Oracle® Product Data Quality

Application Studio Reference Guide

Version 5.6

E20596-01

January 2011

ORACLE®

Oracle Product Data Quality Application Studio Reference Guide, Version 5.6

E20596-01

Primary Author: Lorna Vallad

Contents

5 Decision Map Builder

6 Advanced Mapping and DSA Concepts

Preface

This reference guide is intended to explain the basic capabilities of the Oracle Product Data Quality Application Studio. The document is organized as follows:

- Chapters 1 through 5 describe the basic application features.
- Chapter 6 describes more advanced features and functionality.

To understand all of the advanced features presented, you must use this reference guide in conjunction with the Oracle Product Data Quality documents listed in "Related Documents" on page -ix.

Review the following Oracle Product Data Quality documentation prior to the use of this manual is recommended:

- *Oracle Product Data Quality Governance Studio Reference Guide*
- *Oracle Product Data Quality Knowledge Studio Reference Guide*

You must have the Oracle Product Data Quality client software installed on your computer including all of the sample files.

In addition, Oracle Product Data Quality Application Studio training is encouraged.

Audience

You should have a basic understanding of the DataLens Technology. Including the functionality of the Oracle Product Data Quality Knowledge Studio and the Oracle Product Data Quality Governance Studio applications.

This document is intended for all users of the DataLens Technology, including:

- Business Analysts
- Subject Matter Experts (SMEs)
- IT Administrators
- Application/Solution Owners

Related Documents

For more information, see the following documents in the documentation set:

- The *Oracle Product Data Quality Oracle DataLens Server Installation Guide* provides detailed Oracle Product Data Quality Oracle DataLens Server installation instructions.

- The *Oracle Product Data Quality Oracle DataLens Server Administration Guide* provides information about installing and managing an Oracle DataLens Server.
- The *Oracle Product Data Quality COM Interface Guide* provides information about installing and using the Oracle DataLens Server COM APIs.
- The *Oracle Product Data Quality Java Interface Guide* provides information about installing and using the Oracle DataLens Server Java APIs.
- The *Oracle Product Data Quality AutoBuild Reference Guide* provides information about creating initial data lens based on existing product information and data lens knowledge.
- The *Oracle Product Data Quality Knowledge Studio Reference Guide* provides information about creating and maintaining data lenses.
- The *Oracle Product Data Quality Governance Studio Reference Guide* provides information about creating and maintaining Data Service Applications (DSAs).
- The *Oracle Product Data Quality Glossary* provides definitions to commonly used Oracle Product Data Quality technology terms.
- The *Oracle Product Data Quality Services for Excel Reference Guide* provides information about creating a DSA based on data contained in a Microsoft Excel spreadsheet.
- The *Oracle Product Data Quality Task Manager Reference Guide* provides information about managing tasks created with the Task Manager or Governance Studio applications.

See the latest version of this and all documents listed at the Oracle Product Data Quality Documentation Web site at:

`http://download.oracle.com/docs/cd/E20593_01/index.htm`

Conventions

The following text conventions are used in this document:

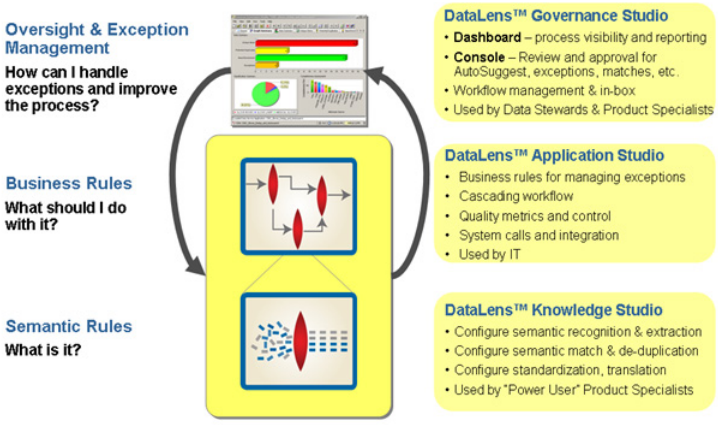
Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, text that you enter, or a file, directory, or path name.
`monospace`	Boldface, monospace type indicates commands or text that you enter.

1

Overview

Oracle DataLens Server is built on industry-leading DataLens™ Technology to standardize, match, enrich, and correct product data from different sources and systems. The core DataLens Technology uses patented semantic technology designed from the ground up to tackle the extreme variability typical of product data.

Oracle Product Data Quality uses three core DataLens Technology modules: Governance Studio, Knowledge Studio, and Application Studio. The following figure illustrates the process flow of these modules.



Data Service Applications (DSAs) leverage semantic knowledge to enable data centric business applications by controlling the flow of data and processes for a given task. While the data lens contains detailed knowledge of specific items and products, the DSA is a much broader tool. It allows you to define data input and output, and routes data from one processing step to the next. In other words, a DSA defines the flow of data and processing steps used to accomplish a task.

The Application Studio is a graphical user interface (GUI) tool that you use to build DSAs. Using drag-and-drop techniques, you define and sequence the elements of a process, which include some or all of the following:

Preprocessing Steps

These steps allow you to do preliminary tasks, like setting up temporary database tables.

Input Data Sources

DSAs can accept data input from spreadsheets, text files, database sources, or XML files.

Process Steps

These are the basic units of work performed by a DSA, and they are contained in objects called Transformation Maps. Transformation Maps perform a wide range of tasks, including calls to data lenses, updates of database tables and string manipulation. Broadly speaking, Transformation Maps serve as core steps that are executed as part of the normal process flow or as alternative steps that perform exception handling.

There are two basic forms of maps that can be used in a DSA, **Transformation Maps** and **Decision Maps**.

- Transformation Maps are the standard maps for content transformation to define how the data is processes and reformed.
- Decision Maps have branch points similar to the 'if, then, else' functionality of programming languages. These maps use the quality metrics for each record as branch points to transfer control to different data lenses, classification schemas, other databases, etc. to extract or modify other information needed for the definition of a business process.

Outputs

DSAs can output data in a variety of forms compatible with many destinations, including Microsoft Excel spreadsheets, databases, XML files, or e-mail messages. One DSA can include multiple outputs defined at various points in the process flow.

Post-processing Steps

These steps allow you to define clean-up tasks, such as dropping temporary database tables or performing other end-of-job housekeeping.

Starting the Software

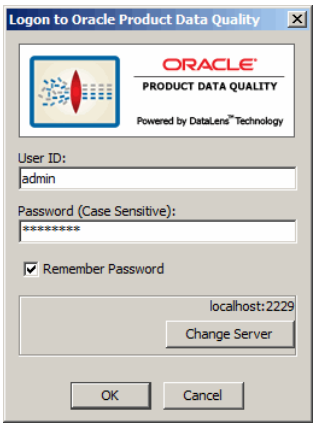
You can start Oracle Product Data Quality by using either the desktop shortcut or the Windows **Start** menu as follows:

> **Note:** If Oracle Product Data Quality is not installed, you can install it using the instructions in Appendix A, "Installing the Client Software."

- Double-click the desktop shortcut.

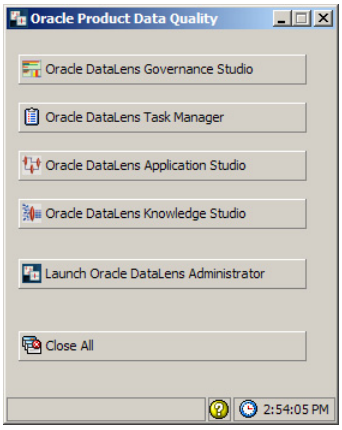
- Click **Start, Programs**, **Oracle Product Data Quality**, and select **Oracle Product Data Quality**.

The **Oracle Product Data Quality Login** dialog box appears.



Enter your user name and password and click **OK**. You can avoid entering your password every time you logon by selecting the **Remember Password** check box. If you want to change your Oracle DataLens Server, click **Change Server** to select a new server.

The **Oracle Product Data Quality Launch Pad** appears.



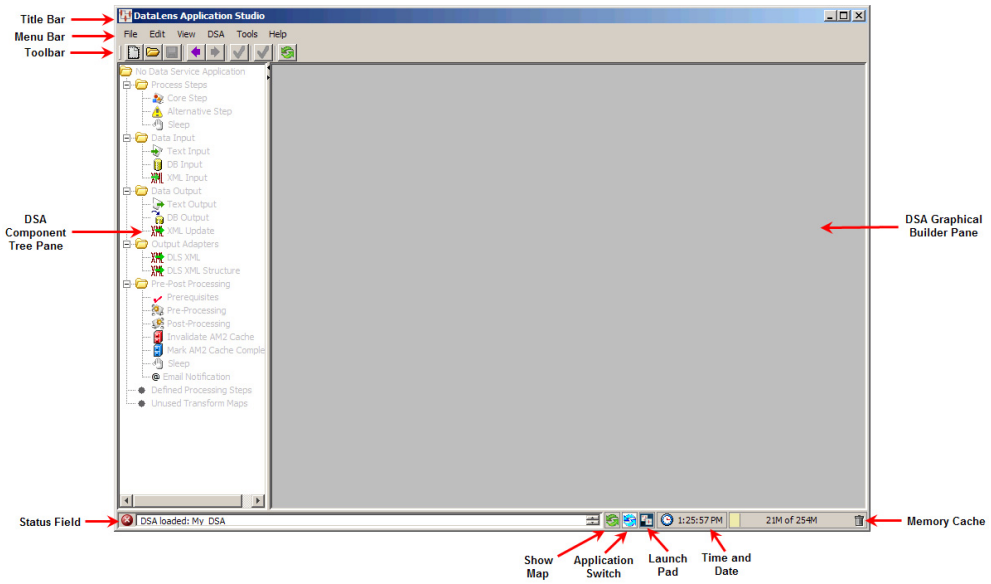
The **Oracle Product Data Quality Launch Pad** allows you to quickly start any of the Oracle DataLens Server applications by clicking on any of the buttons. You can close all open Oracle Product Data Quality applications using the **Close All** button.

Click the **Oracle Application Studio** button to start the application.

Understanding the Client Workspace

The Application Studio graphical user interface (GUI) provides the client workspace used to create and manage a DSA. It is comprised of two separate GUI workspaces:

- The **DSA Builder**, which is the default when the Application Studio is opened, that allows you to build a DSA.
- The **Transformation Map Builder**, which allows you construct the output processing instructions for the DSA.



This section describes the following client workspace functionality:

Frame Functionality

The Application Studio client workspace frame contains useful information and interactive functions including the following:

Title Bar

Indicates the current application and open DSA or Transformation Map.

Status Field

Provides the status of the DSA or Transformation Map one line at a time. Though this field cannot be resized, the scroll arrows on the right-hand side can be used to view all available status information. The status data does not change based on the selected tab; rather it is a compilation of all data.

Show

Changes the view between the DSA Builder and Transformation Map Builder client workspaces.

Application Switch

Returns you to the last Oracle Product Data Quality application used.

Oracle Product Data Quality Launch Pad

This button opens the Oracle Product Data Quality Launch Pad so that you can select other applications.

Time and Date

The time is displayed and when you hover over this field, the date displays.

Memory Cache

Indicates the amount of memory cache currently used and the total amount allowed. You can dump the memory cache by clicking on the trash can icon in this interactive field. This functionality does not appear in the Transformation Map Builder workspace.

Note: This feature is only used for system diagnosis and should not be used unless requested by Oracle Support.

Menus and Toolbars

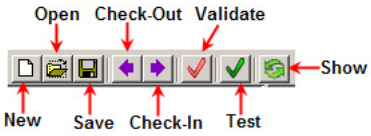
The Application Studio menus and toolbars allow easy access to the most frequently used Application Studio functions. Though the set of toolbar buttons remains the same during user interface operation the buttons are enabled or disabled based the current state of you interface and options set. Buttons displayed with shades of gray are disabled. Full-color buttons are enabled. All toolbar buttons are standard push buttons, requiring a single click of the mouse to activate.

The Application Studio GUI menus provide access to most Application Studio functions. All of the buttons on the toolbar have a corresponding menu command, which are indicated on each menu with the button icon displaying adjacent to the command. The set of menu commands remains the same during the GUI operation.

Menu commands are enabled or disabled based on the current state of the DSA; commands that are dimmed are unavailable. Some menu commands perform functions that are more complex and are indicated by an ellipsis symbol (...). These commands open dialog boxes to collect information needed to complete the requested function. Menu commands that toggle user functions are preceded by check mark.

DSA Menu and Toolbar Explained

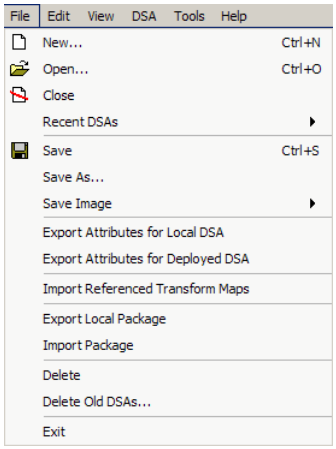
The DSA menus and toolbar are the main client workspace and the default when the Application Studio is opened. The following briefly describes the DSA toolbar buttons from left to right:



Tip: The tooltips appear when you rest your mouse pointer on a menu item, button, tab, icon, or similar content.

