \$(default) else \$(variable) end</code></p>
Associated technologies	List of the technologies associated to this implementation. There can be only one implementation per technology.
Automatically include new technologies	If this box is checked, the new technologies are automatically added to this implementation. Only one implementation may have this box checked.

Knowledge Modules

Knowledge Modules

Oracle Data Integrator uses knowledge modules to define methods related to a given technology. These modules enable processes generation for the technology, dedicated to a precise function.

It is possible to encrypt a Knowledge Module (KM) to make it unreadable by its users, protecting valuable development.

LKM - Loading

The LKM - Loading Knowledge Modules - load (or extract data) from one data server to another.

They are named as follows:

- **LKM <source server technology> to <target server technology> (<module specificity>)**

Note: Some KM use a generic ISO (SQL or JMS) to reach the data servers.(for source, target or staging area). The KM indicated with an ISO technology work with data servers supporting the standards.

Note: The specificity of the knowledge module indicates often a method or feature specific to the technologies managed by the module, allowing a faster data processing, and which is used actively in the module.

JKM - Journalizing

The JKM - Journalizing Knowledge Modules - setup journalizing on a datastore in a data model.

They are named as follows:

- **JKM <technology to journalize>**

Note: Journalizing uses triggers and views. Therefore technologies that do not support these functions generally do not have an associated JKM.

IKM - Integration

The IKM - Integration Knowledge Modules - integrate data in a target data server from a Staging Area. Frequently, the Staging Area and target space are on the same data server. The IKM can manage options such as data insertion (**Append**), or update (**Incremental Update**), etc.

They are named as follows:

- **IKM <Staging Area data server technology> to <target data server technology> <option managed in the module> (<module specificity>)**
- **IKM <Staging Area and target data server technology> <option managed in the module> (<module specificity>)**

Note: The option managed by the module frequently indicate the more complex option supported in this knowledge module. For instance, a module **Incremental Update** (update) should also be able to manage the option **append**, which is data insertion, and suppression of data in the target..

CKM - Check

The CKM - Check Knowledge Modules - manage the data quality and cleansing, on static data and on data flows.

They are named as follows:

- **CKM <Checked Technology> (<module specificity>)**

RKM - Reverse-engineering

The RKM - Reverse Knowledge Module - reverse-engineer a data model stored in a given technology. As default, the **standard reverse** allows to reverse all the information and meta-data in most models. It is required for certain technologies (files for instance) to use a **personalized reverse** and therefore a RKM. They are named specifically.

SKM - Services

SKM (Service Knowledge Modules) are used to generate the code required for creating data services. This code (typically Java) can be automatically compiled and deployed to a web service container.

KM window

Definition tab

For a IKM, RKM, JKM and CKM, the definition tab appears with the following fields.

Properties	Description
Name	Name of the KM, as it appears in the graphical interface.
Type	Type of KM
Consistent Set Journalizing / Simple Journalizing	This field appears for a journalizing KM only, to define which journalizing mode is managed by this knowledge module.
Description	Detailed description of the KM.
Default KM for this couple of technologies	Shows that the KM will be selected by default for the data servers with technologies shown in Source technology and Target Technology
Multi-connections	Shows if the KM has access to paired data servers. If this box is checked, it becomes possible to exchange data loading commands between two data servers, of different technologies if necessary. For example, on procedure commands, you would be able to specify a command "Select" in one database and a command "Insert into another database " to transfer the whole result.
Source Technology	Source technology used by default on the procedure commands. This information is only available if the procedure is "Multi-connections"-type.
Target Technology	The default technology to which the commands of this procedure will be addressed.

For a SKM, the tab appears with the following fields:

Properties	Description
Name	Name of the KM, as it appears in the graphical interface.
Command Text	Code of the SKM
Description	Detailed description of the KM.

There is no **Details** and **Options** tabs for an SKM.

Details tab

A KM is made up of several procedure commands . The detail tab lists these command, which you can create, delete and organize using the buttons beside the grid. To edit an existing procedure command, double-click on the row to be edited.

Options tab

A KM can be parameterized when executed by means of options. The **options** tab allows all options to be displayed, as well as their default values. The options values specified on this tab will only be used during execution of the KM only.

KM Commands

A KM command is the unit of execution of a knowledge module.

Definition

This tab allows the command row to be defined.

Properties	Description
Name	Name of the command, as it appears in the graphical interface.
Log Counter	Shows which counter (Insert, Update, Delete or Errors) will record the number of rows processed by this command.
Log level	Level of importance of the command, for consulting the execution log according to this level.
Ignore Errors	Shows that the KM is not interrupted in the event of an invalid return code. If this box is checked, the command will go into "warning" instead of "error", and the KM will not be stopped.
Journalizing (IKM)	These options only appear for a IKM command and define the behavior of the command when processing data from a journalized datastore.
Journalized table in the Staging Area	Execute command if the journalized table is in the Staging Area.
Journalized table source of current interface	Execute command if the journalized table is source of the current interface.

Journalizing (LKM)	These options only appear for a LKM command and define the behavior of the command when processing data from a journalized datastore.
Table Journalized in current space.	Execute command if the current source set contains a journalized table.
Journalizing (JKM)	These options only appear for a JKM command. They define the behavior of the command when setting up journalizing.
Model Creation	Execute command when starting a journal for a model.
Model Deletion	Execute command when stopping a journal for a model.
Table Creation	Execute command when starting a journal for a datastore.
Table Deletion	Execute command when stopping a journal for a datastore.
Subscribe	Execute command when creating a subscription.
Unsubscribe	Execute command when deleting a subscription.
Consumption (JKM)	These options only appear for a JKM command. They define the behavior of the command when consuming from the journals.
Extend Window	Execute command when performing an Extend Window operation on the CDC set.
Purge Journal	Execute command when performing a Purge Journal operation on the CDC set.
Lock Subscribers	Execute command when performing a Lock Subscriber operation on the CDC set.
Unlock Subscribers	Execute command when performing an Unlock Subscriber operation on the CDC set.
Repeat for each table (JKM)	This option group defines if the command should be iterated for all the datastores when processing a CDC related operation on a group of datastores (model or CDC set) .
No	The command is not repeated.
Ascending Order	The command is repeated for each datastore. The repetition is performed in the ascending value of datastore Order in the CDC set.
Descending	The command is repeated for each datastore. The repetition is performed in the descending value of datastore Order in the CDC set.
Checks (CKM)	These options only appear for a CKM command. They indicate the types of action for which this command is required and must be executed.
Primary Key	Execute command for a Primary Key check.
Alternate Key	Execute command for a Alternate Key check.

Join	Execute command for a reference or join check.
Condition	Execute command for a Condition check.
Mandatory	Execute command for a mandatory column (Not Null) check.
Remove Errors	Execute command for error removal.
Loading (LKM)	These options only appear for a LKM command. They indicate when this command is required and must be executed.
Pre integration (IKM)	Execute command before integration.
Post integration (IKM)	Execute command after integration.
Command	The command launched on the data server. Two similar tabs appear (Source Command and Destination Command) when dealing with a multi-connections KM.
Technology	Technology on which the command will be executed. Note: To use Oracle Data Integrator tools in KM commands or procedures commands, you must set the technology to <code>Sunopsis API</code> . To use OS commands or to execute external programs, you must set the technology to <code>Operating System</code> .
Transaction Isolation	The transaction isolation level for Select orders. The isolation levels shown are those of the SQL ISO. They are not supported by all data servers. The levels of isolation are: <ul style="list-style-type: none"> • Read Uncommitted: The transaction can read data not committed by another transaction. • Read Committed: The transaction can only read data committed by other transactions (in general, this is the default mode of many data servers). • Repeatable Read: The transaction is certain to read the same information if it executes the same SQL query several times, even if the rows have been modified and committed by another transaction in the meantime. • Serializable: The transaction is certain to read the same information if it executes the same SQL query several times, even if the rows have been modified, deleted or created and committed by another transaction in the meantime.
Context	Execution context of the query. If "Execution context" is left, the command will be executed in the execution context chosen on launching or on the step.
Schema	Logical schema for execution. This logical schema, linked to the context, allows the physical execution schema to be defined.
Transaction	You can execute commands on several concurrent transactions

	(numbered from 0 to 9) or work off-transaction by choosing the option "Autocommit". The latter may be more efficient.
Commit	<p>If your command is being executed in a transaction (numbered 0 to 9), you can decide to continue or to commit the current transaction according to the following modes:</p> <ul style="list-style-type: none"> • No Commit: The transaction is not committed. In this case, it can be committed in a later command. If a session ends normally, all transactions are committed by default. • Commit: The transaction is committed. • Commit 1000 rows: Oracle Data Integrator commits every 1000 records processed. This choice is only possible on a loading command, that is, one containing a source command returning a result set and a destination command that carries out Inserts.
Command	<p>The text of the command expressed in a native or in a standard language (SQL, PL/SQL, Transact-SQL, shell, etc). You can use the expression editor, accessible through the button to the right of the text.</p> <p>Warning: If the command seems to be made of random characters, then the KM is encrypted. For more details about encryption and decryption, please read Encrypt a KM or a Procedure. Contact also the KM's provider.</p>

Options

This tab allows you to specify whether or not the command is executed according to the value of a KM option.

Properties	Description
Always execute	The command is executed regardless of the options values.
Options	This chart shows all the KM options. The checkbox preceding each option shows that the command is executed if the (checkbox-type) option has the value "yes".

Options

An option is attached to either a Procedure or a Knowledge Module (KM), and has two functions:

- Defining textual variables (**text** type or **value** type) that are internal to the procedure or to the KM, and which will be specified when it is used.
- Defining **checkbox** type options, to determine whether or not a procedure or a KM task should be executed.

For a KM, for example, the options will be "UPDATE" (to determine if the actions relating to the records update should be taken into account), "TRUNCATE" (should the target data be unloaded or not?), or "INSERT" (should the data be inserted into the target?).

Definition

Properties	Description
Name	Name of the option as it appears in the graphical interface.
Type	Type of option. There are three types: <ul style="list-style-type: none"> • Check Box: The option is Boolean-type (Yes = 1/No = 0). These are the only options used for procedures and KMs to determine if such tasks should be executed or not. • Default Value: It is an alphanumeric-type option. Its maximum size is 250 characters. • Text: It is an alphanumeric-type option. Its size is not limited. Accessing this type of option is slower than for a value-type option.
Description	Short description of the option. For Check Box options, this is displayed on the Command window where you select which options will trigger the execution of the command.
Position	Determines the order of appearance of the option when the procedure or KM options list is displayed.
Help	Descriptive help on the option. For KMs, is displayed in the properties pane when the KM is selected in an interface.
Default value	Value that the option will take, if no value has been specified by the user of the procedure or the KM.

Marker Groups

A **marker group** is a set of flags that can be applied to elements of a project or a model to organize the developments.

Definition

Properties	Definition
Group Name	Name of the marker group. This name appears in the user interface.
Group Code	Code of the marker group. This code is used to access a marker from the Oracle Data Integrator tools and API.
Display Position	Position of the marker relatively to the marked object icon in the tree view. If set to Never, the marker appears only in the Marker tab of the object window, and is not visible in the tree view.
Order	Relative position of markers from this group among all markers. The markers with the lowest order value appears first in the lists and the tree view.
Attributes	Multi-state marker groups allow several markers from the same group to be applied to the same object. If this box is unchecked, only one marker from this group may be applied at a time to an object. A priority marker group would be mono-state.

Auto-Increment marker groups do not allow multiples markers on the same object. In addition, when an auto-increment marker is clicked in the tree view, it switches to the next marker from the group. A progress flag would be auto-increment.

Markers

Icon	Icon for the marker. Icons are allowed for string markers only. If a marker stores date or a number, the icon should be set to <none>. Markers with no icon do not appear in the tree view.
Name	Name of the marker. This name appears in the user interface.
Code	Marker code. This code is used to access a marker from the Oracle Data Integrator tools and API.
Type	Type of the marker. This field reflects the type of data the marker stores: <ul style="list-style-type: none"> ▪ String. This type is also used for icons. ▪ Number ▪ Date <p>Number and dates do not appear in the tree view and can be changed through the Marker tab for the element window.</p>
Active	If this box is checked, the marker is displayed for this group. Otherwise, it is always hidden.
Tooltip	Check this box to have a tooltip on the marker's icon.

Models

What is a Model?

A **model** is a set of **datastores** corresponding to data structures contained in a physical schema. Models can be organized into **model folders**.

- See Creating and Reverse-engineering a Model

Model Folders

A **model folder** is an object which groups models together. For example, you can group all models based on a particular technology, located at a particular site or used in a particular project.

Sub-models

A **sub-model** is an object used to organize and classify the datastores of a model into a hierarchical structure. The root of the structure is the model itself.

Reverse-engineering

A model is created without any datastores in it. **Reverse-engineering** a model allows you to automatically retrieve data structures to define the model's datastores in Oracle Data Integrator. There are two different modes:

- **Standard** reverse-engineering uses standard JDBC features to request the metadata.
- **Customized** reverse-engineering uses a technology-specific Reverse Knowledge Module (RKM) to retrieve the metadata, using a method specific to the given technology.

Datastore

A **datastore** describes data as a table structure. Datastores are composed of **columns**.

Datastores are defined in Oracle Data Integrator in a relational model. Therefore, it is possible to attach the following elements to a datastore:

Keys

A **key** is a set of datastore columns that enables each datastore row to be uniquely identified. If it is also an index, it may also allow row access to be optimized. Some drivers retrieve key descriptions during the reverse-engineering process. It is also possible to define keys directly in the repository.

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

A **condition** or a **filter** is a WHERE-type SQL expression attached to a datastore based on any RDBMS that supports SQL. They serve to filter or check data in the datastore.

Journalizing

Journalizing keeps track of changes to data. Journalizing is used in Oracle Data Integrator to eliminate the transfer of unchanged data. This feature has many uses, such as in data synchronization and replication.

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

Models

Definition

Property	Description
Name	Name of the model used in the user interface.
Code	Unique code for this model.

Technology	Technology of the data model. A data model being linked to a single technology, this is a physical data model whose datatypes are native to this technology.
Display the Metadata changes in the Model tree	<p>If this option is checked, all elements of the model that have been deleted or changed in the database since the last reverse engineering operation are flagged in the tree.</p> <p>Note: Checking this option may cause reduce performance when browsing the model in the tree.</p> <p>Important: This options only works for models using Standard reverse-engineering</p>
Logical schema	The logical schema of the topology that this model is attached to.
Action Group	Action group used to generate DDL scripts for this model. If no action group is selected the <i><Generic Action></i> group is used. For more information see Common Format Designer.
Default Folder	Default project folder into which DDL scripts for this model will be generated. For more information see Common Format Designer.
Description	Detailed description of the model.

Reverse

Reverse engineering consists of recovering the metadata (the description of the data structure) of an application from the dictionary of the technology which stores this information, and storing it in the Oracle Data Integrator repository.