The following sections briefly describe each of the DSA menu commands and corresponding buttons:

File Menu



New…

Creates a new DSA for building reports about your enterprise data. These project files are stored in one of the following directories:

`C:\Documents and Settings\`*`Username`*`\Applications\DataLens\data\project`

or

`C:\Users\`*`Username`*`\AppData\DataLens\export`

Open…

Opens a DSA from a local drive version of the map; it does not check out a new copy from the Oracle DataLens Server.

Close

Closes the open DSA and saves it to disk.

Save

Saves the open DSA to your local disk. This does not perform a check-in or save the DSA to the Oracle DataLens Server.

Save As

Allows you to save the current DSA to a new name.

Save Image

Allows you to save an image of the design panel or the entire application dialog as a JPEG file.

Export Attributes for Local DSA

Creates a `.csv` file containing detailed attribute information for the Item Definition Transformations used in the current DSA. In other words, all Item Definitions known to all data lenses that are known to the DSA are exported. This export process is performed on the open version of the DSA as stored on the local machine.

Export Attributes for Deployed DSA

Creates a `.csv` file with detailed attribute information for the Item Definition Transformations used in the deployed version of the current map. In other words, all Item Definitions known to all data lenses known the DSA are exported.

Import Referenced Transformation Maps

When a new Transformation Map is defined in the current lens, this option tells Application Studio to check its export directory for a transform of the same name. If one is found, that transform is imported to the current map. This is a very handy technique for copying Transformation Maps from one DSA to another: open the source DSA, export the desired transform, then open the destination map and select Import Referenced Transformation Maps. For more information, see "Importing and Exporting Maps" on page 4-56.

Export Local Package

Exports the entire open DSA, including data lenses, into a single zipped file. This allows you to easily transport and share DSAs outside of the application.

Import Package

Imports an exported local package file.

Delete

Deletes the local copy of the current DSA. This has no effect on deployed or selected-in copies of the current DSA in the Oracle DataLens Server.

Delete Old DSAs…

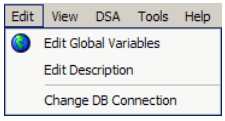
Deletes local copies of read-only DSAs. This is particularly useful in cleaning up completed DSAs at the end of a project.

Exit

Exits the Application Studio application; a prompt is given for unsaved changes.

Edit Menu

Edit Global Variables



Allows you to change variables globally in SQL statements in your DSA. For more information, see "Modifying Global Variables" on page 3-1.

Edit Description

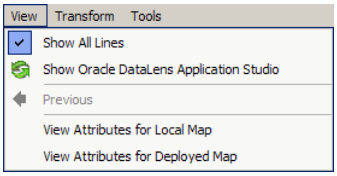
Allows you to edit the description for the open DSA. This is the description displayed on the **DSA** page in the Oracle DataLens Server.

Change DB Connection

Allows you to select a new database connection for the open DSA. All database connections in all DSA steps are automatically updated.

View Menu

View My Tasks



Allows you to view any tasks that are scheduled to run or have run. For more information, see "View My Tasks" on page 3-11.

Show All Lines

Works as a toggle and when selected, all of the connecting lines between inputs, transformations, and outputs

in the **Graphical DSA Builder** pane are active. When this option is not selected, only the connecting lines for the selected node are active and ghosting is activated if set in the application options. For more information, see "Modifying Application Studio Options" on page 3-2.

Show Transformation Map Builder

Works as a toggles when a Transformation Map is open so that you can switch the display between the main DSA map and the Transformation Map.

View Paths

Creates an XML report showing all possible paths through the current map that is displayed in your default browser.

View Attributes for Local DSA

Displays an alphabetized list of all attributes associated with all Item Definition Transformation nodes used in any of the steps in the current (local) DSA. This report includes only the Item Definitions for the open version of the DSA, as stored on the local machine; it does *not* include any selected-in or deployed versions of the same DSA.

The report can be saved in `.csv` format using the **Save** button, and then providing a file name and save location.

View Attributes for Deployed DSA

Displays a list of attributes associated with all Item Definition Transformation nodes used in any of the steps in the deployed version of the open DSA. This report includes the Item Definitions for the deployed version of the DSA, which can differ from the local version of the DSA.

The report can be saved in `.csv` format using the **Save** button, and then providing a file name and save location.

View Local Dependencies

Displays a list of all data lenses referred to by the local version of the *current* (local) DSA and shows the DSAs that use each of those lenses. The report shows the lens name, local revision number, deployed revision number, and whether the lens is locked.

View Production Dependencies

Displays a list of all data lenses referred to by the *deployed* version of the open DSA and shows the DSAs that use each of those lenses. The report shows the lens name, current (local) revision number, the development, QA, and the production Oracle DataLens Server revision numbers, and whether the lens is locked or not.

Revision number columns may indicate that the lens is not deployed.

View Native DB Connections

Displays the details for the current DSA database connection including the driver, test string, and usage information.

View Server Information

Displays the name and port information for the Oracle DataLens Server to which the current user is connected.

View User Roles

Displays all roles granted the current user.

View Check-In History

Displays a list of the DSAs that the current user has selected in including the comments regarding the check-in.

View My Check-Outs

Displays a list of the DSAs that the current user has checked out.

View All Check-Outs

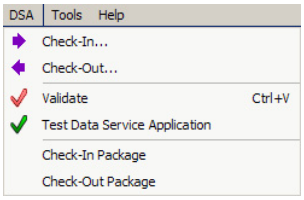
Displays a list of the DSAs that *all* users have selected out by user id.

View DB Connections Tree

Displays the database connections used by each step and map in the DSA in a graphical hierarchical structure. This includes pre-processing and post-processing steps.

DSA Menu

Check-In



Allows you to check-in a DSA file into your Oracle DataLens Server repository. Each time you check a DSA into the Oracle DataLens Server, the DSA revision number is incremented. For more information, see "DSA Actions" on page 3-4.

Check-Out

Allows you to check out a local copy of a DSA from the current Oracle DataLens Server for review, maintenance, or as a new DSA. For more information, see "DSA Actions" on page 3-4.

Validate

Validates the entire DSA and informs you of any issues found or that no problems were found. For more information, see "DSA Actions" on page 3-4.

Test Data Service Application

Allows you to test the DSA with a single line input. For more information, see "DSA Actions" on page 3-4.

Check-In Package

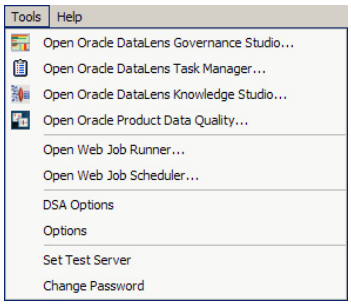
Allows you to check-in both the current (local) DSA and all lenses associated with it. For more information, see "DSA Actions" on page 3-4.

Check-Out Package

Allows you to checkout both the desired DSA and all lenses associated with that map all at once. For more information, see "DSA Actions" on page 3-4.

Tools Menu

Open Oracle DataLens Task Manager…



Starts the Oracle Product Data Quality Task Manager. For more information, see *Oracle Product Data Quality Task Manager Reference Guide*.

Open Oracle Governance Studio…

Starts the Oracle Product Data Quality Governance Studio. For more information, see *Oracle Product Data Quality Governance Studio Reference Guide*.

Open Oracle DataLens Knowledge Studio…

Starts the Oracle Product Data Quality Knowledge Studio. For more information, see *Oracle Product Data Quality Knowledge Studio Reference Guide*.

Open Oracle Product Data Quality…

Starts the Oracle Product Data Quality Launch Pad.

Open Web Job Runner…

Launches the **Job Runner** function of the Oracle DataLens Server in a browser. This submits a single job to run immediately. For more information about how to use the Job Runner, see *Oracle Product Data Quality Oracle DataLens Server Administration Guide.*

Open Web Job Scheduler…

Launches the **Job Scheduler** function of the Oracle DataLens Server in a browser. This schedules a job for deferred execution unlike the immediate execution of the **Job Runner**. For more information about how to use the Job Runner, see *Oracle Product Data Quality Oracle DataLens Server Administration Guide.*

DSA Options

Allows you to set the option to make the DSA an ultra-high priority DSA for super fast, single-line requests to the Oracle DataLens Server.

Options

Allows you to set options for global use in the DSA workspace. For more information, see "Decision Map Options" on page 5-9.

Set Test Server

Identifies the Oracle DataLens Server to be used for DSA testing. For more information, see "Setting the Test Server" on page 3-10.

Change Password

Allows you to change your password by starting the **Change Password** dialog box so that you can enter your old password followed by the new password.

Help Menu

Product Guide

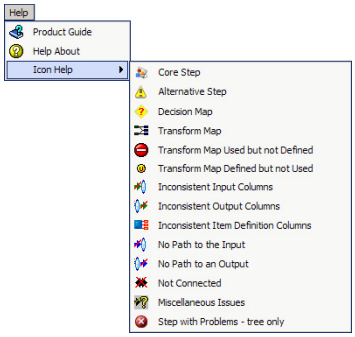
Opens a list of Oracle Product Data Quality documents for your selection in a browser.

Help About

Provides information regarding the product including the version number and a link to view third party product licenses.

Icon Help

Explains each of the icons that can appear in the Application Studio.



Keyboard Shortcuts

The following table contains keyboard shortcuts that can help make the Application Studio easier to use:

Function	Shortcut Key
New DSA	Ctrl-N
Open DSA	Ctrl-O
Save	Ctrl-S
Validate	Ctrl-V

Task Panes

The interactive task panes allow you to perform actions specific to the type of pane and these actions are described throughout this reference.

DSA Component Tree Pane

This task pane contains all of the components necessary to building a DSA. These components are categorized into the hierarchical tree structure with the main folder indicting the name of the current DSA or 'No Data Service Application' to indicate that there is no DSA open. The sub-folders of any DSA are as follows:

Process Steps
Contains the main components to transform data.

Data Input
Contains the nodes used to indicate one of the three methods that data is communicated to the DSA: text, a database, or an XML file.

Data Output
Contains the nodes to form how the data is output from the Application Studio.

Output Adapters
Contains the nodes to output XML information to correspond with XML input data.

Pre-Post Processing Steps
Contains the steps that can be used before and after the data is processed.

Defined Processing Steps
Lists all of the processing steps defined for the current DSA. This folder is helpful in locating processing steps in large, complex DSAs that do not easily fit into the **Graphical DSA Builder** pane.

Unused Transformation Maps
Lists any transformation maps that you have created in the current DSA or imported, and are not assigned to a processing step. This folder acts as a container for maps that you are developing and as an intermediary for you to be able to import maps from other DSAs so that you do not have to recreate a map that you may use repetitively.

Graphical DSA Builder Pane

The **Graphical DSA Builder** pane is a free-form building pane into which you can drag and drop steps from the DSA Component Tree pane. It is from these steps that transformation and decision maps are built thereby transforming your data.

Context-Sensitive Menus

There are various context-sensitive (shortcut) menus that appear in the Application Studio panes when you right-click on data within a task pane. The contents of these menus are described throughout this reference though may contain the following standard options:

There are various context-sensitive (shortcut) menus that appear in the Application Studio panes when you right-click on data within a pane. The contents of these menus are described throughout this reference though may contain the following standard options:

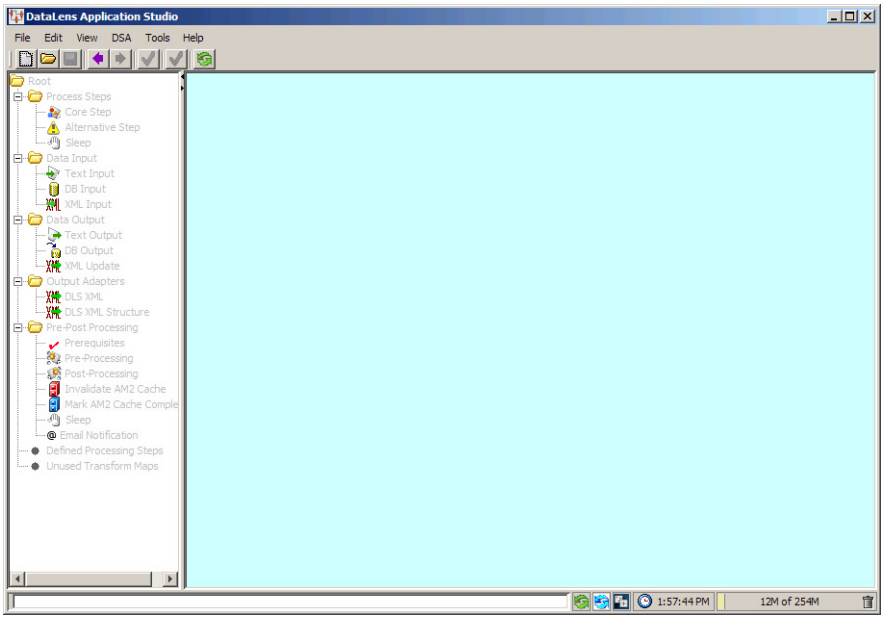
Expand Node
Expands all sub-nodes (steps, database queries, etc.) of the selected node in a hierarchical manner.