Property	Description
Type of Reverse	<ul style="list-style-type: none"> A Standard reverse uses the capacities of the driver (JDBC, ODBC, etc) to retrieve the metadata and store them in the Repository A Customized reverse uses a procedure, the Reverse Knowledge Module (RKM), to extract the metadata for a specific type of application and to store them in the Repository.
Context	The context in which the reverse is executed. The combination of the context and the logical schema allows Oracle Data Integrator to connect to the required data server to access the metadata.
Logical agent	This is the agent used in customized reverse engineering mode.
Types of objects to reverse-engineer	The types of the objects that should be taken into account by the reverse-engineering process. This list may include tables, views, queues, system tables, table aliases and synonyms.
Mask	The mask pre-selects the objects to reverse. This mask uses the LIKE syntax of the SQL language, that is: the % symbol stands for zero or more characters, and the _ symbol stands for one character.
Characters to remove for the	The characters to delete in order to derive the alias. Each datastore has an alias to be used in transformation, filter and check expressions. This is a short

Table Alias	name made up, by default, of the first three characters of the datastore name. For applications containing many tables beginning with the same prefix, you can ignore the prefix by completing the section "Characters to delete". For example, the value 'DWG_' allows you to derive the aliases PRO, CUS, ITE for the tables named DWG_PROD, DWG_CUS, DW_ITEM.
Table Alias maximum length	Maximum length of a generated alias for this model. Characters after this length are truncated.

Selective reverse

Selective reverse allows you, for a **Standard reverse**, to select the datastores to reverse from a list that takes into account the mask and the type of object to reverse that have been defined on the **Reverse** tab.

Property	Description
Selective Reverse	Specifies that the next reverse engineering operation will be selective. That is, it will use the parameters on this tab.
Existing Datastores	Specifies that you wish to re-reverse-engineer datastores already present in the selected model. The metadata for these objects will be updated in the repository. Any changes are noted in the execution log.
New Datastores	Specifies that you wish to reverse-engineer datastores that are not present in the current model.
Objects to Reverse	Allows you to individually specify which datastores to reverse-engineer. The list displayed takes into account the Mask and parameters. You can further reduce this list by un-checking the objects you do not wish to reverse. This list is not available in Custom mode.

Control (data quality audit)

The data quality audit (also known as static control) checks that all the data in a model meets the integrity constraints defined on the datastores. These constraints include references, primary keys, alternate keys, conditions and mandatory columns. A Check Knowledge Module (CKM) is always used.

Property	Description
Check Knowledge Module	The strategy used for data quality control. Only a CKM valid for the technology of the model can be selected here. To be available, the CKM must have been imported into at least one project
Options	Parameters specified by the CKM.
Description	A detailed description of the CKM.
Execute	Launches a session to perform the data quality agent. You must select an agent and a context.

Journalizing

Journalizing tracks data changes (Insert/Delete/Update) in the datastores of a model. This tab allows you to define and configure the journalizing method used for this model.

Property	Description
Journalizing Mode	Mode used for journalizing this model. See Changed Data Capture for more information.
Journalizing Knowledge Module	The strategy used for data journalizing. Only a JKM valid for the technology of the model can be selected here. To be available, the JKM must have been imported into at least one project
Options	Parameters specified by the JKM.
Description	A detailed description of the JKM
Subscriber List	List of the subscribers currently tracking changes in this model.

Journalized Tables

This table lists the datastores from the model included in the CDC. It lets you organize the datastores in the CDC set. See Changed Data Capture for more information.

Property	Description
Order	Position of the journalized datastore within the CDC set. This order usually depends on the foreign key relations. A referenced table should have a lower order number than all its referencing tables.
Table Name	Name of the journalized datastore.

Services

This tab contains the configuration for generating and deploying data services for this model. See Setting Up Data Services for more information.

Property	Description
Application server	Logical schema corresponding to the application server (web services container) into which the generated web services will be deployed.
Namespace	Namespace for your web services. It is used to generate the WSDL.
Package name	Name of the java package generated to contain all the web services. Generally, this is of the form <code>com.<company name>.<project name></code>
Name of the datasource	Name of the datasource that you defined in the application server, and that corresponds to the server containing your model's data. This name should be prefixed by <code>java:/comp/env/</code> .
Name of the Data Service	Name of the Data Service providing (CDC related) features at model-level.
Service KM	Service Knowledge Module used to generate the web service. Only KMs

imported in your projects appear in this list.

Deployed Datastores List of datastore that will be deployed as Data Services. For each datastore, you can give a **Data Service Name** and the name of the **Published Entity**.

Model Folders

A folder is a group of models. Folders and sub-folders allow these models to be grouped and organized according to criteria specific to the project. Sub-folders can be created to an unlimited number of levels.

Note: To create a sub-folder, simply create a folder, then drag and drop it into the parent folder.

Definition

Properties	Description
Name	Name of the folder or sub-folder, as it appears in the user interface.
Description	Detailed description of the folder.

Sub-models

A sub-model is a group of functionally homogeneous datastores within a model. The datastores of a model can be inserted into a sub-model using drag and drop, or by automatic distribution.

Definition

Property	Description
Name	Name of the sub-model used in the graphical interface of the Designer module.
Code	Unique code of the sub-model

Control

With this procedure, an immediate static control of the data stored in the sub-model Datastores can be executed. This procedure is executed on a context specified when the check is launched. This check uses the check knowledge module shown at the model level.

Distribution

Distribution allows you to define an automatic distribution of the datastores in your sub-models. Datastores, depending on the **datastores distribution rule** for each sub-model, are compared to the **automatic assignment mask**. If they match this pattern, then they are moved into this model. There are two methods to classify:

- By clicking on the button **Distribution** of a sub-model, the current rule is applied to the datastores.
- At the end of a reverse, all rules are applied in the **mask application order after a reverse**.

Property	Description
Datastores Distribution rule	<p>Determines which datastores will be taken in account and compared to the automatic assignment mask:</p> <p>No automatic distribution: No datastore is taken in account.</p> <p>Automatic Distribution of all Datastores not classified...: Datastores located in the root model in the sub-model tree are taken in account.</p> <p>Automatic Distribution of all Datastores: All datastores in the model (and sub-models) are taken in account.</p>
Automatic Assignment Mask	<p>Pattern the datastores names must respect to be classified in this sub-model.</p>
Mask application order after a reverse.	<p>At the end of a reverse, all rules are applied in the mask application order after a reverse.</p> <p>Consequently, a rule with a high order on all datastores will have precedence. A rule with a high order on non classified datastores will apply only on datastores ignored by the other rules' patterns.</p> <p>At the end of the reverse, new datastores are considered as non classified. Those already classified in a sub-model stay attached to their sub-model.</p>

Datastore

Datastores

A datastore is a structure that allows data to be stored. It can be a table, a file, a message queue or any other data structure accessible by middleware used by Oracle Data Integrator (JDBC/ODBC, JMS or JNDI).

Definition

Property	Definition
Name	<p>Name of the Datastore. This is the name that appears in the trees and that is used to reference the datastore from a project (interface, etc). This name should be different from the resource name, if the latter is unknown or not very clear.</p> <p>For example, if the model contains tables named C3556F or C5677D, you can give them more explicit names, such as CUSTOMER or ORDER, leaving C3556F and C5677D as resource names.</p>
Datastore Type	<p>The type of object the datastore represents.</p>
OLAP Type	<p>Table type in a multidimensional model for Online Analytic Processing (OLAP):</p> <ul style="list-style-type: none"> ▪ Fact table ▪ Dimension

- Slowly Changing Dimension

Resource Name	Name of the object in the form recognized by the data server which stores it. This may be a table name or a file name.
Alias	This is a short name used in check and filter expressions. This name must be used systematically before each column name to allow cross-references to be compiled. For example, for the table CUSTOMER, the alias CUS can be given. In this case, the expression of the condition CLIENT_TYPE could be CUS.TYPE in ('A', 'S', 'D'). The alias name is not necessarily unique for all tables in a certain model, its only purpose is to make writing expressions easier and to make managing cross-references possible.
Description	Detailed description of the datastore.
Number of rows	Number of datastore rows obtained by clicking on refresh .

Files

This tab appears only for datastores attached to a file technology.

Property	Definition
File Format	Format of the file datastore: <ul style="list-style-type: none"> • Delimited : The fields of a line are separated by a record separator. • Fixed : The fields of a line are not separated, but their length is fixed.
Header (number of lines)	Number of lines at the beginning of the file that are not data. This lines are ignored.
Record separator	One or several characters separating lines (or records) in the file: <ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • Tabulation • Space • Other : Free text you can input as characters or hexadecimal codes.
Text delimiter	Pair of characters delimiting a STRING field.
Decimal separator	Character separating the integer and decimal part of a NUMERIC.

Column

This tab enables an overall view of the datastore columns to be displayed, and to be added and deleted. The buttons **Reverse** and **Reverse COBOL Copybook** allow you to reverse-engineer the columns of a file in the following conditions:

- **Reverse:** This reverse-engineering launches a **standard** reverse for the current datastore.
- **Reverse:** This reverse-engineering is also possible for a **delimited** file. The column names are retrieved on the header line of the file, or are automatically generated. The length of the columns are set with default values, and must be redefined manually. If the file is a fixed file, this button opens a Wizard for defining the columns.
- **Reverse COBOL Copybook:** This reverse-engineering is possible for a **Fixed** file when you have its description file as a COBOL Copybook format. For more information, see Reverse a COBOL Copybook.

Automatic adjustment: With this option selected for a fixed file only, starting positions are automatically adjusted as a function of column widths, to avoid gaps and overlaps. For example, if you increase the width of the first column from 5 to 16, then the starting position of every other column will be increased by 11 characters. It is not possible to manually specify the starting position of a column when Automatic adjustment is enabled.

Control

This procedure allows you to execute a static control of the data stored in the Datastore. This procedure is executed on a context specified when the control is launched. This check uses the knowledge module shown on the model level.

Journalizing

This tab displays the status of journalizing for the datastore and list of subscribers tracking changes on this datastore, with their subscription creation date.

Services

This tab contains the configuration for generating and deploying data services for this datastore. See Setting Up Data Services for more information.

Property	Description
Deploy as a Data Service	Check this box if you want this datastore to be deployed as a data service.
Data Service Name	Name of the web service generated for this datastore.
Published entity	Name that is used to generate all operations names for the Data Service. For example, if you specify <code>customer</code> as the published entity, the generated operations will be named <code>addcustomer</code> , <code>getcustomer</code> , etc.

Columns

Datastores describe data accessed by Oracle Data Integrator through two-dimensional tabular structures. Oracle Data Integrator contains the metadata related to the columns. The data contained in the applications is stored in rows.

Definition

Property	Description
----------	-------------

Name	Name of the column, as recognized by the data server.
Datatype	The datatype, corresponding to a type recognized by the data server.
Short description	Descriptive label of the column. This label makes the graphical interface easier to read if the column names are not explicit.
Physical format	Description of the physical storage format of the data
Order	Order number or physical rank of the column in the datastore.
Start	Physical position of the column. This information is to be given for fixed format files. The first position is position 1.
Length	<p>The physical length in number of bytes. A column with start position '1' and length '4' occupies bytes 1,2,3,4 in each row.</p> <p>Note:For delimited files, the maximum length taken by the column should be shown. If the maximum length is not known beforehand, try to put a length that is greater than the maximum length. This length is used to reserve sufficient memory space for reading the information.</p>
Record codes	<p>The list of record codes allows files with several record patterns to be processed. For example, for a file INVOICE containing different headers and invoices rows marked by a record code CODE_REC with the value 'HEAD' or 'LINE', you must create a datastore for each of these conceptual entities. Specify HEAD in the list of record codes for the column CODE_REC of the datastore INVOICE, and LINE in the corresponding column for the datastore INVOICE_LINE.</p> <p>You can specify several values separated by the character ' ; '.</p>
Exclude the Record Codes	If this box is checked, the rows respecting the record codes will be excluded instead of being included when loading the file.
Logical format	The description of the data representation
Length	<p>The logical length of the column, as it is functionally seen by the end user. For a number, this is the total number of digits (Precision).</p> <p>Specify this length even if the physical length is given.</p>
Format	The format used for dates. This format is a Java date format that matches your machine's local parameters.
Precision	Total number of digits contained in the number.
Scale	Number of digits contained in the decimal part.
Decimal separator	The decimal separator (by default '.')

Description

Property	Description
Default Value	Value inserted in this column if no value is specified.
Read only	Check this box if this column cannot be used in an INSERT or UPDATE command. This is the case, for example, for columns whose values are automatically set and cannot be modified, such as IDENTITY columns.
Slowly Changing Dimensions Behavior	<p>This field defines the column behavior when loading a slowly changing dimension table for OLAP. The following behavior are supported:</p> <ul style="list-style-type: none"> ▪ Surrogate Key: The column is the unique (technical) identifier for a record version. This column is usually loaded by an auto-incremented value (sequence), and is references by the fact table. ▪ Natural Key: The column is part of the key identifying one record regardless of the versions. This key usually corresponds to the primary key of the source table. ▪ Overwrite on Change: If this column's value changes, the current record version is updated, and this column overwritten. ▪ Add Row on Change: If this column's value changes, a new record version is created. ▪ Current Record Flag: This column is the flag identifying the current record version. It is usually set to 1 for the current version and to 0 for older versions ▪ Starting Timestamp: record version validity starting date and time. ▪ Ending Timestamp: record version validity ending date and time. <p>These behavior is taken into account in some of the knowledge modules supporting slowly changing dimensions (SCD).</p>
Description	Detailed description of the column.

Control

The quality control properties are taken into account during a flow control, during a static control, or while loading a file.

Property	Description
Mandatory	Shows if the column must be completed.
Control	Shows the type of quality control for which the mandatory status will be checked.
Flow	During a flow control for data integrated to this datastore, the mandatory status of the column will be verified if this box is checked. This information is a default value that can be modified when the interface is designed. It is recommended that you activate this type of check to ensure data quality, even if the target technology also checks the information.
Static	During a static control, that is, when data already present in this datastore is checked, the mandatory status of this column will be verified if the box is checked.
On Error	Data in the file may be inconsistent with the datastore definition. When

	reading the file, if one value from the row is inconsistent with the column description, the On Error option defines the action Oracle Data Integrator will perform.
Reject Error	The row containing the error is moved into a file with the "BAD" extension, and an explanation of the error is put in a file with the "ERROR" extension . The "BAD" and "ERROR" are located in the same directory as the file being read.
Null if error (inactive trace)	The row is kept in the flow and the erroneous value is replaced by null.
Null if error (active trace)	The row is kept in the flow, the erroneous value is replaced by null, and an explanation of the error is put in a file with the "ERROR" extension .

Services

This tab contains the configuration for generating and deploying data services for this column. See [Setting Up Data Services](#) for more information.

Property	Description
Allowed operations	Check the box corresponding to the type of actions (INSERT/UPDATE/SELECT) you want to allow data services to perform on this column. One important use of this tab is to lock a column against being written to via Data Services.

References

A reference is a functional link between two Datastores. The reference corresponds to the foreign key concept in the relational model. For example: The INVOICE Datastore references the Customer Datastore through the customer number.

What is the point of a reference?

A reference is useful in several ways:

- Declaring references (or reversing them) improves a model's readability.
- Defining a reference allows the consistency of linked data to be checked.
- Checking a reference also validates the understanding of the model. If the application data is 100% inconsistent compared to the reference you have declared, you can assume that the cross-referral rule is wrong.
- The cross-reference rules will allow a flow control procedure to be generated when integrating data this Datastore. For example, if you declare (or reverse) a reference between the invoice and the customer, all the interfaces loading the invoice will exclude invoices referencing a non-existent customer.

Caution: Only technologies supporting SQL language support static data consistency check.

Caution: Creating a reference in the repository does not create an object in the data server on which the model is based.

Definition

Properties	Description
Name	Reference name.
Type	Simple: A reference defined by the user, based on columns equality. Complex: A reference between two datastores using a complex expression. External: A reference declared in a metadata repository that is external to Oracle Data Integrator.
Model	The model of the referenced table, or the primary key table, or the table known as parent.
Table	The referenced table, or the primary key table, or the table known as parent. If this information is not completed, or if a table not defined in the repository is referenced, its name must be entered in the group "External table".
External Table	Name of a table outside the models defined in Oracle Data Integrator.
Catalog	The catalog containing the external table.
Schema	The schema containing the external table.
Table	Name of the external table.
Active on the database	Shows that the reference exists on the database containing the Datastore.

Columns (non complex reference)

This tab is only displayed for simple or external references

For simple or external references, a list of the correspondence between the foreign key table columns (child table) and the primary key table (parent or referenced table) must be defined. In accordance with the ISO standard for relational models, only the rows in the foreign key table that include all the non-null foreign key columns will be checked.

Expression (complex reference)

The criterion for connecting two tables may be complex. In this case, a free expression may be entered in this tab.

Behavior (external reference)

Behavior is a purely indicative metadata, that concerns external foreign keys. This information shows if the reference (the foreign key) is checked (active) by the data server, and what action is launched by the data server if a row is deleted from the primary key table, or if the primary key value for a row is modified in the primary key table.

Control

The quality control properties are taken into account during a flow control or during a static control.

Property	Description
Type of quality control	Shows the type of quality control for which this reference will be checked.
Flow	During a flow control for this datastore, the reference will be verified if this box is checked. This information is a default value that can be modified when the interface is designed. It is advisable to activate this type of control to ensure data quality, even if the target technology also controls the information.
Static	During a static control, that is, a quality control of data already present in this datastore, the reference will be verified if the box is checked.

When the **Check** button is clicked, the static control is executed, and the only visible result is that the number of erroneous rows is displayed on the **Control** tab. To obtain a trace of the rows in error, a static control (called asynchronous) must be launched from either the **Datastore** or the **Model** window.

Keys

A key is a set of datastore columns that enables a datastore row to be identified and/or accessed in a privileged way in terms of performance (index). Some drivers recover the key descriptions during a reverse engineering process. It is also possible to enter a key definition directly into the Repository.

What is the point of declaring a primary or alternate key?

A primary or alternate key is useful in several ways:

- It improves readability of the model
- It enables the consistency checking of linked data, and thus the non-existence of duplicates to be verified
- Checking a reference also allows comprehension of the model to be validated. If the application data is 100% inconsistent compared to the key you have declared, you can assume that an invalid key identification was used.
- Identification rules allow an flow quality control procedure to be generated in this Datastore. For example, if you declare (or reverse engineer) a primary or alternate key on the datastore "Invoice", all the interfaces loading the invoice will exclude those with the same invoice number.

Caution: Only technologies supporting SQL language support static data consistency control.

Note: A non-unique index is purely informative and will not be checked.

Caution: Creating a key or an index in the repository does not create an object in the data server on which the model is based.

General Properties

Properties	Description
------------	-------------

Name	Name of the key
Key or Type of index	<ul style="list-style-type: none"> Primary key: a unique key, preferred for all criteria, objective (all these columns must be completed) and subjective (conciseness, performance, etc). Data Integrator uses the primary key as the default update key when the datastore is an interface target. Alternate key: a unique key not used as the primary key. Not unique index: An index used simply to improve access performance.

Columns

This tab allows the columns of the table participating in the key to be selected.

Control

The quality control properties are taken into account during a flow control or during a static control.

Property	Description
Defined in the database	Shows if this key has been defined in the database dictionary (or in the described application). A key that has been reverse engineered has been defined in the database.
Active	Shows if the key is active in the database. In some technologies, the dictionary may contain active keys (Enable) or non-active keys (Disable), that is, where the data server does not check data consistency.
Control	Shows the type of quality control for which this key will be checked.
Flow	During an flow control in this datastore, the key will be verified if this box is checked. This information is a default value that can be modified when the interface is designed. It is recommended to activate this type of control to ensure data quality, even if the target technology also checks the information.
Static	During a static control, that is, during a check of the data already present in this datastore, the key will be verified if the box is checked.

When the **Check** button is clicked, the static control is executed by the graphical interface, and the only visible result is that the number of erroneous rows is displayed on the **Control** tab. To obtain a trace of the rows in error, a static control (called asynchronous) must be launched from either the Datastore or the Model window.

Conditions

A condition is a Where-type SQL expression attached to a Datastore based on a RDBMS that supports SQL. The purpose of this condition is to filter or check the data in the Datastore in question.

What is the point of declaring a condition in Oracle Data Integrator?

An Oracle Data Integrator condition is useful in several ways:

- It allows the consistency of the datastore data to be checked according to a rule, for example.

- The conditions will allow a flow control to be generated in this Datastore. For example, if you declare a condition that checks customers' age on the "Client" datastore, all the interfaces loading the "Client" datastore will exclude those where the age is outside the limits.

Caution: Only technologies supporting SQL language support static control.

General Properties

Properties	Description
Name	Name of the condition
Type	<ul style="list-style-type: none"> Filters are a way of targeting the rows in a table involved in check and interface processes. A filter is used when a Datastore is used as source by an interface. An example of a filter that allows active customers to be targeted: <code>CLI.TYPE_CLIENT like 'A%'</code> Oracle Data Integrator controls represent a data consistency rule attached to the Datastore. This rule can be checked synchronously or asynchronously. Flows having this Datastore as target are also checked. Erroneous data can be isolated on the application workspace. An example of an Oracle Data Integrator condition that allows customers' age to be checked: <code>CLI.AGE between 0 and 130</code> DBMS constraints represent check constraints that have been reverse-engineered from an RDBMS or another application dictionary.
Where	<p>Where-type expression defining the condition. This expression must use the table alias (defined in the Datastore window) before each column.</p> <p>An example of an expression on the datastore "CLIENT" with the alias "CLI": <code>CLI.TYPE_CLIENT like 'A%'</code></p>
Message	Error message listed in the error table for rows in a Datastore containing an error. This information is not applicable to filters.

Control

The quality control properties are taken into account during a flow control or during a static control. Control information is not available for filters conditions, as a filter is not checked.

Property	Description
Defined in the database	Shows if this condition is defined in the database dictionary (or in the described application). A condition that has been reversed has been defined in the database.
Active	Shows if the condition is active in the database. In some technologies, the dictionary may contain active (Enable) or non-active (Disable) conditions, that is, where the data server does not check data consistency.
Control	Shows the type of control for which this condition will be checked.
Flow	During an flow control for this datastore, the condition will be verified if this box is checked. This information is a default value that can be modified when the interface is designed. It is recommended to activate this type of control to ensure

data quality, even if the target technology also checks the information.

Static During a static quality control, that is, of data already present in this datastore, the condition will be verified if the box is checked.

When the **Check** button is clicked, the static control is executed by the graphical interface, and the number of erroneous rows is displayed on the **Control** tab. To obtain a list of the rows in error, a static control (called asynchronous) must be launched from either the **Datastore** or the **Model** window.

Introduction to Operator

Through the **Operator** module, you can manage your interface executions in the **sessions**, as well as the **scenarios** in production.

The Operator module stores this information in a work repository, while using the topology defined in the master repository.

- Working with Operator

Operator's Interface

The Operator GUI appears as follow:

The Menu

The **Menu** contains pull-down menus to access the following features:

- Import
- Log purge
- Scheduling
- Display options
- Open modules or tree views
- Change the user's password and options

The Toolbar

The **Toolbar** lets you:

- Open other modules
- Browse the log
- Purge the log
- Display the scheduling information
- Refresh the log manually or automatically
- Open the on-online help

The Tree Views

Operator objects available to the current user are organized into the following tree views:

- **Session List** displays all sessions organized per date, physical agent, state, keywords, etc
- **Hierarchical Sessions** displays the execution sessions also organized in a hierarchy with their child sessions

- **Scheduling** displays the list of physical agents and schedules
- **Scenarios** displays the list of scenarios available

Each tree view appears in a floatable frames that may be docked to the sides of the main window. These frames can be also be stacked up. When several frames are stacked up, tabs appear at the bottom of the frame window to access each frame of the stack.

Tree view frames can be moved, docked and stacked by selecting and dragging the frame title or tab. To lock the position of views, select **Lock window layout** from the **Windows** menu.

If a tree view frame does not appear in the main window or has been closed, it can be opened using the **Windows > Show View** menu.

It is possible in each tree view to perform the following operations:

- Expand and collapse nodes by clicking on them
- Activate the methods associated to the objects (Edit, Delete, ...) through the popup menus
- Edit objects by double-clicking on them, or by drag and dropping them on the **Workbench**.

The Workbench

The Workbench displays the list of sub-objects for the object currently selected in the tree view. For example, if a step is selected in the tree view, the list of tasks for this step will be displayed in the workbench. The columns appearing in each list can be customized through the popup menu of the column titles.

The windows for object being edited or displayed appear in the **Workbench**.

Session

Session

A session is an execution (of a scenario, an interface, a package or a procedure, ...) undertaken by an execution agent. A session is made up of steps which are made up of tasks.

Definition

Properties	Description
Name	Name of the scenario, the package, the interface, the procedure, etc, executed in this session.
Version	Version (for a scenario).
Execution Context	The context in which the execution was run.
Execution Agent	Agent responsible for execution of the tasks in this session.
State	The session state when the session window was opened. The possible states are Done, Error, Running, Waiting and Warning.
Stop	Stops the Session

Restart	Restarts the session
Execution	Execution information details
Start	Start date and time of execution of the session.
End	End date and time of execution of the scenario.
Duration	The time taken for execution of the scenario.
Return code	Return code for the session
Message	Session execution error message, if needed.

Session Steps

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 an interface or a single variable, for example, a session has only one session step.

Definition

Properties	Description
Session	Unique session number.
Order number	The order number linked to the step for this session.
Name	Name of the step executed.
Type	Type of step executed. For more information about types of steps, refer to step.
Execution context	The context in which the step was executed.
Maximum number of errors allowed	<ul style="list-style-type: none"> If the box % is not checked, this is the maximum number of errors allowed (as total number of errors). If the box % is checked, this is the maximum percentage of errors (based on the total number of errors) allowed. <p>When this field is left empty, then an infinite number of errors is allowed.</p> <p>When this number or percentage of errors is exceeded, the step goes into error state.</p>

Execution

Properties	Description
Number of executions	In the event of successive executions of the step (loops), the execution number.
Execution	Gives the detailed result of execution of the step

Start	Start date and time of execution of the step.
End	End date and time of execution of the step.
Duration	The time taken for execution of the step.
State	State of the step (Done, Error, Running, Waiting or Warning).
Return code	Return code for the step.
Message	Step execution error message, if needed.
No. of Inserts	Number of rows inserted during the step.
No. of Deletes	Number of rows deleted during the step.
No. of Updates	Number of rows updated during the step.
No. of Errors	Number of rows in error in the step.

Session Tasks

The task is the smallest execution unit. It corresponds to a procedure command in a KM, a procedure, assignment of a variable, etc

Definition

Properties	Description
Description	Description of the type of task. Shows the type of operation in which the task participates. For an interface, this shows which procedure phase the interface is located in (loading, integration, check, etc) Name of the object handled by the task Detailed name of the task
Order	These two fields detail the order of the task in the session (unique for the session), and the task order in the KM or the procedure which contributed to its creation.
Type	Type of task executed.
Details	Logging level of the task.
Ignore Errors	Shows if this task sets up an error tolerance, and therefore, if errors for this task are blocking.

Description

This tab contains the command or commands launched on the source connections and, by default, during the task. These orders are in code in the native language of the technology. Nevertheless, they contain the variables in clear, as these are interpreted when the order is actually executed.

Execution

Properties	Description
Default/Loading Connection	The connection on which the default or loading commands of the description tab are launched.
Context	The execution context of the task
Schema	The logical schema for execution of the task
Connection	The data server for execution of the task
Transaction	Transaction Number in which the command is executed
Transaction Isolation	The transaction isolation level for Select orders. For more information on transaction isolation, refer to Specific procedure command.
Commit	Managing Commit for the transaction. For more information, refer to Specific procedure command.
Execution	Details the result of execution of the task
Start	Start date and time of execution of the task.
End	End date and time of execution of the task.
Duration	The time taken for execution of the task.
State	State of the task (Done, Error, Running, Waiting or Warning).
Return code	Return code for the task.
Message	Task execution error message, if needed.
No. of inserts	Number of rows inserted during the task.
No. of Deletes	Number of rows deleted during the task.
No. of Updates	Number of rows updated during the task.
No. of Errors	Number of rows in error in the task.
No. of Rows	Total number of rows handled during this task.

Session Variables

Session variable displays the values taken by a variable during the execution sessions.

Definition

Properties	Description
Name	Name of the variable. It is prefixed by GLOBAL or the code of the project

	containing this variable.
Description	Detailed description of the variable
Datatype	Type of variable: Alphanumeric, Date or Numeric (Maximum 10 digits).
Action	This parameter shows the length of time the value of a variable is kept for: <ul style="list-style-type: none"> • Non-persistent: The value of the variable is kept in memory for a whole session. • Last value: Oracle Data Integrator stores in the repository the latest value held by the variable. • Historize: Oracle Data Integrator keeps an history of all the values held by this variable.
Default Value	The default value assigned to the variable.

History

This window displays the history of the values of the variable with the context for variables with the "log" action, or the last value for a variable with the "last value" action.

Note: Each value is attached to the context in which the session was executed.

- The button allows to delete a line of the history.
- The **Context Filter** field lets you filter the entries for a given context.

Scenario

Scenarios

A scenario is designed to put a source component (interface, package, procedure, variable) into production. A scenario results from the generation of code (SQL, shell, etc) for this component.

Note: Once generated, the scenario is stored inside the work repository. The scenario can be exported then imported to another repository (remote or not) and used in different contexts.

Note: 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.

The scenario properties are displayed on the **Definition** tab in the **Scenario** window in the **Designer** module, or via the **Operator** module. Operation of a scenario takes place from the same window or from an operating system command window. In the Designer module, scenarios are grouped under their respective source components.