Show All Lines / Save Image
These menu functions operate as previously described.

The small up and down arrows between the panes on the left-hand side, allow you to resize the panes. In addition, you can fully expand either pane to see more data by clicking on an arrow, which makes the pane inactive. To redisplay the inactive pane, click the opposite arrow and the pane reappears.

Starting the Application Studio

If this is the first time you have started the Application Studio, the client workspace appears blank as in the following figure; otherwise, the results from the last job run are displayed:



The status field at the bottom of the Application Studio client workspace provides information about any DSAs you load in the white field, and the date and time, and memory usage are displayed in the grey fields. The status field is blank until you have created your first Application Studio project, at which time the status of your project is displayed.

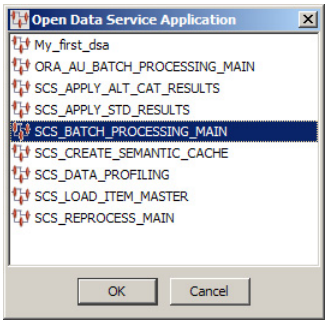
2

DSA Basics

This chapter explains the basic concepts of creating, editing, and using DSAs.

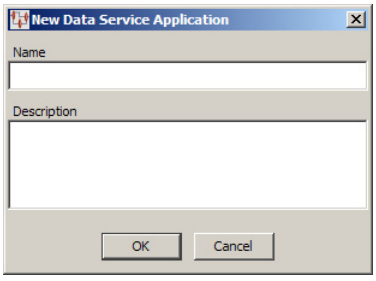
Creating or Opening a DSA

If this is the first time you have started the Governance Studio, the startup window appears blank as in the following figure; otherwise, the results from the last job run are displayed for your selection as in the following figure:



The following steps illustrate how to create a new DSA:

1. From the **File** menu, click **New…** create your new DSA.



2. Enter the unique name for this DSA. In the Application Studio, entering a space results in an underscore.
3. Enter a description for this DSA; a description must be specified.
4. Click **OK**. The Application Studio creates your new DSA.

   Your new DSA is located in:

```
\Documents%and%Settings\%USERNAME%\Application%Data\DataLens\
data\workflow
```

Your DSA opens and is now ready for use.

DSA Building Components

There are three types of DSA building components and the use of each is described in this section:

- Processing Steps Folder
- Data Input Folder
- Data Output Folder

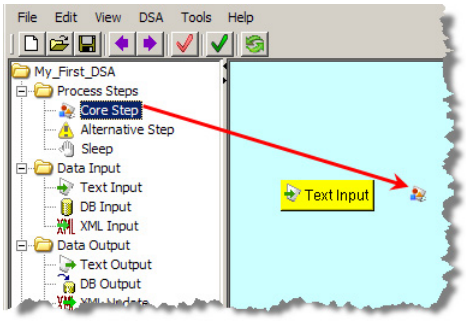
Processing Steps Folder

Processing your input data can take on a myriad of different steps. You can create a simple DSA that only looks for a minimum of data matching or one that vigorously reviews the data for matches, duplication, and enriches it. This section describes how the processing nodes can be used to process data.

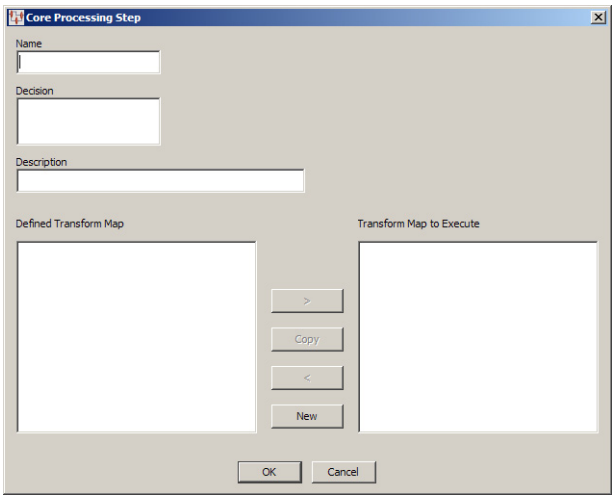
Core Processing Steps

The core processing steps are the foundation of transforming data. These steps are used to define how the input data will be manipulated in preparation for output formatting. You can add as many core processing steps as needed to transform your data effectively.

The addition of core steps is a simple by selecting a **Core Step** node from the **Map Component Tree** pane and dropping it into the **Graphical DSA Builder** pane to the right of an input node.



The **Core Processing Step** dialog is displayed so that you can begin to build the step.



Use this dialog as follows:

1. Enter a name for the step. Spaces are automatically converted to underscores.
2. Enter a brief description of the function this step will perform in the Decision field.

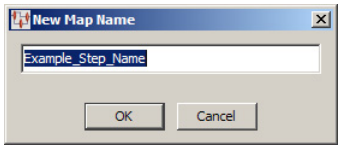
   Tip: Use the **Enter** key to create a line feed in this field; otherwise, a left-right scroll bar appears.

3. Enter a more informative description of the step to make step differentiation easier in future. This is very useful in large DSAs that contain numerous similar steps.

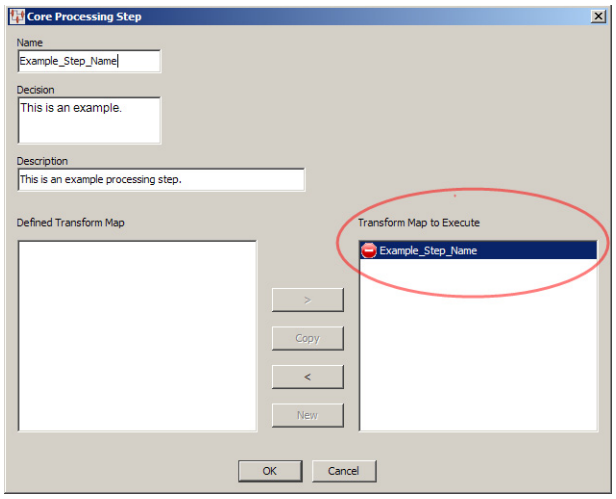
   Note: There is no character limit for this field though when the DSA is loaded this description is truncated to 255 characters. While the entire description is retained for the step, the truncated description is used as a reference to the step.

4. If you have already defined one or more Transformation Maps, they are listed in the **Defined Transformation Map** list. You can choose to select one of these maps to be executed by this step or you can click the **New** button to create a new map.

   Creating a new map automatically populates the resulting dialog with the name of the Transformation Map as the same name as the core step.



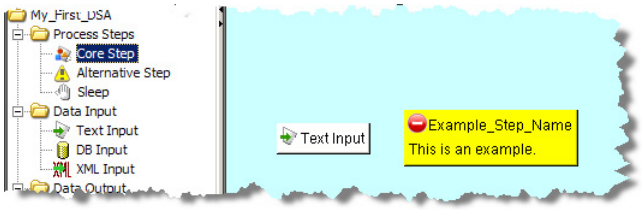
   You can accept this default or enter the name you want to use, and then click **OK**.



The **Core Processing Step** dialog is updated and the new map is listed in the **Transformation Map to Execute** list.

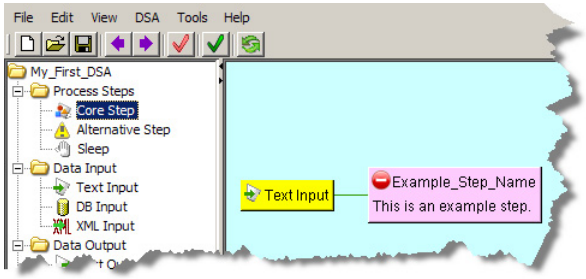
5. Click **OK** to accept the step you have defined.

   This adds the new core processing step to the **Graphical DSA Builder** pane.



The step will have a warning icon associated with it to inform you that a Transformation Map has not yet been created for the step.

6. Add the input data to the core step by dragging the Input node and dropping it onto the core step.



Alternative Step Nodes

Alternative steps receive items identified as exceptions by a core step in the process. These steps are added the same way as core steps as previously described, and are then connected to core steps by dragging the alternate step onto the core step.

The Application Studio adds a route line between the two nodes. It is important that connect alternate steps to core steps prior to defining the map for the alternate step.

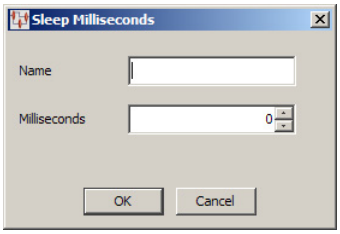
Note: If you do not create an alternative step for exception data handling, the exceptions are discarded and so are not processed or output in any form.

Sleep Step Nodes

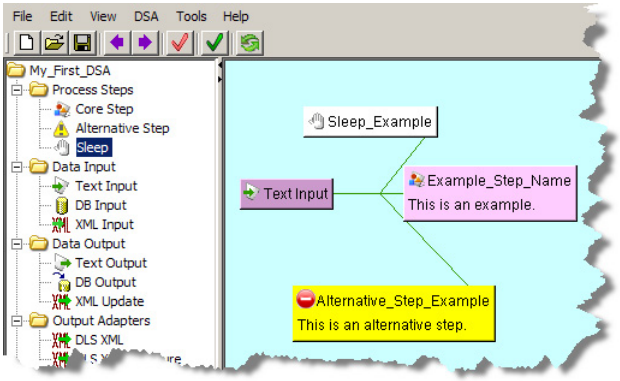
You can use a sleep (wait) step in your processing schema to tell the DSA to halt for a predetermined amount of time. The addition of these steps can help you in debugging your DSA, allow a particular step that operates slowly the time to complete, or to aid in a processing race condition.

Sleep steps are added anywhere in the processing schema to any process step as follows:

1. Drag a **Sleep** node from the **Map Component Tree** pane and drop it into the **Graphical DSA Builder** pane above and to the right of the step that you want to wait.



2. Enter a name for this sleep step. Since there is no description for the step, ensure that the name will differentiate it from any other sleep steps that may add.
3. Use the arrows to select or enter the sleep time. The sleep time is in milliseconds so be sure that you set the correct amount of wait time needed. For example, one minute is 60000 milliseconds.
4. Only the name of the step appears on it. However, the tooltip that appears when you hover over a sleep step is comprised of the name and number of milliseconds for which the step is set.
5. Click **OK**.
6. Drag and drop the sleep step onto the processing step you want to wait, which connects the two steps.

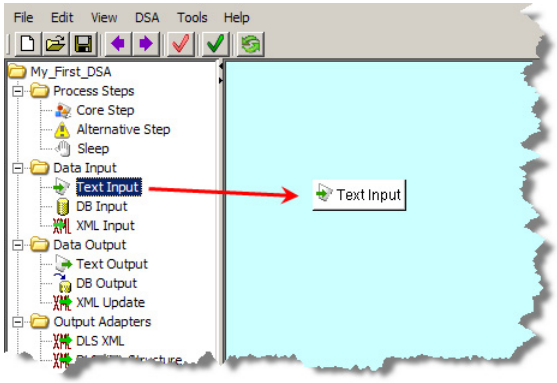


Data Input Folder

The first step in any DSA is to identify how data is the input type. There are three different types of data input containers used in the Application Studio: text, database, or XML. Only one type of container can be used in each DSA though the Transformation and Decision Maps that you add to the DSA can use all three types of data input.

All types of data input nodes are added to a DSA in the same manner as in the following example:

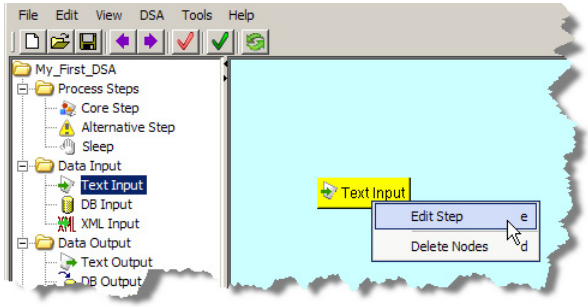
From the **DSA Component Tree** pane, in the **Data Input** folder, drag an Input node (Text, database (DB), or XML) into the **Graphical DSA Builder** pane as in the following example:



The Input node is connected to processing steps by dragging and dropping it onto the appropriate step. One Input node can be connected to numerous processing steps because it is the data source.

Editing or Deleting an Input Node

You can rename the text input node or change the description, by right-clicking the input node, and selecting **Edit Step**.



The **Edit Step** dialog is displayed. You can modify the text input name and the description for this input, by changing the text in the fields of the dialog, and then clicking **OK**.

Note: There is no character limit for the **Description** field though when the DSA is loaded this description is truncated to 255 characters. While entire description is retained for the step, the truncated description is used as a reference to the step.