Properties

Properties	Description
Package/Interface/Variable/Procedure	Name of the component used to create the scenario

Name	Name of the scenario
Version	Version of the scenario
Description	Detailed description of the scenario

Execution

This tab allows you, by using the **Execute** button, to launch the scenario. The window for selecting the execution options opens.

Variables

This tab allows you to select the scenario variables. They will be displayed as parameters when starting the scenario using Metadata Navigator or when creating a schedule. This selection of parameter variables lets you hide from the user variables that cannot be parameterized.

If the **Use All** value is selected, then all variables are considered as parameters. If the **Selective Use** value is selected, you can select the variables to use as parameters.

Operating a scenario

A scenario is operated from the work repository it is stored in. The same scenario can be executed in several contexts from the same work repository.

A scenario can be:

- launched from Designer or Operator
- Scheduled with Data Integrator scheduler
- launched from an Operating System command line interface, using a Web Service or an HTTP URL.
- Scheduled with any Scheduler
- exported to another work repository where it can be similarly launched or scheduled. See [Operating in another repository](#).
- Import a scenario in production.

Scenario Reports

The scenario execution report gives information on the general execution of a scenario.

Definition

Properties	Description
Execution agent	Agent responsible for executing the scenario.
State	State of the scenario (Done, Error, Running, Waiting, Warning).
Execution context	Execution context of the scenario.

Start	Start date and time of execution of the scenario.
End	End date and time of execution of the scenario.
Duration	The time taken for execution of the scenario.
Return code	Return code of the scenario.
Message	Scenario execution error message, if needed.

Details

Properties	Description
No. of inserts	Number of rows inserted during the scenario.
No. of Deletes	Number of rows deleted during the scenario.
No. of Updates	Number of rows updated during the scenario.
No. of Errors	Number of rows in error in the scenario.
No. of Rows	Total number of rows processed by the scenario. Caution, this value is not the sum of the above values.

Step Reports

The step execution report is linked to the scenario execution report.

Definition

Properties	Description
Name	Name of the step executed.
Order number	The order number linked to the step for this session.

Execution


Properties	Description
Number of executions	In the event of successive executions of the step (loops), the execution number.
Execution	Gives the detailed result of execution of the step
Start	Start date and time of execution of the step.

End	End date and time of execution of the step.
Duration	The time taken for execution of the step.
State	State of the step (Done, Error, Running, Waiting or Warning).
Return code	Return code for the step.
Message	Step execution error message, if needed.
No. of Inserts	Number of rows inserted during the step.
No. of Deletes	Number of rows deleted during the step.
No. of Updates	Number of rows updated during the step.
No. of Errors	Number of rows in error in the step.

Scheduling Information

The Scheduling Information lets you visualize the agents' scheduled tasks.

Important: 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.

Properties	Description
Selected Agent	Agent for which the Scheduling is displayed. You can display also the scheduling of all agents
Scheduling from ... to ...	Time range for which the scheduling is displayed. Click the  Refresh button to refresh this schedule.
Upd.	This button updates the schedule for the selected agent(s)
Time Range	The time range specified (1 hour, 2 hours) 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.
Zoom in/out	Zooms in the Gantt diagram. Note that you can also zoom by selecting a zone in the diagram.
Gantt Diagram	This panel displays as a Gantt Diagram the schedule for the agent selected
Scenarios details	This panel displays the details and execution statistics for each scheduled scenario.

Using the diagram

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

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

Topology Manager

Introduction to Topology Manager

Using the **Topology Manager** module, you can manage the topology of your information system, the **technologies** and their **datatypes**, the **data servers** linked to these **technologies** and the **schemas** they contain, the **contexts**, the **languages** and the **agents**. In addition, Topology allows you to manage the **repositories**.

The Topology module stores this information in a master repository. This information can be used by all the other modules.

Topology Manager's Interface

The Topology Manager GUI appears as follow:

The Menu

The **Menu** contains pull-down menus to access the following features:

- Import/Export
- Wizards
- Display options
- Open modules or tree views
- Change the user's password and options

The Toolbar

The **Toolbar** lets you:

- Open other modules
- Refresh the Tree views
- Open the on-online help

The Tree Views

Topology Manager objects available to the current user are organized into the following tree views:

- The **physical architecture**, containing the **technologies** with their associated **data servers** and **physical schemas**, and the **physical agents**,
- The **logical architecture**, containing the **technologies** with their associated **logical schemas**, and the **logical agents**,
- The **contexts** linking logical and physical architectures,
- The **languages**, describing the different type of languages available.
- The **repositories**, including the actual master repository and the attached work repositories.

Each tree view appears in a floatable frames that may be docked to the sides of the main window. These frames can be also be stacked up. When several frames are stacked up, tabs appear at the bottom of the frame window to access each frame of the stack.

Tree view frames can be moved, docked and stacked by selecting and dragging the frame title or tab. To lock the position of views, select **Lock window layout** from the **Windows** menu.

If a tree view frame does not appear in the main window or has been closed, it can be opened using the **Windows > Show View** menu.

It is possible in each tree view to perform the following operations:

- Insert or import root objects to the tree view by clicking the appropriate button in the frame title
- Expand and collapse nodes by clicking on them
- Activate the methods associated to the objects (Edit, Delete, ...) through the popup menus
- Edit objects by double-clicking on them, or by drag and dropping them on the **Workbench**.

The Workbench

The windows for object being edited or displayed appear in the **Workbench**.

What is the Topology?

Through the **Topology Manager** module, the user has at his disposal a genuine physical and logical cartography of the architecture and the components he works on through Oracle Data Integrator.

- Create the Topology

Physical Architecture

The physical architecture defines the different elements of the information system, as well as the characteristics taken into account by Oracle Data Integrator.

A technology handles formatted data. Therefore each technology is associated with one or more Datatypes that allow Oracle Data Integrator to generate data handling scripts.

Note: Each type of database (Oracle, DB2, etc) , file format (XML, File), or application software is represented in Oracle Data Integrator by a technology.

The physical components that store and return data are defined as Data Servers. A data server that can store different information according to a business logic, can be divided into Physical Schemas. A data server is always linked to a single technology.

Note: Every database server, JMS message file, group of flat files, etc, that is used in Data Integrator, must be declared as a data server.
Every schema, database, JMS Topic, etc, used in Oracle Data Integrator, must be declared as a physical schema.

Finally, the physical architecture includes definition of the Physical Agents, the Java software components that allow Oracle Data Integrator jobs to be executed on a remote machine.

Contexts

Contexts bring together components of the physical architecture (the real Architecture) of the information system with components of the Oracle Data Integrator logical architecture (the Architecture on which the user works).

Logical Architecture

The logical architecture allows a user to group physical schemas containing datastores that are structurally identical but located in separate places, into Logical Schemas, structured per technology.

According to the same philosophy, the logical architecture defines the Logical Agents, allowing a unique name to be given to all the physical agents with the same function in different contexts.

For example: The **logical schema** "Accounting" may correspond to two **physical schemas**:

- "Accounting Oracle sample", used in the "development" **context**
- "Accounting corporate", used in the "production" **context**.

The two **physical schemas** are structurally identical (they contain accounting data), but physically different. They are located on two Oracle schemas, and perhaps on two different Oracle servers (data servers).

Languages

This type of component defines the characteristics specific to each language linked to a technology and used by Oracle Data Integrator.

Repositories

This branch of the topology contains information relating to the two types of repositories: the master repository and the work repositories.

Hosts

Hosts and Usages enable to manage user access to the graphical modules.

Connection

Connecting to the Master Repository

With this window, a connection to a master repository can be selected when opening the **Topology Manager** or **Security Manager** module. The creation, modification or deletion of existing connections can also be launched.

Note: The definitions of the master repository connections are stored locally in the `/bin/snps_login_security.xml` file

General Properties

Properties	Description
Login	List of the defined connections to the master repositories.

name	
User	The default Oracle Data Integrator user for connecting to the selected master repository. If no default user has been specified, this must be completed.
Password	The password corresponding to the user for the connection. If no default password has been specified, this must be completed.

Toolbar

Button	Description
	Allows a new connection to a master repository to be created.
	Allows the connection selected in the login name field to be modified
	Allows the connection selected in the login name field to be deleted

Master Repository Connection Details

With this window, a connection to a **master repository** can be defined or modified.

Note: The definitions of the master repository connections are stored locally in the `/bin/snps_login_secu.xml` file

General Properties

Properties	Description
Oracle Data Integrator Connection	With this information group, the properties displayed in the connection window can be defined.
Login name	This is the name presented in the list when a Data Integrator module is opened.
User	The default Oracle Data Integrator user for connecting to the master repository. If no default user has been specified, the user must enter it each time this connection is used.
Password	Password corresponding to the Oracle Data Integrator user. If no default password has been specified, the user must enter it each time this connection is used.
Database Connection (Master Repository)	This information group shows the connection parameters for the relational database that hosts the master repository.
User	User allowing you to connect to the schema of the database containing the master repository.
Password	The data server password for the database user.

Driver List	Type of technology the master repository is based on. You can obtain more information about the technology and the driver used by clicking on the button. For more information, refer to Driver details.
Driver name	Name of the JDBC driver used for connecting to the data server of the master repository. When a technology is selected, Data Integrator displays the most frequently used driver for this technology. Nevertheless, a different one can be used. For this, this driver must be correctly installed. For more information, refer to Installing JDBC / JMS drivers in the installation guide.
Url	URL for connecting to the data server of the master repository. You can select a URL model for the driver used by clicking on the button. For more information, refer to URL Samples. For more information on drivers and URLs, refer to the section JDBC URL Sample.
Default Connection	If this box is checked, when an Oracle Data Integrator graphic module is started, the current connection will be proposed by default.

Driver Details

This window provides information about the driver used with a technology.

General Properties

Properties	Description
Name	The driver used for the selected technology.
Description	Information about the driver.

For more information on drivers and URLs, refer to the section JDBC URL Sample.

URL Samples

This window provides URL models for a driver.

General Properties

Properties	Description
Name	List of valid URL models for the selected driver.
Description	Details on the URL model.

For more information on drivers and URLs, refer to the section JDBC URL Sample.

Physical Architecture

Technologies

Technology

In the Oracle Data Integrator terminology, this is any type of technology accessible by JDBC, ODBC, JMS, JNDI, JCA, or any operating system.

Oracle Data Integrator allows scripts to be generated and executed in the languages compatible with the technologies defined in its repository. Any type of technology can be declared and defined provided that Oracle Data Integrator can access it through appropriate middleware (JDBC, ODBC, JMS) or through an operating system command.

Example of technologies: Oracle, Sybase, Sybase IQ, DB2, Files, etc

Definition

Properties	Description
Name	Name of the technology, as it appears in the different lists in the graphical interface.
Code	<p>The technology code allows the technology to be referenced among the different repositories.</p> <p>Caution: Avoid changing this code, as this could make some Oracle Data Integrator objects inconsistent if referencing a technology from another repository.</p>
Technology type	<p>Classification of the technology. The possible values are:</p> <ul style="list-style-type: none"> • Database or files: Any technology accessible through ODBC and/or JDBC. Flat and XML files are part of this category, as Data Integrator includes a JDBC access driver for flat files, and a JDBC driver for XML files. • Operating system: Any operating system on which Oracle Data Integrator can launch commands via a Java virtual machine. • Topics (JMS): Any MOM accessible through JMS and including topic management (publication and subscription). • Queue (JMS): Any MOM accessible through JMS. • Oracle Data Integrator API: Technology allowing calls to the Oracle Data Integrator tools. • Oracle Data Integrator Connector: Technology allowing calls to a Java API. • Bean Scripting Framework: Technology allowing calls to a script interpreter. • Web Service Container: Technology allowing the deployment of web services.
Logical/Physical	<p>Defines if the technology is physical and/or logical.</p> <p>Three cases are possible:</p> <ul style="list-style-type: none"> • Logical and Physical Technology: If a technology is both logical and

	<p>physical, it can support logical and physical schemas. Its logical schemas can be mapped in the contexts exclusively on physical schemas <u>of this technology only</u>.</p> <ul style="list-style-type: none"> • Logical Technology: If a technology is exclusively logical, it can support logical schemas only, that can be mapped in the contexts on physical schemas <u>of any technology</u>. Example: If you define an exclusively logical technology <code>GENERIC_ISO_SQL</code>, its logical schemas can give access to physical schemas based on Oracle, DB2, ... data servers. • Physical Technology: If a technology is exclusively physical, it can support physical schemas only, and its schemas can be accessed only through the logical schemas of an exclusively logical technology.
Data handling	This section details the technology's data handling capacities.
Select	The technology's capacity to construct a data array resulting from a query.
Where	The technology's capacity to filter this array according to conditions
Reference	<p>The technology's capacity to join several arrays (two-dimensional data sets) into one only. This capacity can take several forms:</p> <ul style="list-style-type: none"> • None: The technology does not accept joins. • Not ordered: The specified joins will be used in a not ordered way, these are usually joins specified in SQL Where-type clauses in the form <code>where emp.empno = dept.deptno</code> • Ordered (SQL ISO): The specified joins will be used in an ordered way, that is, the final result data array is constructed by adding elementary sets in a sequenced way. This syntax is adapted to a specified ISO syntax in a "From" clause in SQL language, for example. For example: <code>FROM emp INNER dept ON (emp.empno = dept.deptno)</code>.
Naming rules	This section shows how to locate and name the data containers for this technology.
File	Shows if this technology is based on the use of files.
Using "Data server"	<p>Shows that the technology uses a data server whose name can be used to name objects. If the box is checked, the term used to describe the data servers for the technology must be specified.</p> <p>For example, in "Oracle" technology, the data server is accessed by a "Instance/DBLink".</p>
Using "Catalog"	<p>Shows that the technology uses a catalog to name its datastores. If the box is checked, the term used to describe the catalog concept in the technology must be specified.</p> <p>For example, Microsoft SQL Server uses a catalog named "Database", IBM AS/400 uses a catalog named "Library", Oracle does not use catalog.</p>
Using "Schema"	<p>Shows that the technology uses a schema to name its datastores. If the box is checked, the term used to describe the schema concept in the technology must be specified.</p> <p>For example, Microsoft SQL Server uses a schema named "Owner", Oracle uses a schema named "Schema/User", Microsoft Access does not use</p>

	schema.
Local object mask	<p>The local object mask defines how an object is named when you are connected to the data server the object is based on.</p> <p>For example, for Oracle, you should indicate %SCHEMA.%OBJECT to symbolize the syntax "SCOTT.EMP" giving access to the table EMP belonging to the user SCOTT.</p> <p>In Oracle Data Integrator, the tags available for this mask are:</p> <ul style="list-style-type: none"> • %CATALOG to symbolize the catalog name (property of the physical schema), • %SCHEMA to symbolize the schema name (property of the physical schema), • %OBJECT to symbolize the datastore name. <p>Caution: These tags are case-sensitive.</p>
Remote object mask	<p>The remote object mask defines how an object is named when you are connected to a different data server from the one the object is based on.</p> <p>For example, for Oracle, you should indicate %SCHEMA.%OBJECT@DSERVER to symbolize the syntax SCOTT.EMP@NYORK allowing access to the EMP table of the user SCOTT of the instance (remote data server) NYORK.</p> <p>The tags available for this mask are the same as for local objects, to which must be added %DSERVER representing the name of the data server (property of the connection).</p>

SQL

The SQL properties are reserved for technologies with data and/or join filtering capacities (Where).

Properties	Description
Where	This section details the syntax used for filter clauses. It is only displayed for technologies with a data array filtering capacity.
Order by	<p>Shows the type of syntax used in SQL for the technology, following an ORDER BY clause. The possible values are:</p> <ul style="list-style-type: none"> • Complex Expression: The clause is followed by a complex expression or the column name. • Column Number: The ORDER BY clause is followed by the column numbers. • Alias: The ORDER BY clause is followed by the column aliases.
Group by	Shows the type of syntax used in SQL for the technology, following a GROUP BY clause. The possible values are the same as for an ORDER BY clause.
Having	Shows the type of syntax used in SQL for the technology, following a HAVING clause. The possible values are the same as for an ORDER BY clause.

Ordered joins (ISO)	This section details the syntax used for sequenced joins (SQL ISO). It is only displayed for technologies with a join sequencing capacity.
Clause Location	Shows whether the join clause is located on the "FROM" level or the "WHERE" level in the query syntax.
Brackets Supported in the ON clauses	Shows whether the technology accepts sub-sets delimited by parentheses (this is the case with SQL ISO syntax).
INNER	Shows that the technology supports inner joins. The key word specifying an inner join must be given, for example, "INNER JOIN".
CROSS	Shows that the technology supports Cartesian product. The key word specifying a product set must be given, for example, "CROSS JOIN" or ",".
LEFT OUTER	Shows that the technology supports left outer joins. The key word specifying a left outer join must be given, for example, "LEFT OUTER JOIN".
RIGHT OUTER	Shows that the technology supports right outer joins. The key word specifying a right outer join must be given, for example, "RIGHT OUTER JOIN".
FULL OUTER	Shows that the technology supports full outer joins. The key word specifying a full outer join must be given, for example, "FULL OUTER JOIN".
Not ordered join	The parameters of non-sequenced joins allow outer join clauses to be generated in the WHERE clause.
Outer Keyword	Keyword or sign imposed by the syntax as a reminder that an element is considered external. For Oracle, for example the Outer Keyword is: (+)
Outer location	Shows if the OUTER Keyword is located outside (OUTER side) or inside (INNER side) the join syntax.
Before/After column	Shows if the OUTER Keyword should be placed before or after the name of the object contributing to the join.
Specific Queries	Queries used by Oracle Data Integrator to performed specific tasks. These queries are technology-specific.
Indexes Reverse	Query used to reverse-engineer indexes. This query should return a recordset (one row per index column) containing the following VARCHAR fields: <ul style="list-style-type: none"> • INDEX_NAME: Name of the index. • COLUMN_NAME: Name of the index column. The recordset should be ordered by INDEX_NAME and by the position of COLUMN_NAME in the index.
Check Constraints Reverse	Query used to reverse-engineer check constraints. This query should return a recordset (one row per check constraint) containing the following

	<p>VARCHAR fields:</p> <ul style="list-style-type: none"> • CHECK_NAME: Name of the check constraint. • CHECK_TXT: SQL statement of the constraint. • CHECK_STATUS: Status of the check constraint - '1' for enabled, '0' for disabled. • CHECK_DESC: Description of the check constraint.
Alternate Key Reverse	<p>Query used to reverse-engineer alternate keys. This query should return a recordset (one row per alternate key column) containing the following VARCHAR fields:</p> <ul style="list-style-type: none"> • AK_NAME: Name of the alternate key. • COLUMN_NAME: Name of the alternate key column. <p>The recordset should be ordered by AK_NAME and by the position of COLUMN_NAME in the alternate key.</p>
Current Date	<p>Query that can be sent at anytime to a data server of this technology to return the current date and time of the server. This query is used internally to recover connections.</p>

Others

Properties	Description
Column-alias separator	<p>The element designed to separate a column from its alias in an SQL SELECT clause.</p> <p>In the ISO standard, this separator is "AS", however, it is not supported by all databases and may therefore be left incomplete.</p>
Table-alias separator	<p>The element designed to separate a table from its alias in an SQL FROM clause. This element may be left incomplete.</p>
Date Function	<p>The function allowing the date and the time to be returned.</p> <p>For example: <code>getdate()</code> for Microsoft SQL Server or <code>sysdate</code> under Oracle</p>
String datatype mask	<p>The syntax used to describe Chain-type data in DDL orders (table creation). The Oracle Data Integrator tags %L (data Length) and %P (data Precision) can be used in the syntax description.</p>
Date datatype mask	<p>The syntax used to describe Date-type data in DDL orders (table creation).</p>
Numerical datatype mask	<p>The syntax used to describe Numerical-type data in DDL orders (table creation). The Oracle Data Integrator tags %L (data Length) and %P (data Precision) can be used in the syntax description.</p>
DDL Null key word	<p>The word used to describe a column that can contain empty values (column known as NULLABLE).</p>
Maximum column name length	<p>The maximum length of a column name in number of characters. All column names generated by Oracle Data Integrator are truncated to this size.</p>

Maximum table name length	The maximum length of a datastore (table) name in number of characters. All table names generated by Oracle Data Integrator are truncated to this size.
---------------------------	---

Language

This tab describes the languages implemented by the technology and allows overwriting any of the properties of the language.

Properties	Description
Language	The language that is totally or partially implemented by the technology, selected from the list of defined languages.
Compatible	Shows that the language for the technology inherits all the language elements marked as standard . For more information, refer to language element.
Default	Shows that this is the default language for this technology.
Object delimiter	<p>The character used to delimit object names. Replaces the standard delimiter given in the language.</p> <p>Caution: If the two delimiters (before and after the object name) are different, then you should use both of them. If not, one delimiter is enough.</p> <p>Use a single double-quote " to generate code like this:</p> <pre>create table MySchema."My_Table" (...</pre> <p>Use two square brackets [] to generate code like this:</p> <pre>create table MySchema.[My_Table] (...</pre>
Word separator	The characters that can be used to separate language words (objects, key words, literals).
Literal delimiter	<p>The character used to delimit the literal names (values). Replaces the standard delimiter given in the language.</p> <p>Caution: If the two delimiters (before and after the literal) are different, then you should use both of them. If not, one delimiter is enough.</p>
Objects case-sensitive	<p>The object names in this technology are case-sensitive.</p> <p>For example objects are not case sensitive in the SQL Language by default. Therefore code like this will be generated to preserve names using upper and lowercase letters:</p> <pre>create table Snps_Temp.dbo."My_Table" ("My_Column1" VARCHAR(20) NULL, MY_COLUMN2 VARCHAR(20) NULL,)</pre> <p>If "Objects case-sensitive" is selected, then code like the following will be generated:</p> <pre>create table Snps_Temp.dbo.My_Table (My_Column1 VARCHAR(20) NULL, MY_COLUMN2 VARCHAR(20) NULL)</pre>

Words case-sensitive The keywords of this language are case-sensitive.

The following operations are possible on a technology:

- Automatically reversing the datatypes

Datatypes

The technologies storing the formatted data allocate to each of them a type that defines their nature. *For example:* numerical, character, Date, etc

Some data server access drivers allow you to Reverse the datatypes automatically from the **Technology** window, using the **Reverse** button.

Definition

Property	Description
Code	Code used in Oracle Data Integrator to refer to this datatype. This code is unique for this technology. The code is stored in models and flows in order to refer to the datatype.
Name	Name of the datatype, as it appears in the graphic interface and the reports.
Reversed Code	Code returned by the driver and/or an SQL reverse query.
Create Table Syntax	The syntax used to create a column of this type. The tags %L (Length) and %P (Precision) can be used in this syntax.
Writable Datatype Syntax	This alternate syntax is used to create a column storing data for a non writable datatype. The tags %L (Length) and %P (Precision) can be used in this syntax. If the type is writable then the Writable Datatype Syntax and Create Table Syntax should be the same.
Letter used for the icon	The letter used to represent the datatype (Graphic id code for this datatype).
Writable	Check this box if this datatype can be used in an INSERT or UPDATE command. This is not the case, for example, for datatypes which values are automatically set and cannot be modified such as IDENTITY columns. Columns reversed with a datatype not writable are flagged read-only. A datatype that is not writable should have an alternate Writable Datatype Syntax defined, to enable Oracle Data Integrator to create temporary tables storing data of this type.

Converted to

The **Converted to** tab allows you to specify for the other technologies, the datatype corresponding to the current type. This datatype will be used if Oracle Data Integrator needs to create a work table on another technology that is a reflection of a datastore in the current technology.

For example:

- The Oracle datatype VARCHAR2 will be transformed into VARCHAR on a Microsoft SQL Server data server.

For the data flow of an Oracle instance towards an SQL Server instance, the language generation engine may require conversion in both directions (for example from Oracle to SQL Server and from SQL Server to Oracle).

Converted from

This tab shows, (read only), the other technologies' datatypes that correspond to the current datatype.

Data server

A data server is a data processing resource which stores and reproduces data in the form of tables. It can be a database, a MOM, a connector or a file server.

A **data server** is always linked with one technology and one only. The **data server** is physically identified: it is located on a physical machine accessible by a TCP/IP network.

Caution: You are strongly advised to define each data server only once. If the same data server is declared several times (possibly with different connection parameters), Oracle Data Integrator could generate unnecessary data loading phases that waste execution time.

Definition

The definition tab contains the following fields, except for data servers based on a technology which is a *Web Service Container*, such as Axis2.

Properties	Description
Name	Name of the data server, as it appears in the graphical interface. Note: For naming data servers, it is advisable to use a nomenclature such as: <TECHNOLOGY_NAME>_<SERVER_NAME>.
Technology	Technology linked to the data server.
(data server)	This is the physical name of the data server. Define this name if your data servers can be inter-connected in a native way. For example, for Oracle, show here the name of the Database Link used for accessing this data server from another Oracle data server.
Connection	This section details the authentication method on the data server.
User	User name used for connecting to the data server. 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).
Password	Password linked with the user name. Note: This password is stored encrypted in the repository.

JNDI Connection	Some technologies accept, forbid or impose connection through an LDAP directory. Check this box if you can or should connect to your data server via LDAP.
Array Fetch Size	The number of rows (records read) requested by Data Integrator on each communication with the data server.
Batch update size	The number of rows (records written) in a single Oracle Data Integrator INSERT command.

Caution: The **fetch array** and **batch update** parameters are only accessible with JDBC. However, not all JDBC drivers accept the same values. At times, you are advised to leave them empty.

Note on fetch array and batch update: The greater the number specified in each of these values, the fewer the number of exchanges will be between the data server and Oracle Data Integrator. However, the load on the Oracle Data Integrator machine will be greater, as a greater volume of data will be recovered on each exchange.

Batch update management, like that of fetch array, falls within optimization. We recommend starting from a default value (30), then increase the value by 10 each time, until there is no further improvement in performance.

The definition tab appears as follows for *Web Service Containers*. See *Setting Up Data Services* for more information.

Properties	Description
Name	Name of the data server, as it appears in the graphical interface. Note: For naming data servers, it is advisable to use a nomenclature such as: <TECHNOLOGY_NAME>_<SERVER_NAME>.
Base URL for published services	Base URL into which the web services will be deployed. This URL is used when generating the WSDL.
Deployment Options	This section details the deployment method for the web services in this container.
Save the web services in the following folder:	This selection allows web services to be deployed using file copy in the target directory. The directory should be accessible from the machine performing the web services generation.
Upload the web services using Axis2	This selection allows web services to be deployed using the <i>Web Services Upload</i> feature of Axis2. You must specify the base URL for the Axis2 webapp as well as the user name and password to connect this application.
Base URL for the Axis2 webapp.	HTTP URL of the Axis2 application. Usually, it is <code>http://<Name of Tomcat server>:<HTTP port>/axis2/axis2-admin/</code> .
User name/password	User name and password to connect the Axis2 server. This user is typically the Axis2 administrator.
Upload the web services using FTP	This selection allows web services to be deployed using FTP upload. You must specify the target FTP URL as well as a valid FTP user.
FTP Server URL	FTP URL into which Oracle Data Integrator will deploy the web services.

User name/password	User name and password to connect the FTP server. This user must have write privileges to the FTP URL.
--------------------	--

JDBC

A direct JDBC connection allows you to access a data server without using an LDAP directory. This tab is only displayed if the **JNDI Connection** box on the **definition** tab is not checked.

Properties	Description
JDBC Driver	Name of the JDBC driver used for connecting to the data server For more information on drivers, refer to the section JDBC URL Sample. For more information on installing JDBC drivers, refer to Installing JDBC / JMS drivers in the installation guide.
JDBC URL	URL allowing you to connect to the data server. The URL allows the data server to be located on the TCP/IP network. Each driver has a specific driver syntax. For more information on URLs, refer to the section JDBC URL Sample.

JNDI

A JNDI connection consists of fetching connection information (driver, URL, and user and password if necessary) in a naming or directory service, such as LDAP, Novell Netware NDS, CORBA Naming Service, and File System. This naming or directory service is accessed by a URL on the JNDI protocol. The JNDI connection can be used for some databases (JDBC), it is compulsory for accessing MOM (JMS).

This tab only appears if the **JNDI Connection** box on the **definition** tab has been checked.

Property	Description
JNDI authentication	<ul style="list-style-type: none"> • 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	The user connecting to the naming or directory service
Password	Password of the user connecting to the naming or directory service
JNDI Protocol	The protocol used for the connection. Note: Only the most common protocols are listed here. This is not an exhaustive list. <ul style="list-style-type: none"> • LDAP: Access to an LDAP directory • SMQP: Access to a SwiftMQ MOM directory

	<ul style="list-style-type: none">• <other value>: access following the sub-protocol <other value>
JNDI Driver	The driver allowing the JNDI connection. Example for the Sun LDAP directory: <code>com.sun.jndi.ldap.LdapCtxFactory</code>
JNDI URL	The URL allowing the JNDI connection. For example: <code>ldap://suse70:389/o=linuxfocus.org</code>
JNDI Resource	The directory element containing the connection parameters. For example: <code>cn=sampledb</code>

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.
- For JNDI: These properties depend on the resource used.

Property	Description
Key	Key identifying this property. This key is case-sensitive.
Value	Value for the property.

The following operations are possible on a data server:

- Testing a data server connection

Physical Schemas

The physical schema is a decomposition of the data server, allowing the Datastores (tables, files, etc) to be classified. Objects stored in data servers with this mode of classification can be accessed by specifying the name of the schema attached to the object name.