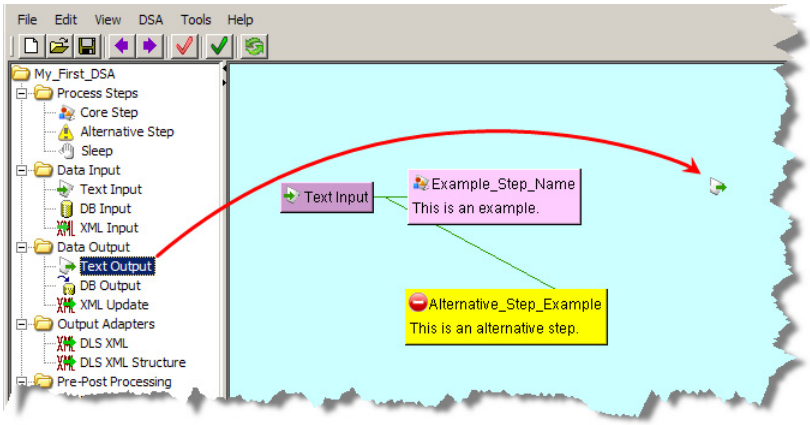
Input nodes can be deleted in much the same manner by right-clicking a node and then selecting **Delete Nodes**. Use this option with care as there is no confirmation prompt, and the node is immediately removed from the DSA. If the Input node has been joined to other nodes in the DSA, all ties are severed when the node is deleted.

Data Output Folder

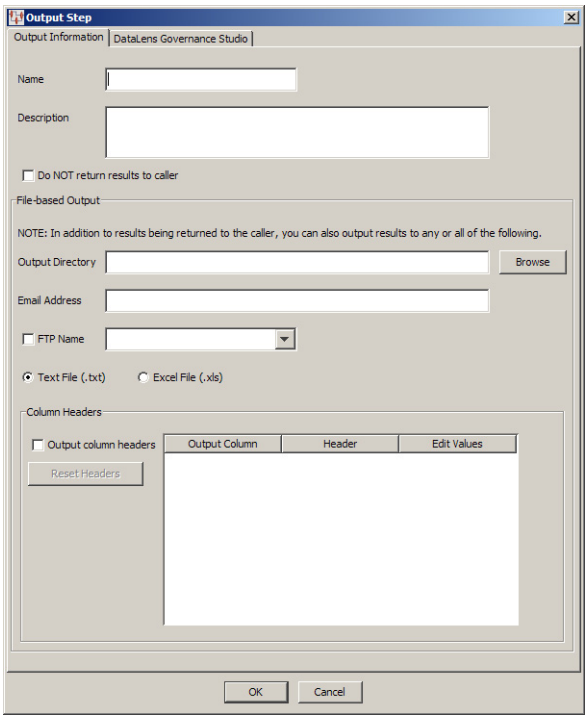
The final step in DSA creation is to form the data as output for use in other parts of Oracle Product Data Quality. You should create an Output node for each Processing Step, including Alternative (exception) Steps.

Text Output Nodes

From the **DSA Component Tree** pane, in the **Data Output** folder, drag an Output node (Text, DB, or XML) into the **Graphical DSA Builder** pane to the left of a processing step as in the following example:



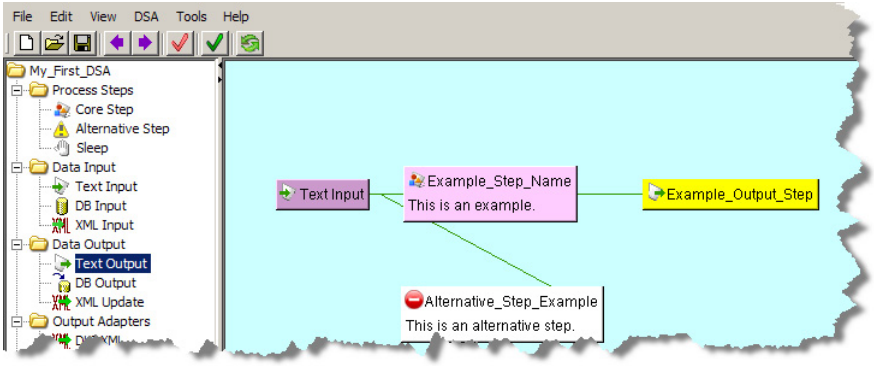
The **Output Step** dialog is displayed so that you can begin to build the step.



Initially, it is easiest to enter a name and description for the Output step, and click **OK**.

Note: There is no character limit for the **Description** field though when the DSA is loaded this description is truncated to 255 characters. While entire description is retained for the step, the truncated description is used as a reference to the step.

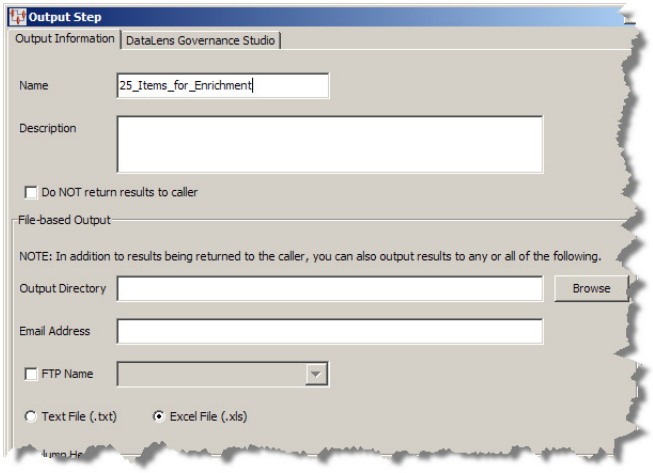
This allows you to connect a Processing Step to this Output Step by dragging the appropriate Processing Step onto the Output Step.



Connecting the end-to-end processing (input, processing, and output) so that the **Column Headers** section of the **Output Step** dialog is populated and ready for you to use by editing the output step as described in the following section.

Editing a Text Output Step

You can edit an Output Step by right-clicking on it and selecting **Edit Steps**. The **Output Step** dialog is displayed so that you can edit the step.



Output Information Tab Modify the **Output Information** tab as follows:

Enter a description of the output step in the **Description** field.

> **Note:** There is no character limit for the **Description** field though when the DSA is loaded this description is truncated to 255 characters. While entire description is retained for the step, the truncated description is used as a reference to the step.

Select the **Do NOT return results to caller** check box *only* if you do not want the data output of this step to be available to the Governance Studio.

> **Caution:** Returning the results of the step to the caller results in a job whose data you must retrieve or delete. Failure to do so can result in an Oracle DataLens Server crash once the server memory is exhausted. The memory can be exhausted from many small jobs that are not retrieved because each DSA job that is below the DSA maximum memory or chunk size set for the server is held in memory until it is retrieved or the server is restarted. For information about setting these values, see *Oracle Product Data Quality Oracle DataLens Server Administration Guide.*
>
> To avoid this situation, use the **Job Status** feature of the calling application (Governance Studio or Services for Excel) to delete or retrieve the persistent data results immediately after the job is run, particularly for those jobs that return large amounts of data.

Use of this check box deactivates the **DataLens Governance Studio** tab and the data output is *only* stored provided in the methods you select in the **File-based Output** section. Notice that the **NOTE:** changes to indicate how the data will be output.

File-based Output Section Use the any or all of the controls in the **File-based Output** section so that the resulting output data from this step is conveyed the way you want:

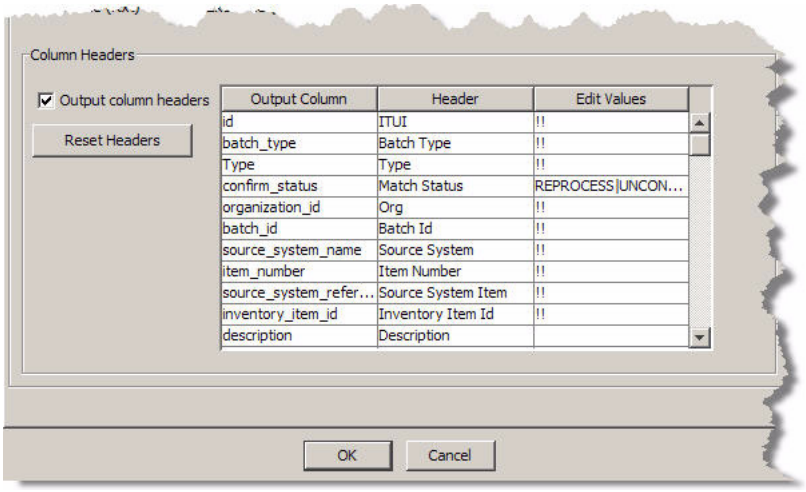
Output Directory
Enter or browse to a directory to store output files.

Email Address
Enter one or more email addresses to whom you want the output results sent.

FTP Name
If you have FTP Connections established in the Oracle DataLens Server, you can select one of these connections to receive output results.

Text/Excel File
Select one of these file types (`.txt` or `.xls`) in which to store the output results.

Column Headers Section Use the **Column Headers** section as follows:



Output column headers
If you want the output results to contain headers to identify each column of data, select this check box. This is the default.

Headers Help
Select to view a brief explanation of how to use the table in the **Column Header** section.

Output Column / Header
When connected to an Input node, the **Output Column** and **Header** columns are populated with the available output selections.

You can change the names of the column headers that appear in the **Header** column so that they are more descriptive for the user or use the defaults. As in the preceding example, 'batch_type' and 'orig_desc' have been changed to 'Batch Type' and 'Description' respectively.

Edit Value
There are several ways you can configure how output columns appear in the Governance Studio as follows:

- **Hidden**

  Indicate that you do not want the column to appear in the output results by entering a hash mark, #. The output data is included in the source data though it does not appear in the user.

- **Read Only**

  A null value in this column means that the column is used for review only so it is not editable.

- **Editable**

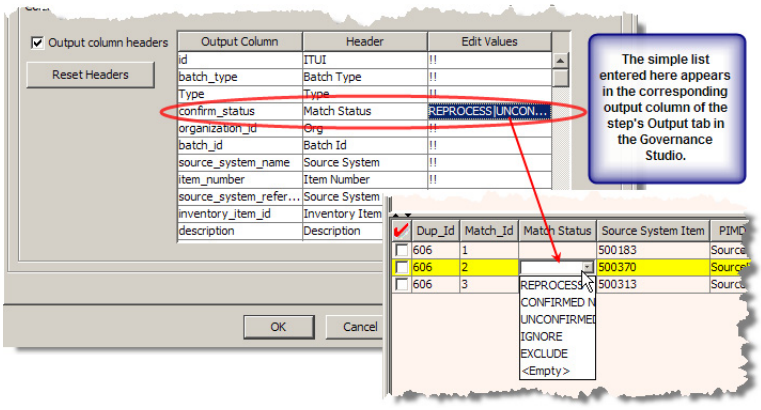
  Indicate that you want the column to appear in the output results and allow it to receive input by entering two exclamation marks (!!). This sets the corresponding column to be edited. Additionally, you can include help text adjacent to the !! to appear to the user.

- **List of Values**

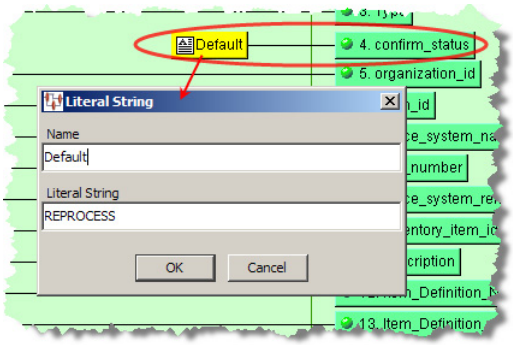
  You can add a simple list of values that are translated into a list menu to comprise the available selections for users. This list of values can contain any combination of text, space, number, and valid UTF-8 characters. The list items are delineated by a vertical bar and appear in the list menu in the exact order entered. For example, a list of matching process status may be:

  `REPROCESS|UNCONFIRMED|CONFIRMED NEW|IGNORE|EXCLUDE`

  For example, you could use this list in a Match Status output column so that it would force the user to set the data record consistently to improve consistency.



  If you want to set a default value for a list menu, you must precede the output node with a default string value using a Literal String widget (as described in "Strings Widgets" on page 4-14.



- **List from SQL Statement**

  Indicate that you want the column to appear in the output results and allow it to populate a list of options for the user to select from using a database by entering two caret symbols (^^) preceding a valid SQL statement. This allows you to automatically update the list by changing the database from which you are retrieving data. For example, to populate a list of manufactures that users can

chose from based on the frequency they are used, you could use the following statement:

```
^^select value from editvalues where id = 'mfg' order by frequency desc
```

DataLens Governance Studio Tab

Click the **DataLens Governance Studio** tab to view the Governance Studio output configuration options.

Use the **DataLens Governance Studio** tab as follows:

Output Type Section As shown in previous figure, you may choose one of the following output types to determine how data is displayed in the Governance Studio:

Not shown (will not appear in application)
Results will not appear in the Governance Studio Output tabs. Use this output type for results that do not conform to Governance Studio requirements, such as database output steps.

Review, Approve and Route
Records the data that exactly meet match criteria are displayed. Use this type for displaying the results of a DSA template that runs a match process.

Review, Approve and Route with AutoSuggest
Alternatives to records meeting match criteria are displayed below a split-screen in the output spreadsheet. You must select one record from the alternatives; the records are available for downstream processing. Use this type for displaying the results of a DSA template that runs a match process.

Match Results (Single Selection Review, Approve, and Route)
Possible records meeting match or duplicate criteria are displayed below a split-screen in the output spreadsheet. Only a single record from the list of matched items can be selected; the record is available for downstream processing. Use this type for displaying the results of a DSA template that runs a match process.

Match Results (Multiple Selection Review, Approve, and Route)
Records meeting match or duplicate criteria are displayed below a split-screen in the output spreadsheet. Multiple records from the list of matched items can be selected; the records are available for downstream processing. Use this type for displaying the results of a DSA template that runs a match process.

Summary Data for Graphing
Summary information is displayed in the graph; no selection of records is available with this option. The bars in the graph operate like buttons and when clicked jump to the Output tab for the data represented in the selected bar.

Summary Data for Graphing
An individual tab is not created; the data appears in the **Graph Summary** tab.

Other (Display Only)
Result information is displayed in a spreadsheet form that does not allow user interaction.

Show Message Dialog
All informational messages, including errors and warnings, are captured and displayed to the user.

Graph and Data Summaries Section The options in this section are activated by your selections in the **Output Type** section and are used as follows:

Don't Show in Graph Summary
This option is active when the **Summary Data for Graphing** option is selected. The default is that the graph summary for the output step is included on the **Graph Summary** tab and selecting this option turns off that behavior so that the graph does not appear.

Don't Show In Data Summary
This option is active when either of the review or match options, or the **Other** option is selected. The default is that the data summary for the output step is included on the **Data Summary** tab and selecting this option turns off that behavior so that the data does not appear as a bar in the graph.

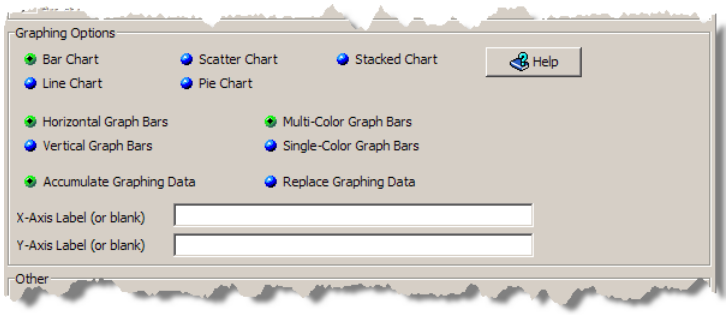
Trend Reporting Section This section is active when any of the **Output Type** options are selected with the exception of the summary data options. The default is that there is no trend analysis data collected. Selecting this option turns off that behavior so that quality data is collected for use in additional DSA steps that chart the data for display.

Graphing Options Section This section of the tab is active only when one of the summary data **Output Type** options is selected.

There are four graphing options available for the output types that include graphs. You can mix and match between two sets of options for the graphs and create a label for the X and Y axes as follows:

- Bar direction
  - Horizontal
  - Vertical bars
- Bar color
  - Multi-color bars
  - Single-color bars

As shown in following figure, you may choose from the following graphing options to configure how graphs appear in the Governance Studio:



The data values across the defined output steps can be compared and displayed in the following types of charts:

Bar Chart
A 3-D, colorized visual effect that shows the data in a bar format. The orientation of the bars is determined by the selection of **Horizontal Graph Bars** or **Vertical Graph Bars** options. The use of color in this type of chart is defined using the **Multi-Color Graph Bars** (a different color for each Output tab) or **Single-Color Graph Bars** (one color for all Output tabs.)

Line Chart
A line connecting the various output data points in a simplified manner.

Scatter Chart
A series of connected markers that show the data relationship.

Pie Chart
A 3-D, colorized representation where each tab is depicted by its percentage of contribution to the total.

Stacked Chart
A 3-D, colorized representation where each tab is depicted by its percentage of contribution to the total.

Accumulate Graphing Data
The original data is retained when new data is added and all are rendered in the graphs.

Replace Graphing Data
All data is replaced by new data prior to rendering in a graph. The DSA is responsible for recalculating the results each time to ensure that the graph has the correct values.

X and Y-Axis Labels
You can add meaningful labels to indicate the data is graphed on the X and Y axes or you can leave it blank to use the default labeling.

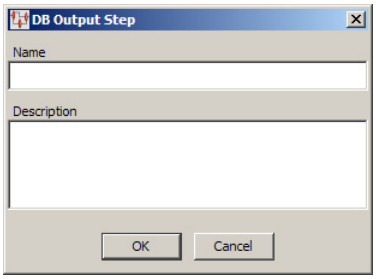
Help Button
Use the **Help** button to review information about the various graphing options.

Other Section This section of the tab is active only when one of the review or match **Output Type** options are selected.

This advanced functionality is described in on Using Secondary DSAs on page 199.

DB Output Nodes

From the **DSA Component Tree** pane, in the **Data Output** folder, drag a **DB Output** node into the **Graphical DSA Builder** pane to the left of a processing step.



Enter a name and description for this step, and then click **OK**. Connect a Processing Step to this DB Output Step by dragging the appropriate Processing Step onto it.

> **Note:** There is no character limit for the **Description** field though when the DSA is loaded this description is truncated to 255 characters. While entire description is retained for the step, the truncated description is used as a reference to the step.

XML Update Nodes

From the **DSA Component Tree** pane, in the **Data Output** folder, drag the **XML Update** node into the **Graphical DSA Builder** pane to the left of a processing step. You connect the node by dragging a Processing Step and dropping it onto the new XML Update node. The data resulting from the connected process step is reproduced in an XML output file.

You can change the default name and description of an XML Update node by double-clicking on it and entering the new information.

Output Adapters Folder

Use these nodes to output XML type input data.

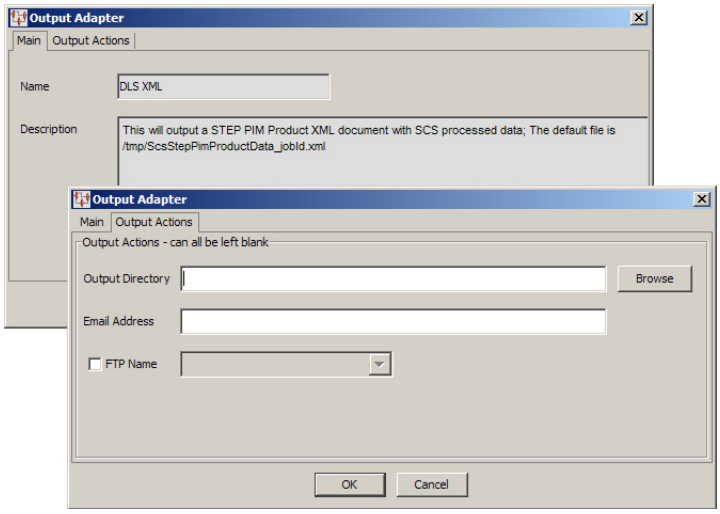
DSL XML Nodes

You can use the DSL XML node to output processed data into an XML file, which is by default stored in `/tmp/ScsStepPimProductData_jobid.xml`. Because only one of

these nodes can exist in a DSA, the name and description are defaulted and cannot be changed.

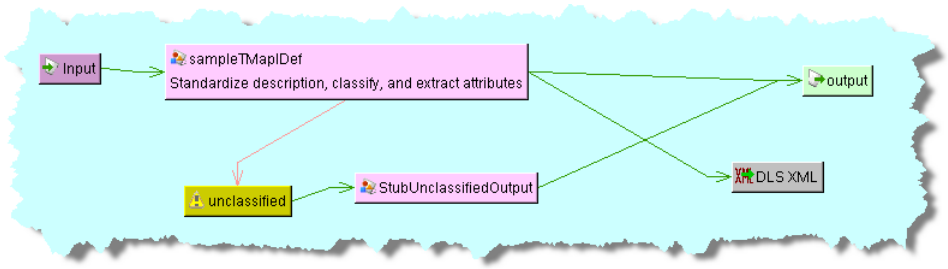
From the **DSA Component Tree** pane, in the **Data Output** folder, drag the **DSLXML** node into the **Graphical DSA Builder** pane to the left of a processing step. You connect the node by dragging a Processing Step and dropping it onto the new **DSL XML** node.

The output actions for the node can be changed by double-clicking on it.



You can enter or locate the filename and path of where the data will be stored, enter an email address so that the file is emailed, or select an FTP site that is configured in your Oracle DataLens Server to store the file.

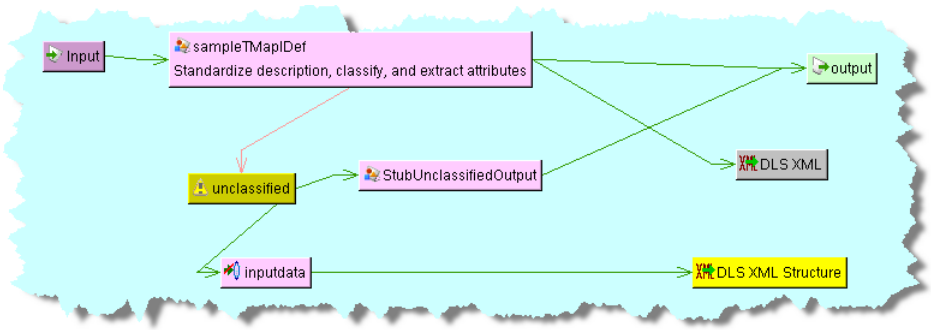
A simple DSA that processes input data to standardize the description, classify it by UNSPSC category, and extract all attributes then output the data to an XML file might look like the following:



DSL XML Structure Nodes

The DSL XML Structure node is used to output a report of the structure of the processed data, *not* the data itself, into an XML file, which is by default stored in `/tmp/ScsStepPimClassificationStructure_jobid.xml`. Because only one of these nodes can exist in a DSA, the name and description are defaulted and cannot be changed.

This node is used identically to the **DLS XML** node described in the previous section. Adding this node to the example DSA, in the previous section, to capture and report the XML data structure created in the **DLS XML** output node might look like the following:



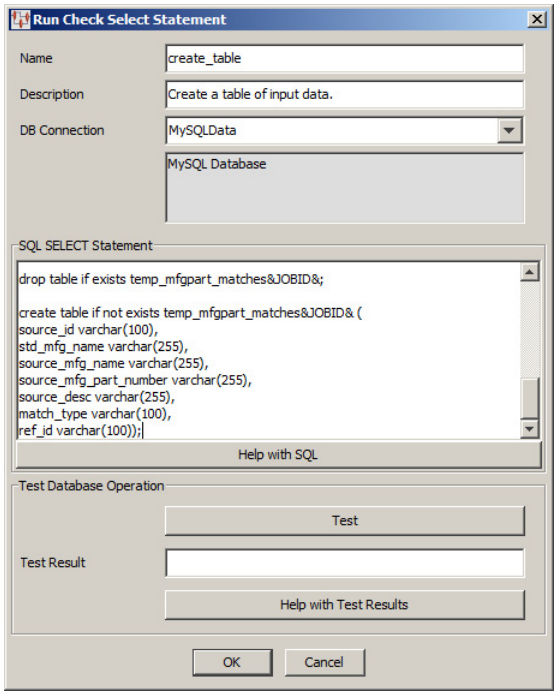
Pre-Post Processing Folder

You can add instructions that your DSA executes before or after all of the processing steps are executed. It is immaterial where these nodes are placed in your map as they are processed as indicated though the convention is in the top line of the DSA.

Prerequisites Nodes

To ensure that any data manipulation that is required before processing, such as creating a temporary attribute table, **Prerequisites** nodes are used.

From the **DSA Component Tree** pane, in the **Pre-Post Processing** folder, drag a **Prerequisites** node into the **Graphical DSA Builder** pane to the top left of the map.



The **Run Check Select Statements** dialog box appears; use it as follows:

Name and Description

Enter a name and a brief description for the node.

Note: There is no character limit for the **Description** field though when the DSA is loaded this description is truncated to 255 characters. While entire description is retained for the step, the truncated description is used as a reference to the step.

DB Connection

You must select the type of database connection that you want to use. The list of database connections is populated based on those that you are configured in the Oracle DataLens Server. If the type of database connection is not listed, you must configure it in the Oracle DataLens Server so that it is available for selection when creating Transformation Maps.

SQL SELECT Statements

Use this section to construct your database query with standard SQL statements and syntax. For example, to create a temporary table of data you could use the following statements:

```
create table if not exists temp_source&JOBID& (
interface_table_unique_id varchar(128),
batch_type varchar(128),
organization_id varchar(128),
source_system_name varchar(128),
batch_id varchar(128),
manufacturer_name varchar(255),
mfg_part_num varchar(255),
source_system_reference varchar(255),
item_number varchar(512),
inventory_item_id varchar(128),
source_system_reference_desc varchar(240),
description varchar(240),
attribute_concatenated varchar(512));
```

These SQL statements must be compatible with your database. Optionally, you can use a question mark (?) in the `select` clause. At run time, the question mark character is replaced with the transformation input data. This allows you to create a database transformation that varies with the content of the record being processed. Additionally, when database transforms are used to aggregate data fields from several different data sources, common access key information can be used across all data sources.