Examples:

- Oracle classifies its tables by "schema" (or User). Each table is linked to a schema, thus SCOTT.EMP represents the table EMP in the schema SCOTT.
- Microsoft Access does not have schemas.
- DB2/400 has schemas called "Libraries".
- Microsoft SQL Server has a schema called "Owner" for each database. This owner is named by default: 'dbo'. A table is accessed under the form HR.dbo.EMP to access the table EMP belonging to the user dbo in the database HR.

Note: Technologies that don't really have schemas must still undergo a default physical schema creation.

Note: To access the data in a data server, all the schemas containing the datastores used in your project must be declared.

Definition

Property	Description
Name	Name of the physical schema, as it appears in the graphic interface. It is calculated automatically.
(Schema)	Name of the schema in the data server. Schema, owner, or library where the required data is stored. Caution: 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.
(Work schema)	For some data validation or transformation operations, Oracle Data Integrator may require work objects to be created. Indicate the schema you want to create these objects in. Caution: 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. Note: It is preferable to 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.
Default	If this box is checked, then the physical schema will be the data server default schema, when no schema has been specified. Only one physical schema can be marked by default . The default schema of each data server is indicated with this icon:
Work tables prefixes	This section details the work table prefixes that Oracle Data Integrator is likely to create in the work schema of this physical schema.
Errors	Prefix used to create the tables that contain erroneous data. These tables are created and/or updated during a data quality control and can be consulted from the graphic interface.
Loading	Prefix used to create the objects (tables, views, files, etc) that allow data loading between two data servers.
Integration	Prefix used to create the objects (tables, files, etc) dedicated to data integration during execution of an interface.
Journalizing Elements Prefixes	This section details the prefixes that Data Integrator is likely to use to create elements for journalizing in this schema.
Tables	Prefix used to create the journalizing tables (containing the changes indications.)

Views	Prefix user to create the views linking journalizing tables and the data tables.
Triggers	Prefix used to create triggers on the data stables enabling to update the journalizing tables.
Naming rules	This section shows how to locate and name (term) the data containers for this technology.
Local object mask	<p>The local object mask shows how to name an object in the physical schema, when you are connected to the data server on which the object is based.</p> <p>For example, for Oracle, you should indicate %SCHEMA . %OBJECT to symbolize the syntax "SCOTT.EMP" giving access to the table EMP belonging to the user SCOTT.</p> <p>In Oracle Data Integrator, the tags available for this mask are:</p> <ul style="list-style-type: none"> • %CATALOG to symbolize the catalog name (property of the physical schema), • %SCHEMA to symbolize the schema name (property of the physical schema), • %OBJECT to symbolize the datastore name. <p>Caution: These tags are case-sensitive.</p>
Remote object mask	<p>The remote object mask shows how to name an object in the physical schema, when you are connected to a different data server from the one this object is based on.</p> <p>For example, for Oracle, you should indicate %SCHEMA . %OBJECT@DSERVER to symbolize the syntax SCOTT . EMP@NYORK allowing access to the EMP table of the user SCOTT of the instance (remote data server) NYORK.</p> <p>The tags available for this mask are the same as for local objects, to which must be added %DSERVER representing the name of the data server (property of the connection).</p>

Context

In a project, the datastores of a physical schema are always accessed by specifying the logical schema and the context.

Caution: To be able to use a **physical schema** in Oracle Data Integrator, it is imperative that it be associated with a **physical schema** in a given **context**.

Property	Description
Context	Shows the contexts in which this physical schema is represented.
Logical schema	Shows the name of the logical schema through which you can access the physical schema for the specified context.
	Note: If there is no suitable logical schema name in the list, entering a new name will automatically create a new logical schema.

Actions

Action Groups

Actions (Templates for DDL commands) are organized into **Action Groups**. An action group corresponds to a given syntax and/or purpose (Oracle, SQL-92, etc).

Definition

Properties	Description
Action Group Name	Name of the action group, as it appears in the graphical interface.
Group Code	The group code allows the action group to be referenced among the different repositories. Caution: Avoid changing this code, as this could make some objects inconsistent.
Default Group	Check the Default Group box if you want this action group to be selected for new models created for this technology. If no default action group is selected for a given technology, new models use the generic actions .
Description	Detailed description of the action group.

Actions

Actions are templates for Data Definition Language (DDL) commands. They are used by the Common Format Designer to generate the scripts implementing a data model into a data server or synchronizing the differences between a data model described in Oracle Data Integrator and its implementation in the data server.

An action corresponds to a DDL operation (create table, drop reference, etc).

Definition

Properties	Description
Name	Name of the action, as it appears in the graphical interface.
Type	Type of operation performed by the action. See below for the list of action types.
Description	Detailed description of the action.

Action Types

The following action types are available:

- <Unknown>

- Add
 - Alternate Key
 - Check Constraint
 - Column
 - Foreign Key
 - Index
 - Primary Key
- Begin
- Modify
 - Column Attribute
 - Key Type
 - Column Comment
 - Table Comment
- Create Table
- Disable Key
- Drop
 - Alternate Key
 - Check Constraint
 - Column
 - Foreign Key
 - Index
 - Primary Key
 - Table
- Enable Key
- End
- Rename
 - Column
 - Table

Details

Each action contains several **Action Lines**, corresponding to the commands required to perform the DDL operation (for example, dropping a table requires dropping all its constraints first). The detail tab lists these action lines, which you can create, delete and organize using the buttons beside the grid. To edit an existing action, double-click on the row in the grid. To duplicate an action line, right-click it and select **Duplicate**.

Action Lines

An action corresponds to a DDL operation (create table, drop reference, etc). Each action contains several **Action Lines**, corresponding to the commands required to perform the DDL operation (for example, dropping a table requires dropping all its constraints first).

Definition

Properties	Description
Name	Name of the action line. This name will be given to the procedure command generated.
Ignore Errors	This option must be checked if you do not want the procedure to stop if this specific command returns an error. (For example, dropping a non-existing table should now block the whole process)
Commit	Indicates if a commit should be issued at the end of the command.
Isolation Level	It is the transaction isolation level for the command. The levels of isolation are: <ul style="list-style-type: none"> • Read Uncommitted: The transaction can read data not committed by another transaction. • Read Committed: The transaction can only read data committed by other transactions (in general, this is the default mode of many data servers). • Repeatable Read: The transaction is certain to read the same information if it executes the same SQL query several times, even if the rows have been modified and committed by another transaction in the meantime. • Serializable: The transaction is certain to read the same information if it executes the same SQL query several times, even if the rows have been modified, deleted or created and committed by another transaction in the meantime.
Action Text	Code of the command to execute. You may call the expression editor by clicking

Agents

Physical Agent

The Agent is a Java service which can be placed as listener on a TCP/IP port.

This service allows:

- execution on demand of jobs (model reverses, packages, scenarios, interfaces, etc), from graphical modules. For this, you must launch a listener agent.
- execution of scheduled scenarios, in addition to executions on demand. The physical agent contains an optional **scheduler** that allows scenarios to be launched automatically according to predefined scheduling . For this, you must launch a scheduler agent.

Definition

Properties	Description
Name	Name of the agent used in the graphical interface.
Host	Network name or IP address of the machine the agent has been launched on. In some TCP/IP configurations, it is preferable to specify an IP address (e.g.

	195.10.10.5) rather than a machine name.
Port	Listening port used by the agent. By default, this port is the 20910.
Maximum number of sessions	Maximum number of session allowed on this agent. This value is used when using Load Balancing.
Scheduling Information	Opens the Agent Scheduling Information Window
Update scheduling	This action reloads the agent's execution planning from the different Agent schedules planned on the scenarios contained in the work repositories.
Test	This action lets you test that the agent specified runs correctly.

Load balancing

For more information, see load balancing.

Properties	Description
Linked	Indicate if the given physical agent can receive delegated sessions from the current agent.
Agent	Name of the physical agent that can receive sessions from the current agent.

Load Balancing

Oracle Data Integrator implements load balancing between physical agents.

Concepts

Each physical agent is defined with:

- a **Maximum number of sessions** it can execute simultaneously,
- optionally, a number of **linked** physical agents to which it can delegate sessions' executions.

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

Determining the Maximum number of sessions

The maximum number of sessions is a value that must be set depending on the capabilities of the machine running the agent. It can be also set depending on the amount of processing power you want to give to the Data Integrator agent.

Delegating sessions

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

If the user parameter "Use new load balancing" is in use, sessions are also re-balanced each time a session finishes. This means that if an agent runs out of sessions, it will possibly be reallocated some from another agent.

Note: An agent can be linked to itself. An agent not linked to itself is only able to delegate sessions to its linked agents, and will never execute a session.

Note: Delegation works on cascades of linked agents. Besides, it is possible to make loops in agents links. This option is not recommended.

Agent unavailable

When for a given agent the number of running sessions is equal to its **Maximum number of sessions**, the agent will set incoming sessions in a "queued" status until the number of running sessions falls below the maximum of sessions for this agent.

Setting up load balancing

To setup load balancing:

1. Define a set of physical agents, and link them to a root agent (see Create a Physical Agent).
2. Start the root and the linked agents.
3. Run the executions on the root agent. Oracle Data Integrator will balance the load of the executions between its linked agents.

Note: The execution agent for a session can be seen in the session window in Operator.

Note: For load balancing to work between agents, it is necessary to have name the agents, that is start them with the `-NAME` parameter. See Launching a Listener Agent for more details.


See also:

- Physical Agent
- Creating a Physical Agent
- Session

Scheduling Information

The Scheduling Information lets you visualize the agents' scheduled tasks.

Important: 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.

Properties	Description
Selected Agent	Agent for which the Scheduling is displayed. You can display also the scheduling of all agents
Scheduling from ... to ...	Time range for which the scheduling is displayed. Click the  Refresh button to refresh this schedule.
Upd.	This button updates the schedule for the selected agent(s)
Time Range	The time range specified (1 hour, 2 hours) 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.

Zoom in/out	Zooms in the Gantt diagram. Note that you can also zoom by selecting a zone in the diagram.
Gantt Diagram	This panel displays as a Gantt Diagram the schedule for the agent selected
Scenarios details	This panel displays the details and execution statistics for each scheduled scenario.

Using the diagram

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

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

Context

A context is a set of resources allowing the operation or simulation of one or more data processing applications. Contexts allow the same jobs (Reverse, Data Quality Control, Package, etc) to be executed on different databases and/or schemas.

In Oracle Data Integrator, a context allows logical objects (**logical agents**, **logical schemas**) to be linked with physical objects (**physical agents**, **physical schemas**).

Examples:

- The contexts "New York", "Boston" and "San Francisco" represent three data processing sites operating the same software with similar data structures concerning sales and marketing management and logistics. New York also operates an accounting package and a Datawarehouse.
- The "Development" and "Test" contexts allow procedures to be simulated, on replicated databases if necessary.

In the **Designer** and **Operator** modules, the current context is shown in the toolbar of the main window, and can be changed from this toolbar.

Definition

Properties	Description
Name	Name of the context, as it appears in the graphical interface.
Code	Code of the context, allowing a context to be referenced and identified among the different repositories. Note: This code must be unique and the most stable possible. Modifying this code may cause readjustment of the context references from the graphical interfaces of the graphical modules.
Password	Password requested when the user requests to work on this context. This password allows you to restrict the access rights and avoid unwanted context changes.

Note: For "sensitive" contexts such as production, it is strongly recommended that you enter a password when creating the context.

Caution: If the password is not specified, no validation is requested for a change of context.

Default Indicate that this context will be displayed by default in the different lists, and selected when opening **Designer** or **Operator**.

Agents

This tab allows all the logical agents accessible in this context to be displayed and updated.

The left column in the list contains all existing logical agents. To be able to use a logical agent in the given context, you must choose in the right column the physical agent corresponding to this logical agent in the context.

Schemas

This tab allows all the logical schemas accessible in this context to be displayed and updated.

The left column in the list contains all existing logical schemas. To be able to use a logical schema a given context, you must choose in the right column the physical schema corresponding to this logical schema in the context.

Logical Architecture

Logical Schemas

A logical schema is an alias that allows a unique name to be given to all the **physical schemas** containing the same datastore structures.

- The aim of the logical schema is to ensure the portability of the procedures and models on the different physical schemas. In this way, all developments in **Designer** are carried out exclusively on logical schemas.
- A logical schema can have one or more physical implementations on separate physical schemas, but they must be based on **data servers** of the same **technology**. A logical schema is always directly linked to a technology.
- To be usable, a logical schema must be declared in a **context**. Declaring a logical schema in a context consists of indicating which physical schema corresponds to the alias - logical schema - for this context.

For example:

- The logical schema LEDGER is the set of Sybase tables required for the functioning of the accounting application. These tables are stored in a physical schema for each installation of the accounting application. The application was installed once in Boston and twice in Seattle (in production and in test).

Name of the logical schema	Context	Physical Schema
LEDGER	Boston	Sybase Boston LDG

LEDGER	Seattle Production	Sybase SEATTLE PROD LDG
LEDGER	Seattle Test	Sybase SEATTLE TEST LDG

- Work in **Designer** or **Operator** is always done on the logical schema LEDGER. Only the context allows the physical schema on which the operations are actually done to be determined. Thus, the user can switch from one physical environment to another in a single action.

Definition

Property	Description
Name	<p>Name of the logical schema. It is advisable to give a name that suggests the functional content of the schema (program or application name). For example: LEDGER, CRM, ACCOUNTING.</p> <p>Caution: This name is used as id code by the objects contained in the work repositories accessed by other Oracle Data Integrator modules. Uncontrolled modification of this name may cause a manual readjustment of the references from the graphical interfaces.</p>
Contexts	List of the contexts declared in Oracle Data Integrator.
Physical Schemas	Shows the physical schema that corresponds to the logical schema in this context. An undefined value shows that the logical schema does not exist in the context.

Logical Agents

A logical agent is an alias that allows a unique name to be given to all the **physical agents** with the same function in different **contexts**.

- The aim of the logical agent is to simplify going into production and scheduling on several contexts.
- To be usable, a logical agent must be declared in a context. Declaring a logical agent in a context consists in indicating the physical agent it corresponds to in the given context.

Definition

Property	Description
Name	<p>Name of the logical agent. You should put a name that suggests the function of the agent or OS it is based on.</p> <p>Caution: This name is used as id code by the objects contained in the work repositories accessed by other Oracle Data Integrator modules. Uncontrolled modification of this name may cause manual readjustment of the references from the graphical interfaces of the Oracle Data Integrator modules.</p>
Contexts	List of the contexts declared in Oracle Data Integrator.
Physical	Shows the physical agent corresponding to the logical agent in this context. An

Agents undefined value shows that the logical agent does not exist in the context.

Languages

Languages

Oracle Data Integrator uses computer languages to access **technologies**. These languages are used to generate the procedures executed on these technologies. The correct definition of the characteristics of a language is therefore essential for the correct generation and execution of the procedures.

In Oracle Data Integrator, a language is described by its language elements assembled in sub-languages.

The languages are used for:

- Determining the language elements available in the **expression editor**.
- Managing the **separators** and **delimiters** of language words when generating procedures.