Help with SQL

Select this button to view brief SQL query explanations and examples.

Test Database Operation

These fields are provided to help you test the setup of the data source connection. You can test the data source definition by clicking the **Test** button and reviewing the query results displayed in the **Test Result** field.

When the query returns more than one record, the database transformation uses only the first record returned.

If there is an error in the connection definition, an information error message appears. Click the **Help with Test Results** button for help in interpreting error messages.

Note: The following message indicates that the database connection tested good and is not an error message:

```
Please check your database now
```

If an error message is received and there are no results in the **Test Result** field, the connection is good and the query is valid, but no matching results were returned.

You can update **Prerequisites** nodes by double-clicking on them and modifying either the Description or the SQL statement in the **Pre-Processor Update Statements** dialog box that appears.

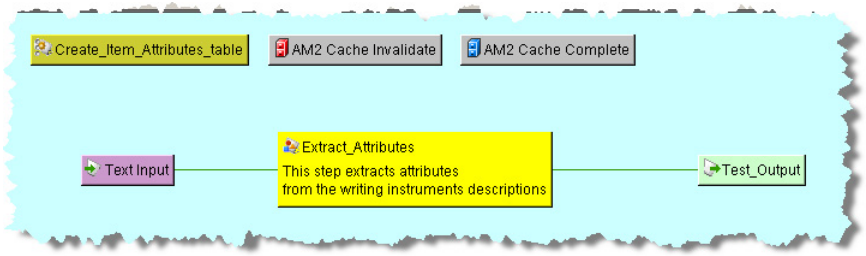
Pre/Post-Processing Nodes

These processing nodes are used to process SQL statements either before or after the DSA processes the input data. They are constructed as the Prerequisites nodes previously described. In addition, the **Post-Processing** node offers the option to for the run of this step even if the DSA job fails or is cancelled.

Invalidate AM2 Cache and Mark AM2 Cache Complete Nodes

These nodes are used in a DSA when the processing steps need to indicate that stale data exists that must be refreshed.

To add either node, from the **DSA Component Tree** pane, in the **Pre-Post Processing** folder, drag either node into the **Graphical DSA Builder** pane to the top left of the map. Because the DSA processes left-to-right, you should ensure that the **AM2 Cache Complete** node is to the right of the **AM2 Cache Invalidate** node as in the following example:



Sleep Nodes

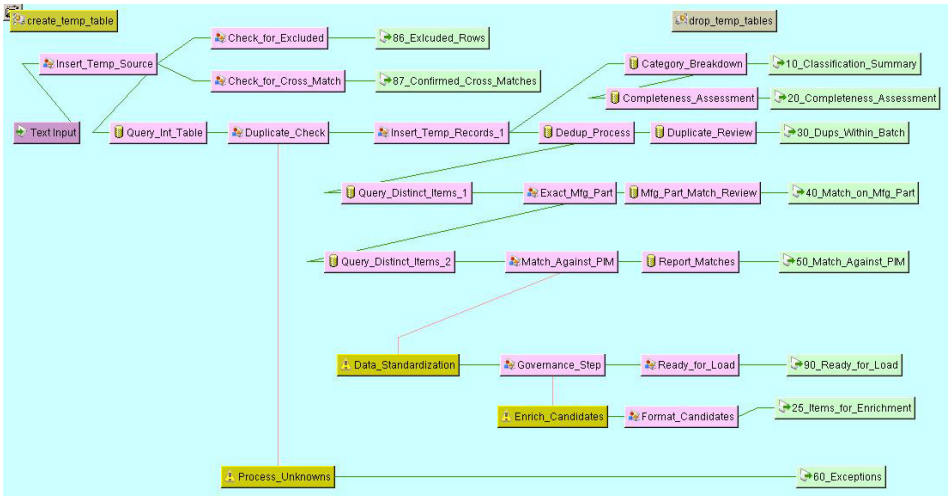
You can use a sleep (wait) step with your pre and post-processing steps to tell the DSA to halt for a predetermined amount of time. The sleep nodes are constructed the same way as Sleep processing nodes described in Sleep Step with the exception of how they are connected to other steps. With pre and post-processing sleep nodes, you drag and drop the sleep step onto one of the other nodes found in the same folder, which connects the two steps.

Putting It All Together

Assembling the various DSA building components into an effective DSA to process your data requires forethought in design and a clear delineation of the ultimate goal. You can build several small, discrete DSAs that accomplish one or two specific processing tasks, and then incorporate them into a large DSA to affect an end-to-end data processing schema.

Example DSA

The following is an example of a complex DSA that performs duplicate location, data standardization, location of missing attributes, and data clean-up:



This image was produced using the **Show Image** option as described in "File Menu" on page 1-6.

> **Tip:** The output steps in your DSA correspond to the titles of the output tabs in your Governance Studio project and appear in alphabetized order by default. The output steps in the example DSA are preceded by numbers, which when alphabetized for the output tabs to appear in this exact order. The numbers are ignored and do not appear on the tab labels. Using a numbering scheme similar to the example is an easy way to control the appearance of your output tabs in the Governance Studio.

3

DSA Management

This chapter explains how to manage your DSAs.

Modifying Global Variables

You can quickly change variables that appear in your SQL statements throughout your DSA globally. This is very useful to ensure that variables are modified in all places to avoid errors, improves consistency, and avoids opening each SQL statement to modify it manually. The following two types of global variables can be changed:

System Defined

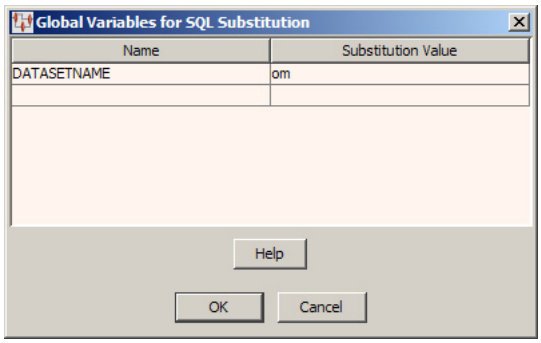
There are two System Defined Variables, `&JOBID&` and `&USERID&`. If a SQL statement contains one of these two strings, the entire string, including the ampersands (`&`), is replaced with the ID of the current job, or the ID of the user who created the job.

User Defined

Defined by specifying a name and a value in the table. The name must consist of letters and numbers *only*; all other characters are invalid.

Use the following steps to change global variables:

1. From the **Edit** menu, select **Edit Global Variables**.



> **Note:** All values entered are case insensitive.

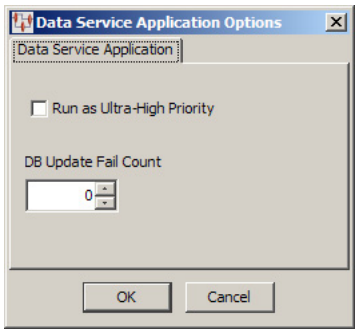
2. Enter the names of the variables you want replaced in the **Name** field.
3. Enter the substitution values for each variable in the **Substitution Value** field.
4. Click **OK**.

All values are globally replaced in all SQL statements that contain the named values in your DSA.

Modifying DSA Options

The Application Studio DSA options that you can configure are restricting the use of DSAs to specific server types, running them with ultra-high priority, and the allowable number of database errors per job.

1. From the **Tools** menu, select **DSA Options**.



2. Select the **Run Ultra-High Priority DSA** check box to run all DSA jobs with the highest priority and click **OK**. You should use this option for DSAs that are designed to deliver super fast, single-line requests to the Oracle DataLens Server.

   Note: No logging takes place so nothing is displayed in the Job Status page and the Job Id number is negative. Jobs that start and finish with a negative Job Id are shown in the `dataserver.log` file.

3. Select the number of database errors that you want to allow before a job is terminated by the Oracle DataLens Server with the **DB Update Fail Count** control. When set to zero (the default), jobs continue to run regardless of a database failure and do not terminate.

4. Select one or more check boxes to indicate the Oracle DataLens Servers on which this DSA will be restricted for use and viewing by users.

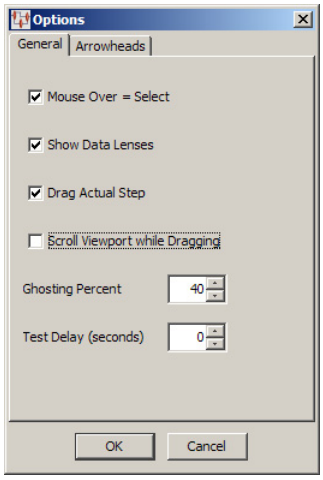
   You can restrict users from using and viewing DSAs on your various types of servers by using these check boxes in conjunction with editing a users roles for the use of the Application Studio on **Role Administration** page in the Oracle DataLens Servers Administration Web pages. For more information about role administration, see *Oracle Product Data Quality Oracle DataLens Server Administration Guide*.

5. Click **OK** to use the changed options immediately.

Modifying Application Studio Options

To set the global Application Studio options to be used throughout the application:

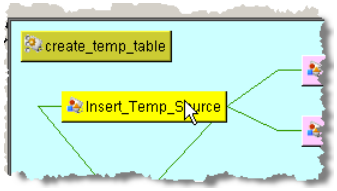
1. From the **Tools** menu, select **Options**.



2. Select the available controls as appropriate from the following:

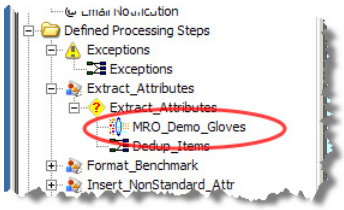
   - **Mouse Over = Select Check Box**

     Allows you to select nodes in the DSA simply by hovering over a node with the mouse pointer. This option is set by default.



   - **Show Data Lenses Check Box**

     Allows you to toggle the display of data lenses in use in the **DSA Component Tree** pane. This option is set by default.



   - **Drag Actual Step Check Box**

     Enables a visual effect that causes a step's connection lines to follow the step when it is being dragged to a new position; it has no impact on system processing. When disabled, the visual effect is that the step appears to be unconnected.

   - **Scroll Viewport while Dragging Check Box**

     Enables a visual effect that automatically scrolls the view in the **Graphical DSA Builder** pane when you drag a node to the edge of the pane. This is useful for DSAs that are larger than the pane. When deselect, you must

manually scroll the display when you drag a node to the edge of the pane to view the rest of the DSA and continue dragging the node.

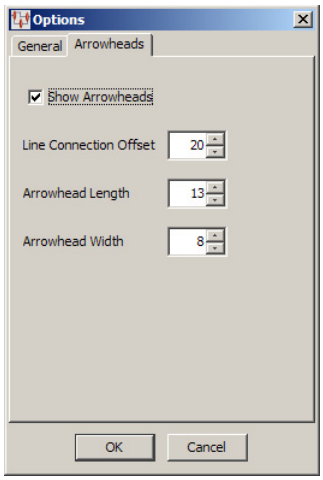
- **Ghosting Percent List**

  Allows you to select the level of ghosting on the steps when the **Show All Lines** option is not selected. This option is set to 100% by default. For more information about toggling the lines in the Graphical DSA Builder, see "Graphical DSA Builder Pane" on page 1-11.

- **Test Delay List**

  This setting controls the delay between consecutive steps changing color during single-line testing of a map. This option is set to 0% by default.

3. Click the **Arrowheads** tab.



4. Select the available controls as appropriate from the following:

   - **Show Arrowheads Check Box**

     Allows you to toggle the arrowheads on the DSA connection lines on and off.

   - **Line Connection Offset List**

     Allows you to set how far away the connection line is from the step. The maximum is 40 pixels.

   - **Arrowhead Length and Width Lists**

     Allows you to set how long and wide you want the arrowheads to appear in the Graphical DSA and Map Builders. The maximum length is 20 pixels; the maximum width is 12 pixels.

5. Click **OK**.

All of your selections are immediately implemented.

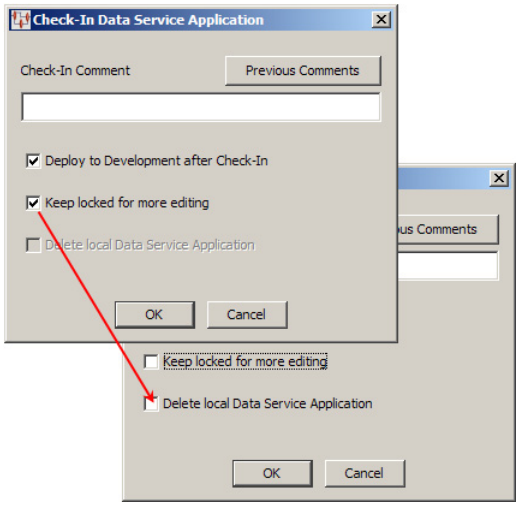
DSA Actions

The following sections explain the actions you must take so that your DSA interacts with your Oracle DataLens Server.

Checking In a DSA

Check in the current DSA to the Oracle DataLens Server so that you can use your DSA to process data.