Concepts

A language is made up of three types of **words**:

- **Objects**: Named entities handled by the language (Tables, Schemas, Columns)
- **Literals**: Values that the language handles (for example, content of a record column in a table, for SQL language).
- **Reserved words**: which are specific to the language, and are usually the names of functions, commands, etc

Definition

Property	Description
Name	Name of the language
Delimiters	This section allows the characters used as delimiters to be defined for the language.
Objects	<p>Pairs of characters that allow the names of objects to be delimited. This type of character is generally used to protect the case (upper/lower) or to protect the possible separators of special words or characters used in object names.</p> <p>Each pair is made up of a start character and an end character, and there cannot be more than two characters.</p> <p>Examples:</p> <ul style="list-style-type: none"> • <code>[]: [Active Customers]</code>. The square bracket is necessary to protect the space and the case in the name of the object Active Customers. • <code>" " : "Account"."Licenses"</code>. The quotation marks are used to protect the accents and the case in the name of the objects Account (schema) and Licenses (Datastore).

Literals	<p>Pairs of characters allowing literals (values) to be delimited.</p> <p>Each pair is made up of a start character and an end character, and there cannot be more than two characters.</p> <p>For example:</p> <ul style="list-style-type: none"> ' ' : 'Welcome to Paris !'. The quotation marks are used to protect the accents, the case and the special characters in this literal.
Case sensitive	Determines the case-sensitivity for the language.
Objects	Indicates whether the language differentiates upper and lower case for the names of the objects it processes, such as the names of columns, tables, and schemas.
Reserved keywords	Indicates whether the language differentiates upper and lower case for its reserved words , such as SELECT or INSERT.
Word separators	The characters used to separate the words of the language.

Sub-languages

A sub-language is a group of language elements which all have a common type, a use within a language.

For example:

- The sub-language "Aggregation" groups all the aggregation operators in SQL language.

Definition

Property	Description
Name	Name of the sub-language.
Type	Types of language elements in this sub-language. The possible types are: <ul style="list-style-type: none"> F: Function, i.e. elements accepting parameters and returning values. O: Operators, that put two words in relation. KW: Key words of the language.
Used in a procedure	Shows that elements of this sub-language can be used in procedures.
Used in the Mapping	Shows that elements of this sub-language can be used in mapping.
Used in the From clause	Shows that elements of this sub-language can be used in FROM clauses.
Used in the Filter	Shows that elements of this sub-language can be used in filters.
Color	The color of the elements of this sub-language when displayed in the expression editor .

Language Elements

A language element is an operator, a function or a keyword of a language. This language element can have several implementations depending on the technology the language is implemented on.

For example:

- The function that returns the current date in an SQL language query is declared as the language element `CURDATE`, but is implemented in the technologies supporting SQL language as `TODAY` (Sybase, Informix) or `CURDATE` (Progress, DB2/400, etc).

Definition

Property	Description
Name	Name of the language element.
Expression	Generic syntax of the language element. The implementation syntax for a technology is described on the implementation tab. However, if no implementation exists for the technology and if the latter must have access to the language element (Universal or Standard -type), then this is the default syntax used by the expression editor .
Group function	Shows that this language element handles a group of elements. (Some examples: MAX, MIN, etc)
Universal	Shows that this element is a universal type and should be inherited by all technologies supporting this language.
Standard	Shows that this element is a standard type and should be inherited by all technologies with the compatible box checked for this language.
Help	Help text and description of the element.

Implementation

Property	Description
Name	Name of the implementation of the language element.
Expression	The syntax of the language element in this implementation. This syntax is the one displayed in the expressions editor .
Technology	Technology in which this implementation takes place.
Exception	If this box is checked, then the language element does not exist for the given technology, even if it is universal or standard -type.

Repositories

Master Repository

The **Master Repository** is a data structure containing information on the topology of a company's IT resources, on security and on version management of projects and data models. This repository is stored on a relational database accessible in client/server mode from the different modules.

Generally, only one master repository is necessary.

However, in exceptional circumstances, it may be necessary to create several master repositories in one of the following cases:

- Project construction over several sites not linked by a high-speed network (off-site development, for example).
- Necessity to clearly separate the interfaces' operating environments (development, test, production), including on the database containing the master repository. This may be the case if these environments are on several sites.

Master repository domains

The master repository has two **functional domains**:

- **Topology**: This domain is compulsory. It contains the description of the technologies, the data servers and the agents. The information in this domain can be modified through the Topology Manager module.
- **Security**: This domain is compulsory. It contains Oracle Data Integrator internal metadata, and the storage structure for information about the users and their privileges. The information in this domain can be modified through the Security Manager module.

Definition

Property	Description
Name	Name of the master repository.
External id code	Unique ID code for the repository.
Successful Installation	Shows if the installation has finished successfully. If this indicator is not positioned, it is probably because the installation has not finished correctly.
Repository Version	Version of the master repository.
Connection	With this button, you can open the information for connecting to the master repository. This connection information is that of a data server. For more information on connecting to the master repository, refer to the section Creating repositories. Caution: Be cautious when modifying a repository's connection information. This operation which can cause issues in your Oracle Data Integrator installation.

Work Repository

The Work Repository is a data structure containing information about data models, projects, and their operation. This repository is stored on a relational database accessible in client/server mode

from the different Oracle Data Integrator modules.

Several work repositories can, if necessary, be declared with several master repositories.

However, a work repository can be linked with only one master repository for version management purposes.

A work repository contains several **functional domains**:

- **Execution:** This domain is compulsory, it allows launching and monitoring of job operations executed locally or by an agent. This domain is generally accessed through the **Operator** module. An **operating repository** contains only this domain.
- **Project/Model:** This domain is optional, it allows management of data models and projects (interfaces, procedures, etc). This domain is generally accessed through the **Designer** module. A **development repository** contains this domain as well as the execution domain.

General Properties

Property	Description
ID	<p>ID code for the work repository.</p> <p>Important note: You are strongly advised to give a unique id code (across the whole information system) for each work repository when it is created. This unique id code allows:</p> <ul style="list-style-type: none"> - work repositories to be referenced on several master repositories without conflict. - objects in this repository to be referenced on the master repository for version management (by a company domain), - objects to be transferred between work repositories without conflict. Since the unique id code for the work repository enters the composition of the id codes of these objects, this ensures their uniqueness among repositories. <p>For greater security, we recommend creating all work repositories from the same master repository.</p>
Name	Name of repository.
External id code	The unique ID code of the repository.
Type	Type of work repository. The possible types are Execution or Development .
Connection	<p>With this button, you can open the information for connecting to the work repository. This connection information is that of a data server. For more information about connecting to the work repository, refer to the section Creating repositories.</p> <p>Caution: Be cautious when modifying a repository's connection information. This operation which can cause issues in your Oracle Data Integrator installation.</p>

Hosts

Oracle Data Integrator gives you access to a number of module (Designer, Topology, etc.). Hosts and Usages let you manage access to these modules.

Definition

Property	Description
Machine Name	Unique name identifying the machine. It is usually the machine's network name.
IP Address	IP address of the machine on the network common to the machine and the Repository server.

Usage

Property	Description
Module	Module name.
Usage Type	Type of usage: <ul style="list-style-type: none">• Always allowed: The host can always use the given module.• Denied: The host cannot use the given module.• Automatic (default behaviour): The host automatically takes access to the given module when connecting to the repository with this module.
Last usage	Date and time of the last use of this module on this host.

Security Manager

Introduction to Security Manager

Using the **Security Manager** module, you can manage security in Oracle Data Integrator. The Security Manager module allows **users** and **profiles** to be created. It is used to assign user rights for **methods** (edit, delete, etc) on generic **objects** (data server, datatypes, etc), and to fine-tune these rights on the object **instances** (Server 1, Server 2, etc).

The Security Manager module stores this information in a master repository. This information can be used by all the other modules.

- Define the Security Policy

Security Manager's Interface

The Security Manager GUI appears as follow:

The Menu

The **Menu** contains pull-down menus to access the following features:

- Import/Export
- Wizards
- Display options
- Open modules or tree views
- Change the user's password and options

The Toolbar

The **Toolbar** lets you:

- Open other modules
- Refresh the Tree views
- Open the on-online help

The Tree Views

Security Manager objects available to the current user are organized into the following tree views:

- The **objects**, describing each Oracle Data Integrator elements type (datastore, model,...),
- The users **profiles** and their **authorizations**,
- the **users** and their **authorizations**.

Each tree view appears in a floatable frames that may be docked to the sides of the main window. These frames can be also be stacked up. When several frames are stacked up, tabs appear at the bottom of the frame window to access each frame of the stack.

Tree view frames can be moved, docked and stacked by selecting and dragging the frame title or tab. To lock the position of views, select **Lock window layout** from the **Windows** menu.

If a tree view frame does not appear in the main window or has been closed, it can be opened using the **Windows > Show View** menu.

It is possible in each tree view to perform the following operations:

- Insert or import root objects to the tree view by clicking the appropriate button in the frame title
- Expand and collapse nodes by clicking on them
- Activate the methods associated to the objects (Edit, Delete, ...) through the popup menus
- Edit objects by double-clicking on them, or by drag and dropping them on the **Workbench**.

The Workbench

The windows for object being edited or displayed appear in the **Workbench**.

Connection

Connecting to the Master Repository




With this window, a connection to a master repository can be selected when opening the **Topology Manager** or **Security Manager** module. The creation, modification or deletion of existing connections can also be launched.

Note: The definitions of the master repository connections are stored locally in the `/bin/snps_login_security.xml` file

General Properties

Properties	Description
Login name	List of the defined connections to the master repositories.
User	The default Oracle Data Integrator user for connecting to the selected master repository. If no default user has been specified, this must be completed.
Password	The password corresponding to the user for the connection. If no default password has been specified, this must be completed.

Toolbar

Button	Description
	Allows a new connection to a master repository to be created.
	Allows the connection selected in the login name field to be modified
	Allows the connection selected in the login name field to be deleted

Master Repository Connection Details

With this window, a connection to a **master repository** can be defined or modified.

Note: The definitions of the master repository connections are stored locally in the `/bin/snps_login_secu.xml` file

General Properties

Properties	Description
Oracle Data Integrator Connection	With this information group, the properties displayed in the connection window can be defined.
Login name	This is the name presented in the list when a Data Integrator module is opened.
User	The default Oracle Data Integrator user for connecting to the master repository. If no default user has been specified, the user must enter it each time this connection is used.
Password	Password corresponding to the Oracle Data Integrator user. If no default password has been specified, the user must enter it each time this connection is used.
Database Connection (Master Repository)	This information group shows the connection parameters for the relational database that hosts the master repository.
User	User allowing you to connect to the schema of the database containing the master repository.
Password	The data server password for the database user.
Driver List	Type of technology the master repository is based on. You can obtain more information about the technology and the driver used by clicking on the button. For more information, refer to Driver details.
Driver name	Name of the JDBC driver used for connecting to the data server of the master repository. When a technology is selected, Data Integrator displays the most frequently used driver for this technology. Nevertheless, a different one can be used. For this, this driver must be correctly installed. For more information, refer to Installing JDBC / JMS drivers in the installation guide.
Url	URL for connecting to the data server of the master repository. You can select a URL model for the driver used by clicking on the button. For more information, refer to URL Samples. For more information on drivers and URLs, refer to the section JDBC URL Sample.
Default Connection	If this box is checked, when an Oracle Data Integrator graphic module is started, the current connection will be proposed by default.

Driver Details

This window provides information about the driver used with a technology.

General Properties

Properties	Description
Name	The driver used for the selected technology.
Description	Information about the driver.

For more information on drivers and URLs, refer to the section JDBC URL Sample.

URL Samples

This window provides URL models for a driver.

General Properties

Properties	Description
Name	List of valid URL models for the selected driver.
Description	Details on the URL model.

For more information on drivers and URLs, refer to the section JDBC URL Sample.

Objects

Object

An object is a representation of an element that can be handled through Oracle Data Integrator (agents, models, datastores, etc). The objects are the visible part of Oracle Data Integrator object components (Java classes). It is necessary to bring together the notions of object and object instance (or instance), which in Oracle Data Integrator are similar to Object-Oriented programming notions. An instance (object instance) is attached to an object type (an object). For example, the project MY_PROJ_1 is an instance of the project object type (or object). Similarly, another instance of a project-type object is YOUR_PROJ_2.

Definition

Property	Description
Name	Label of the object that appears in the graphical interface.

Flexfields

Flexfields are user-defined fields enabling to customize the properties of Oracle Data Integrator' objects. Flexfields are set for each object instance through the **Flexfield** tab of **Object** window, and their values are defined for each object instance through the **Flexfield** tab of the object's instance window. Flexfield values are usable through Oracle Data Integrator substitution methods.

Important: Flexfield exist only for certain object type. Objects that do not have a flexfield tab do not support them.

Property	Description
Name	Label of the flexfield as it will appear in the Flexfield tab of the object instance window.
Code	Code of the Flexfield.
Technology	Technology for which the flexfield will be activated. For instance, a Flexfield on a procedure will only appear for procedures using this technology.
Type	Type of the flexfield (String or Numeric).
Default	Default value of the Flexfield.

Method

A method is a type of action which can be performed on an object. Each object has a series of methods that are specific to it. The notion of method in Oracle Data Integrator is similar to that of Object-Oriented programming.

Definition

Property	Description
Name	Label of the method that appears in the graphical interface.
Internal Java name	Name of the Oracle Data Integrator internal Java method implemented on this object.
Object creation method	Shows that this method contributes to creating an object instance
Default method granted for the object creator	If this box is checked, the object creator is automatically entitled to the method. Note: This checkbox takes effect only for objects for which authorizations by object instance are supported.
Object Concerned	Internal id code of the object

Profiles

Profile

A profile represents a generic rights model for working with Oracle Data Integrator. One or more profiles can be assigned to a user.

Definition

Property	Description
Name	Label of the profile that appears in the graphical interface.

Predefined profile types

Profile	Description
CONNECT	Minimum generic profile for connecting to Oracle Data Integrator. All users must have at least this profile.
DESIGNER	Generic profile for interface developers, or users working on the interfaces. This profile give access to any project and project sub components (folders, interfaces, knowledge modules, etc.) stored in the repository. It also authorize users to perform journalizing actions (start journal, create subscriber, etc.) or to run static controls over models and datastores.
METADATA ADMIN	Generic profile for users responsible for managing models and reverse-engineering. Users having this profile are allowed to browse any project in order to select a CKM, RKM or JKM and to attach it to a specific model.
OPERATOR	Generic profile for operators. It allows users to browse execution logs.
REPOSITORY EXPLORER	Generic profile for meta-data browsing through Metadata Navigator. It also allows scenario launching from Metadata Navigator.
SECURITY ADMIN	Generic profile for administrators of user accounts and profiles.
TOPOLOGY ADMIN	Generic profile for users responsible for managing the information system topology. Users granted with this profile are allowed to perform any action through the Topology Manager module.
VERSION ADMIN	Generic profile for managing component versions as well as solutions. This profile must be coupled with DESIGNER and METADATA ADMIN.
NG DESIGNER	Non-Generic profile for DESIGNER.
NG METADATA ADMIN	Non-Generic profile for METADATA ADMIN.
NG REPOSITORY EXPLORER	Non-generic profile for meta-data browsing through Metadata Navigator.
NG VERSION ADMIN	Non-generic profile for VERSION ADMIN. It is recommended that you use this profile with NG DESIGNER and NG METADATA ADMIN.

Generic vs Non-Generic profiles

Generic profiles have the **Generic privilege** option checked for all objects' methods. This implies that a user with such a profile is **by default** entitled for all methods of all instances of an object to which his profile is entitled.

Non-Generic profiles is **not by default** entitled for all methods on the instances, as the **Generic privilege** option is unchecked for all objects' methods. The administrator must grant the user the rights for the methods for each instance.

If an administrator wants a user to have the rights on no instance by default, but wishes to grant the rights by instance, he must grant the user with a non-generic profile.

If an administrator wants a user to have the rights on all instances of an object type by default, he must grant the user with a generic profile.

The following operations are possible on a profile:

- Creating a new profile
- Assigning a profile to a user
- Assigning an authorization by profile
- Deleting an authorization by profile
- Removing a profile from a user
- Deleting a profile

Authorization by Profile

An authorization by profile is placed on a method of an object for a given profile. It allows a user with this profile to be given, either optionally or automatically, the right to this object via the method.

The presence of an authorization by profile for a method, under an object in the tree of a profile, shows that a user with this profile is entitled (either optionally or automatically) to this object's instances via this method.

The absence of authorization by profile shows that a user with this profile cannot in any circumstances invoke the method on an instance of the object.

Definition

Property	Description
Genetic privilege	When this box is checked, the user with the profile has, by default (automatically), the right to the method on all the instances of this object. When it is not checked, he does not have it by default. The privilege must be shown case by case by an authorization by object instance.

The following operations are possible on authorizations by profile:

- Assigning an authorization by profile
- Deleting an authorization by profile

Users

User

A user in the **Security Manager** module represents an Oracle Data Integrator user, and corresponds to the login name used for connecting to a repository.

A user inherits the following rights:

- The profile rights he already has
- Rights on objects
- Rights on instances

Definition

Property	Description
Name	User name - Used as Login name
Initials	The user's initials (used to identify him when objects in the Repository are modified)
Password	The user's password
Input/Change the password	This button opens a window to input and confirm the user's password.
Notes	This is a free text zone for entering information about the user.
Account Expiry	This section allows you to program a date for this account to expire. After this date, the account will require a reactivation by an administrator.
Supervisor Access Privileges	The Supervisor box amounts to giving ALL rights to a user, as it can automatically override the rights given to him via Security manager. Users with Supervisor Access Privileges only have access to this checkbox. Caution! The Supervisor privilege should only be given to a minimum number of users. It is given by default to the user SUPERVISOR in Oracle Data Integrator. It is advisable to keep this privilege only on this account, which should be used mainly for maintenance and administration purposes.

The following operations are possible on a user:

- Creating a new user
- Assigning a profile to a user
- Assigning an authorization by user
- Assigning an authorization by object instance
- Deleting an authorization by object instance
- Deleting an authorization by user
- Removing a profile from a user
- Deleting a user

Authorization by User

An authorization by user is placed on a method of an object, for a given user. It allows the user to be given, either optionally or automatically, the right to this object via the method.

The presence, under an object in a user's tree, of an authorization by user for a method shows that the user has the right (either optionally or automatically) to the instances of this object via this method.

The absence of authorization by user shows that the user cannot in any circumstances invoke the method on an instance of the object.

Definition

Property	Description
Generic privilege	When this box is checked, the user has, by default (automatically), the right to the method on all the instances of this object. When it is not checked, he does not have it by default. The privilege must be added for each instance by an authorization by object instance.

The following operations are possible on an authorization by user:

- Assigning an authorization by user
- Deleting an authorization by user

Authorization by Object Instance

An authorization by object instance is placed on an object instance for a given user. It allows the user to be given rights to certain methods on this object instance.

The presence in a user's tree of an authorization by object instance for an instance shows that the user could have specific rights on the object's methods for the given instance (these rights are specified in the window). If an instance is not in the tree of the user's **instances**, then the authorizations by user or by profile on the object that has given the resulting instance apply.

Authorizations by object instances may be defined by work repository. An object instance may be replicated between work repositories (using export/import in *Synonym* modes) or may be usable in different work repositories (such as context) . When such an instance is "usable" in several repositories, privileges may be granted/denied in all repositories, in no repository, or in selected repositories.

For example, It is common to replicate projects from a *Development* repository to a *Test* repository. A developer may be granted *edit* privileges for his project in the *Development* repository, but not on the *Test* repository. On the *Test* repository, he will only be granted with *view* on the project.

Note: A number of object types support the authorization by object instance. These object types are listed under the **Instances** node for each user.

Security

This window contains a list of objects and associated methods. It contains not only the methods for the instance the authorization applies to, but also the methods for the object instances found under this instance in the tree view.

No default selection is made in the list. To perform a selection, click the lines in the table. You can select a group of items by pressing the <SHIFT> key, or perform multiple selections by keeping the <CTRL> key pressed.

Property	Description
Object	Object type to which the privilege applies.
Method	Name of the object method.
Active	This field defines how the method is granted: <ul style="list-style-type: none">• Always: it is granted for all work repositories attached to this master, even the future ones.• Never: It is denied for all work repositories attached to this master, even the future ones.• By Repository: It is granted for the work repositories select in the Repository Field, and denied for other repositories.
Repository	This field contains either a list of work repositories, or <i>Not defined by the user</i> , meaning that this privilege is Allowed/Denied (Always/Never) in all repositories.

For example: If the instance MODEL1 of the "Model" object is inserted, then the methods for the objects Datastores, Condition, etc are displayed. If for the "Delete" method of the "Datastore" object the **Active** field is set to **Always**, then the user is granted the right to delete in all the datastores in MODEL1 in all repositories.

Note: Any method for which the user has generic rights is not listed in the authorization by object instance window.

The following operations are possible on an authorization by object instance:

- Assigning an authorization by object instance
- Deleting an authorization by object instance

Date Format

Oracle Data Integrator manages internally dates in the java format. When a date format must be specified, to recognize a formatted date in a string or to display a date, a date format **pattern** must be specified.

The pattern is a string associating **symbols** (letters representing parts of the date) and free text, eventually surrounded by delimiters. All characters specified in the pattern that are not letters will be reproduced as is, even if they are not surrounded by delimiters.

Each symbol has a defined **presentation** (text or numeric). For instance, a year (2004) may be presented as a numeric, and a day of the week (monday) as a text.

Available Symbols

Symbol	Description	Presentation	Example
G	Era	Text	AD
y	Year	Numeric	1996
M	Month of the year	Text or Numeric	July or 07
d	Day of the month	Numeric	10
h	Hour (1-12)	Numeric	12
H	Hour (0-23)	Numeric	23
m	Minute in the hour	Numeric	55
s	Second in the minute	Numeric	30
S	Millisecond	Numeric	978
E	Day of the week	Text	Thursday
D	Day of the year	Numeric	189
F	Day of the month	Numeric	2 (2nd Wednesday)
w	Week of the year	Numeric	27
W	Week in the month	Numeric	2
a	AM/PM	Text	PM
k	hour of the day (1-24)	Numeric	24
K	Hour of the day (0-11)	Numeric	0

z	Zone	Text	Pacific Standard Time
'	Escape for free text	N/A	
''	Free text delimiter	N/A	

The number of symbol letters you specify also determines the result, depending on the presentation. For example, if the "zz" pattern results in "PDT," then the "zzzz" pattern generates "Pacific Daylight Time." The following table summarizes these rules:

Presentation	Number of symbols	Result
Text	1-3	Abbreviated form.
Text	>= 4	Full form.
Number	Minimum number of digits is required	If the number of symbols is too long, the form is padded with zeros, or truncated if the number of symbols is too short.
Text & Number	1-2	Number form.
Text & Number	>=3	Text form.

Examples

Format	Result (with a Local US configuration)
yyyy.MM.dd G 'at' hh:mm:ss z	1996.07.10 AD at 15:08:56 PDT
EEE, MMM d, 'yy	Wed, July 10, '96
h:mm a	12:08 PM
hh 'o'clock' a, zzzz	12 o'clock PM, Pacific Daylight Time
K:mm a, z	0:00 PM, PST
yyyyy.MMMMM.dd GGG hh:mm aaa	1996.July.10 AD 12:08 PM

JDBC URL Samples

Oracle Data Integrator does not include any licenses for JDBC Drivers. Some drivers are provided free if you have already purchased certain server products.

A list of all available JDBC drivers is maintained at the following URL:
<http://java.sun.com/products/jdbc/jdbc.drivers.html>

Not all drivers have been validated for use with Oracle Data Integrator. Below is a list of drivers that have been tested and are recommended for use with Oracle Data Integrator.

Driver	Driver and URL parameters, comments
<p>Oracle Data Integrator Driver for File (new version)</p>	<p>driver: <code>com.sunopsis.jdbc.driver.file.FileDriver</code></p> <p>URL: <code>jdbc:snps:dbfile?<option1>&<option2>&...</code></p> <p>Options:</p> <ul style="list-style-type: none"> ENCODING=<encoding_code>: File encoding. The list of supported encodings is available at http://java.sun.com/j2se/1.4.2/docs/guide/intl/encoding.doc.html. The default encoding value is ISO8859_1. TRUNC_FIXED_STRINGS=TRUE FALSE: Truncates strings from fixed-column-width files to the field size. Default value is FALSE. TRUNC_DEL_STRINGS=TRUE FALSE: Truncates strings from delimited files to the field size. Default value is FALSE. OPT=TRUE FALSE: Optimizes file access on multiprocessor machines for better performance. Using this option on single processor machines may actually decrease performance. Default value is FALSE. <p>Warning: Although this driver is not fully JDBC compliant, it supports certain methods used by the agent</p>
<p>Oracle Data Integrator Driver for File (old version - deprecated)</p>	<p>Warning: This driver version is deprecated.</p> <p>driver: <code>com.sunopsis.jdbc.driver.FileDriver</code></p> <p>URL: <code>jdbc:snps:file</code></p> <p>Other URL:</p> <ul style="list-style-type: none"> <code>jdbc.snps.file?CHARSET_ENCODING=<encoding>&RAW_HEADERS=true false</code> <ul style="list-style-type: none"> CHARSET_ENCODING enables you to indicate the encoding of the target file. The list of supported encodings is available at the URL: http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html RAW_HEADERS enables you to read the header line as raw data. This lets you retrieve special characters in this line to generate column names during a reverse-engineering. <p>Warning: This driver is not fully JDBC compliant, it supports certain methods used by the agent</p>
<p>Sun ODBC/JDBC Bridge (JDBC 2 Type 1)</p>	<p>driver: <code>sun.jdbc.odbc.JdbcOdbcDriver</code></p> <p>URL: <code>jdbc:odbc:<odbc datasource name></code></p> <p>Note: You must have installed an ODBC Driver and created an ODBC datasource with the ODBC Administrator before using the type 1 JDBC Driver.</p>
<p>Oracle JDBC Driver (JDBC 1 type 4)</p>	<p>driver: <code>oracle.jdbc.driver.OracleDriver</code></p> <p>URL: <code>jdbc:oracle:thin:@<IP address or name>:<listener port>:<SID></code></p> <p>Note: The SQL*Net client is not required on the client machine.</p>
<p>Inet Software JDBC Driver for Microsoft SQL/Server</p>	<p>driver: <code>com.inet.tds.TdsDriver</code></p> <p>URL: <code>jdbc:inetdae:<IP address or name>:<listener port></code></p> <p>Note: Open Client and ODBC are not required on the client machine.</p>

(Type 4) This driver loads each result set into the client's virtual memory before fetching rows.

Other sub-protocols:

- jdbc:inetdae: classic
- jdbc:inetdae6: SQL Server 6.5 compatible mode (on Sprinta Driver)
- jdbc:inetdae7: supporting of the SQL Server 7.0 (or higher) (On Sprinta Driver)

Other URLs:

- jdbc:inetdae:hostname:portnumber
- jdbc:inetdae:hostname -> with default port 1433
- jdbc:inetdae:hostname:portnumber?database=MyDb&language=deutsch
-> with properties
- jdbc:inetdae://servername/pipe/pipename -> with named pipes

Sybase
JConnect
(Type 4)

driver: com.sybase.jdbc2.jdbc.SybDriver

URL: jdbc:sybase:Tds:<IP address or name>:<listener port>/<database>

Note: DB-LIB, CT_LIB and ODBC are not required on the client machine.

Warning: Array Fetch is not available if you do not have a Unique Index on Table (leave the Array Fetch empty in Oracle Data Integrator). Batch Update is available.

Warning: You must run the correct SQL script sql_server.sql (provided with the driver distribution) on sql/server before use.

Hit JDBC
Driver for
AS/400 via
Client/Access
(Type 4)

driver: hit.as400.As400Driver

URL: jdbc:as400://<IP address or name>

Notes: No extra middleware is required on the client machine. ALL Client/Access services must be started on AS/400 (STRHOSTSVR *ALL)

Other options:=

```
[;user=<user>]
[;password=<password>]
[;options=[secure,][http][<Compressorclass>,[Cryptographerclass],]
[;license=<serial number>]
[;libraries=<lib1,lib2,...,libn>]
[;ccsid=<number>]
[;use_packages=<YES|NO>]
[;use_package_library=<package library>]
[;use_package_name=<package name>]
[;allow_package_update=<YES|NO>]
[;convert_ccsid_65535=<YES|NO>]
[;fetch_block_size=<Ksize>]
[;naming=<sql|system>]
```

IBM
TOOLBOX

driver: com.ibm.as400.access.AS400JDBCDriver

URL: jdbc:as400://<IP address or name>

JDBC Driver
for AS/400
(Type 4)

Note: No extra middleware is required on the client machine.

Oracle Data
Integrator
AS/400
Driver
Wrapper

driver: `com.sunopsis.jdbc.driver.wrapper.SnpsDriverWrapper`

URL:

`jdbc:snps400:<machine_name/IP_address>[;param1=valeur1[;param2=valeur2...]`
]

This driver wrapper enables the use of the Native of JT/400 driver depending on the situation. This driver wrapper connects using the Native driver when a connection to the local AS/400 machine (the one where the agent is runs) is requested. Otherwise it uses the JT/400 driver. This wrapper uses 3 Java properties:

- `HOST_NAME`: Comma-separated list of names for the local AS/400 machine.
- `HOST_IP`: IP address of the local AS/400 machine. It is used if the connection is requested with the IP address.
- `SNPS_VERBOSE`: Driver trace level. Values: 0 (no trace) to 5 (full trace).

For more information, see [Creating a DB2/400 Data Server](#).