1. From the **DSA** menu, click **Check-In…** or use the toolbar button of the same name.



2. Enter a check in comment, which will be associated with the new revision.

   You can use the **Previous Comments** button to view and copy the text of other comments for use in the new check-in comment.

   Tip: Change history can also be viewed on the Oracle DataLens Server Administrator Web pages. For more information, see *Oracle Product Data Quality Oracle DataLens Server Administration Guide*.

3. Make selections from the check boxes that function as follows:

   - **Deploy to Development after Check-In**

     The DSA is available for use by others after check-in. This allows you to deploy the DSA to the Oracle DataLens Server for testing or not deploy to the Oracle DataLens Server it if you are simply checking in work-in-progress.

   - **Keep locked for more editing**

     The DSA remains dimmed for editing. Clearing this check box will make the DSA 'read-only' and activates the **Delete local Data Service Application** check box.

   - **Delete local Data Service Application**

     Deletes this DSA from your local drive.

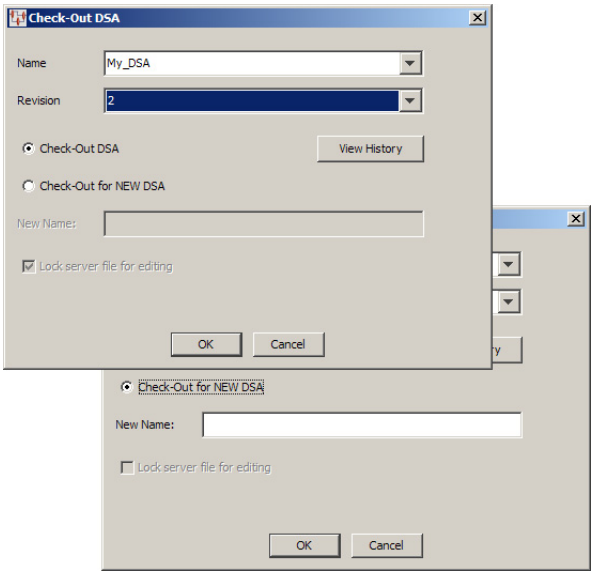
4. Click **OK**.

Upon successful check-in and deployment, the **Status Field** in the client workspace frame is updated.

Checking Out a DSA

Checks out a local copy of a DSA from the current Oracle DataLens Server for review, maintenance, or as a new DSA.

1. From the **DSA** menu, click **Check-Out…** or use the toolbar button of the same name.



2. Select a DSA from the **Name** list.
3. Select the DSA revision that you want to check out. Enter a check in comment, which will be associated with the new revision.

   You can use the **View History** button to view and copy the text of other comments for use in the new check-in comment.
4. Select the appropriate option for the action you want to take.

   If you select **Check-Out for NEW DSA**, a new DSA will be created based on the DSA revision you have selected and you must enter a name for the new DSA.

> **Note:** The Oracle DataLens Server file is automatically locked for editing only when you select the **Check-Out DSA** option.

5. If you want to lock the DSA on the server, select the **Lock Server file for editing** check box.
6. Click **OK**.

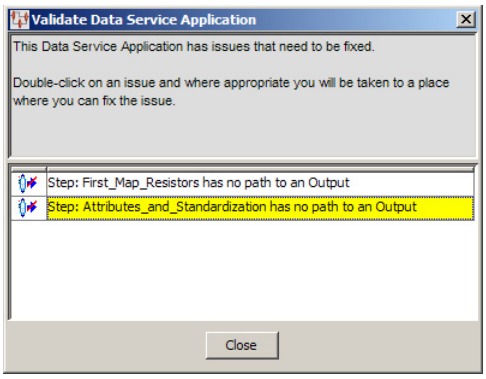
The selected DSA is checked out or created and opened for editing.

Validating a DSA

Validates the entire DSA and informs you of any issues found so that you can correct them to ensure that the DSA operates properly.

From the **DSA** menu, select **Validate** or use the toolbar button of the same name.

The DSA is validated. A message appears indicating that there were no problems found or listing all errors found. The specific steps and transformation types found to be in error are reported.



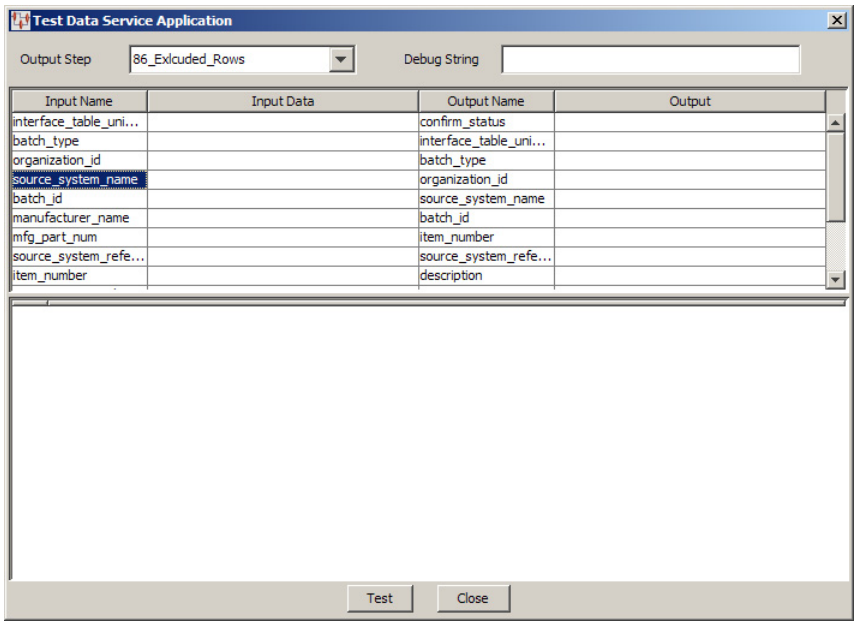
The most common errors are steps that do not flow to an output, and mismatches between the output of a step and the input of the next.

> **Note:** Successful map validation is required prior to checking in a DSA.

Testing a DSA

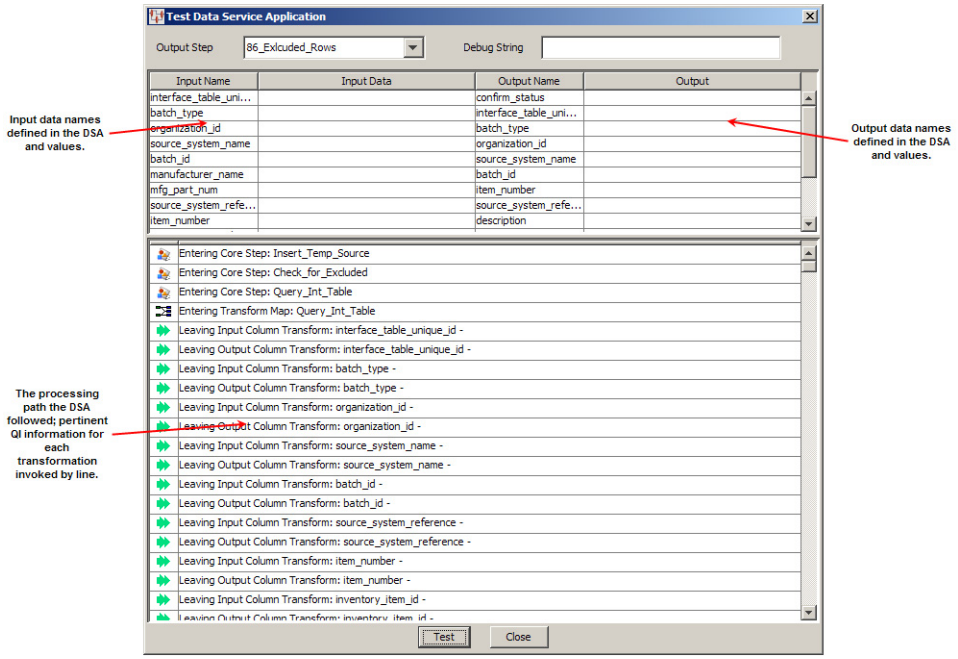
You must first successfully validate you DSA prior to testing. To test the open DSA, select **Test Data Service Application** from the **DSA** menu or click the button of the same name on the toolbar.

> **Note:** If you have unsaved changes or errors are found when the save is attempted, you are alerted and can choose to correct these errors by clicking **No** or **Cancel**.



The **Test Data Service Application** dialog box appears and is used as follows:

1. Select an output step to test from the **Output Step** list.
2. Click **Test** to review your testing results.



You can enter data into any of the **Input Data** fields then click the **Test** button to view the results in the corresponding **Output** field. This can help you verify the operation of the map and isolate any errors that may exist.

Note: The **Debug String** functionality is for use by Oracle Consulting Services *only*.

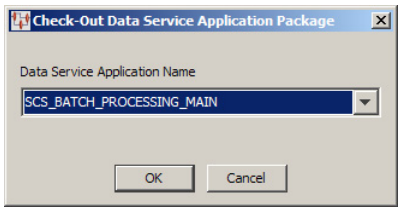
---

3. Click **Close** to conclude testing.

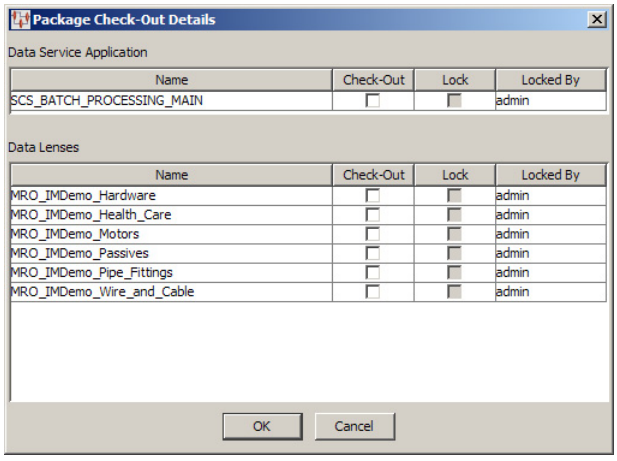
Checking Out DSA Packages

You can check out a DSA and all or some of the data lenses associated with it at one time.

From the **DSA** menu, select **Check-Out Package**.



Select the DSA that you want to check out and click **OK**.



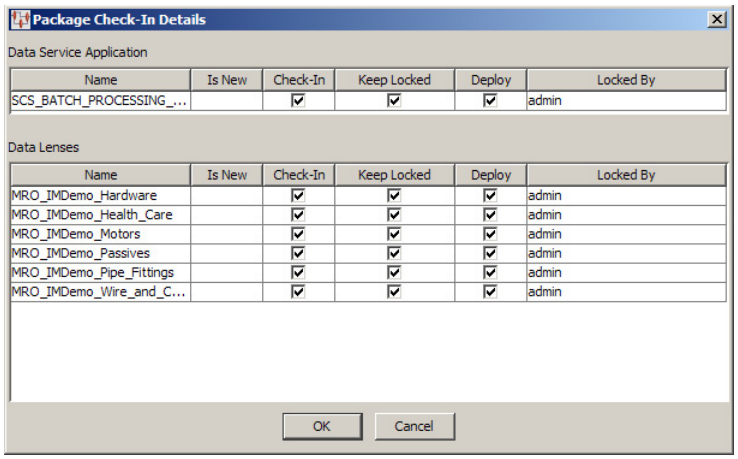
Next, select the check boxes for each individual component to check-out, which automatically sets the component to be locked upon check out. Click **OK** to check out all selected components to your local machine for modification.

A verification prompt appears. If you select **OK**, the check out continues; selecting **Cancel** stops the action. A progress dialog box appears so that you can view the checkouts as they occur and details any errors found.

Checking In DSA Packages

Similar to checking out a DSA package, you can check in a DSA and all or some of the data lenses associated with it at one time.

From the **DSA** menu, select **Check-In Package**.

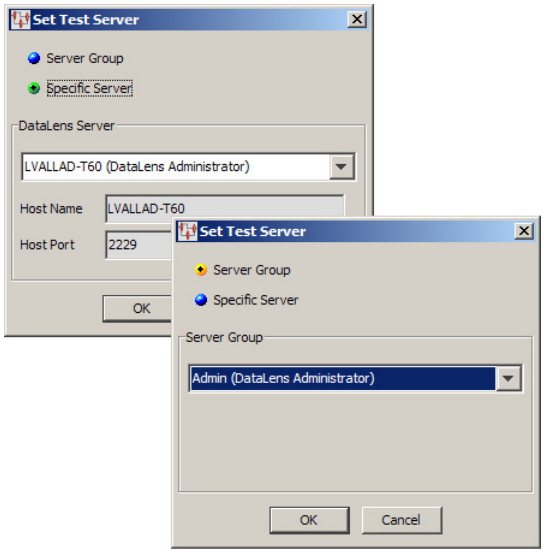


Use the check boxes to select whether you want the DSA and/or the data lenses checked in, kept locked for more editing, or deployed. Click **OK** to execute the actions that you have selected. A progress dialog box appears so that you can view the check ins as they occur and details any errors found.

Setting the Test Server

You use this option to set the test server to the location of the Oracle DataLens Server that you want to use to test your DSAs and maps.

1. From the **Tools** menu, click **Set Test Server**.



You can choose to select a specific Oracle DataLens Server or a group of servers to use for DSA testing.

2. Select either the **Server Group** or **Specific Server** option.
3. Click the down arrow to view your Oracle DataLens Server options, and select the appropriate server.

If selecting a server group, you can choose from the Administrator, Development, Production, or Quality Assurance Server Groups. The Administrator Server Group is the default.

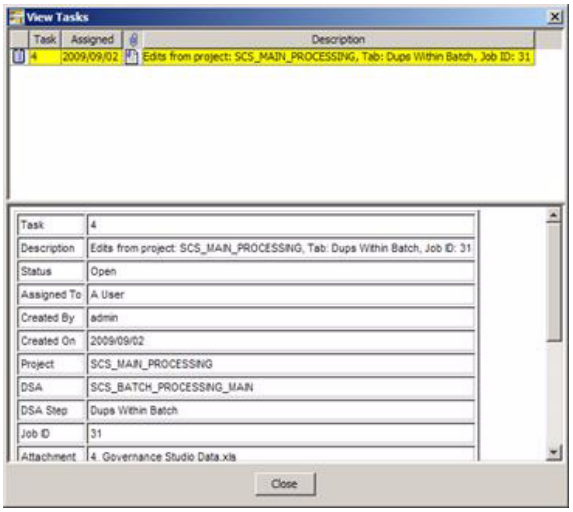
4. Click **OK**.

The test server is contacted and this new Oracle DataLens Server is used for all testing.

View My Tasks

You can see if you have any tasks assigned to use with this feature.

From the **View** menu, select **View My Tasks**.



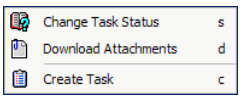
All assigned tasks are displayed in the top pane, while the bottom pane provides the details for the selected task.

> **Note:** Though the fields in the bottom pane appear to be editable, the changes are not saved.

The context-sensitive menu in the top pane is activated by right-clicking the attachment icon and is used as follows:

Change Task Status

See "Changing the Task Status" on page 3-11.



Download Attachments

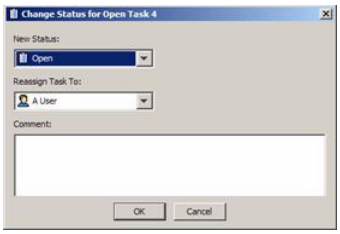
You can download the file that was saved when the task was created for use in completing the task. A file save dialog appears for you to select the directory in which to save the file.

Create Tasks

See "Creating a Task" on page 3-12.

Changing the Task Status

Selecting this option allows you to change the status of the task and/or reassign the task to another user.



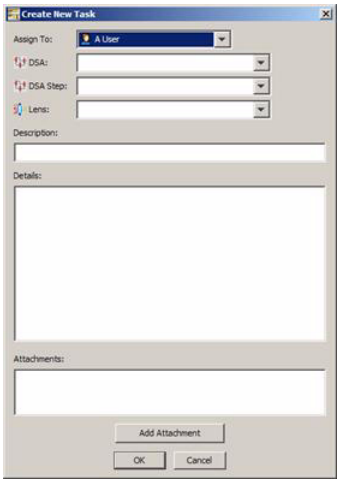
1. Select a new status and/or a user to reassign the task to from the list.

   Tip: You can use the **Unassigned Tasks** user if you are unsure who you want to review this task and intend to assign it the proper person later.

2. Enter a comment that reflects why you have affected the change for future reference or to alert the new recipient of the task why they are now responsible for it.
3. Click **OK**.

Creating a Task

Selecting this option allows you to create an entirely new review task.



1. Select a user to complete this task.
2. Select the DSA and the DSA step that you want to change.
3. Select the data lens to which the change is to be applied.
4. Enter a description and specific instructions on how to perform the task.
5. If you have a data file or other information that you want to attach, click **Add Attachment**, locate the file, and then click **OK**.

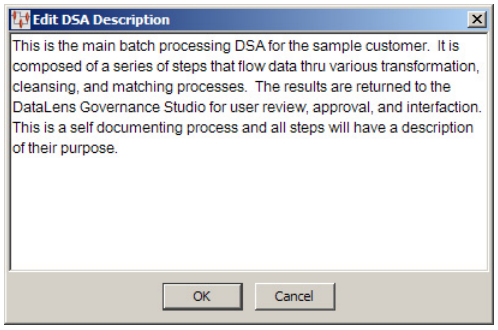
   Repeat this step until all necessary files are attached.
6. Click **OK**.

The task is created and an email containing the task details is sent to the assigned user.

Modifying the DSA Description

The description that was provided when you created a DSA can be changed to provide a different or more detailed description as to the use of the DSA.

1. From the **Edit** menu, click **Edit Description**.



2. Modify or add to the existing description of the DSA
3. Click **OK**.

4

Transformation Map Builder

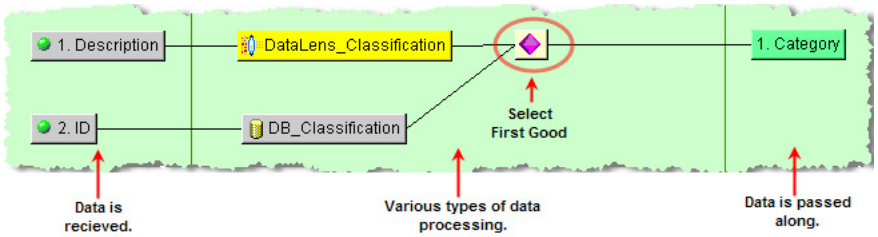
This chapter explains how to create and use a transformation map in your DSA.

Transformation Map Overview

Transformation (or Transform) Maps operate in terms of data lenses, existing databases, web services, a collection of control operations on these knowledge repositories, string operations, and mathematical operations.

Base control flow of Application Studio is from left to right, input to output, and top to bottom. Top to bottom control typically relates to the equivalent of the "if-then" control structure in a programming language. In other words, if the top transformation fails based on the quality metric value, then control moves to the next lower data lens or other transformation.

In the following example, the data lens using the DSA that contains this map classifies an item based on a description:

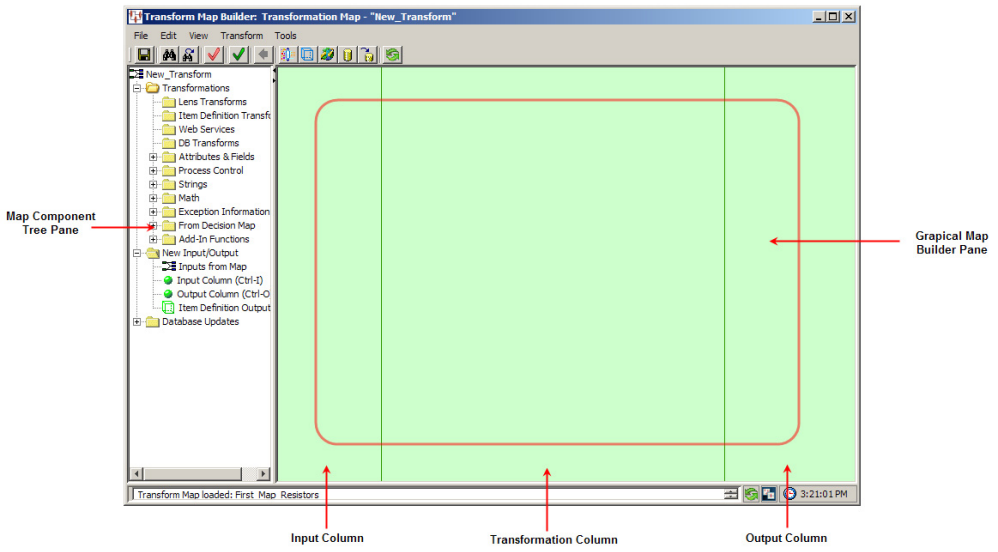


A database can also classify the same item based on its ID. The **Select First Good** decision uses the data lens classification over the database classification if they are both good because it is the top most transformation.

The Transformation Map possibilities are endless and are solely dependent on your input data and how you want to view and use the final output data. This chapter describes how to use the Transformation Map Builder to design your maps.

Understanding the Transformation Map Client Workspace

The Transformation Map client workspace is divided into two panes as follows:



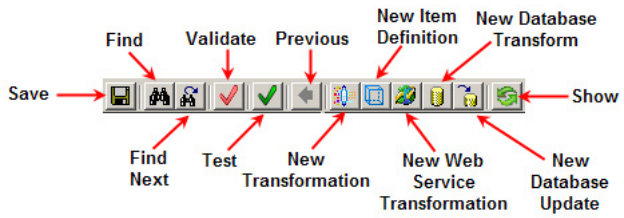
This section describes the following Transformation Map client workspace functionality:

- Transformation Map Menu and Toolbar Explained
- Map Component Tree Task Pane
- Graphical Map Builder Task Pane

The client workspace Frame Functionality is described on page 6.

Transformation Map Menu and Toolbar Explained

The Transformation Map menus and toolbar are the alternate client workspace. The following briefly describes the DSA toolbar buttons from left to right:

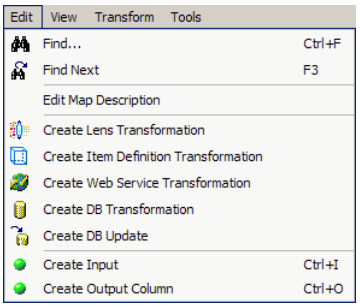


Many of the Transformation Map menus options and toolbar buttons behave the same way though on a transformation map rather than a DSA. For example, the **Save** command saves the open transformation map *not* the open DSA.

> **Tip:** The tooltips appear when you rest your mouse pointer on a menu item, button, tab, icon, or similar content.

The following section briefly describes only the Transformation Map menu commands and corresponding buttons that are in addition to the DSA commands while all others are described in the DSA Menu and Toolbar Explained section on page 8:

Edit Menu



Find…

Allows you to specify a search string (regular expression) and attempts to find it in your map. The text in the map that matches your search string is selected and highlighted in yellow

Find Next

Repeats the last search defined by a **Find** operation.

Edit Map Description

Allows you to edit the description for the open map.

Create Lens Transformation

Adds a data lens transformation to the map. Once a lens transformation is created, it appears in the Map Component Tree under the Lens Transform folder

Create Item Definition Transformation

Adds an Item Definition transformation to the Map. Once the Item Definition transformation is created, it appears in the Map Component Tree under the Item Definitions Transformation folder.

Create Web Service Transformation

Adds a Web services call to the transformation map.

Create DB Transformation

Adds a database transformation to the map. Once a database base transformation is created, it appears in the Map Component Tree under the DB Transform folder.

Create DB Update

Adds a Database update to the transformation map

Create Input

Adds a new input node in the input column.

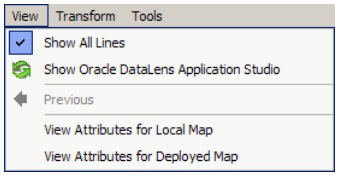
Create Output Column

Adds a new output node in the output column.

View Menu

Previous

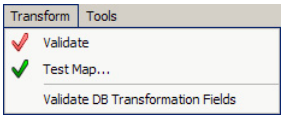
Allows you to return to the parent or main Decision Map when you have navigated away from it.



Transform Menu

Test Map

Allows you to perform a quick test of the map with a single test line of data. For more information, see "Testing and Validating Transformation Maps" on page 4-54.

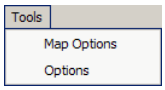


Validate DB Transformation Fields

Allows you to validate the database field's match what is being returned from the SQL query. For more information, see "Testing and Validating Transformation Maps" on page 4-54.

Tools Menu

Map Options



Allows you to set options for the open map. For more information, see "Map Options" on page 4-56.

Options

Allows you to set options for global use in the Transformation Map workspace. For more information, see "Map Options" on page 4-56.

Keyboard Shortcuts The following table contains keyboard shortcuts that can help make the Transformation Map client workspace easier to use:

Function	Shortcut Key
Save	Ctrl-S
Find	Ctrl-F
Find Next	F3
Create Input	Ctrl-I
Create Output	Ctrl-O

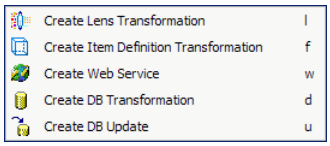
In addition, there may be single letter shortcuts that appear on context-sensitive menus that you can use to invoke the option quickly.

Map Component Tree Task Pane

The **Map Component Tree** pane contains all of the components necessary to building a transformation or decision map and is used like the DSA Component Tree pane. These components are categorized into the hierarchical tree structure with the main folder indicting the name of the current map. These components are described in Transformation Map Builder Creation Components.

Context-Sensitive Menu

You can use the **Map Component Tree** pane context-sensitive menu by right-clicking anywhere in this pane and use one of the options to add a new transformation node quickly as follows:



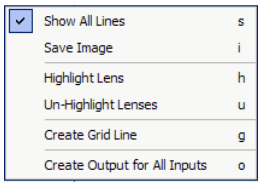
Graphical Map Builder Task Pane

The **Graphical Map Builder** pane is a vertically ruled task pane divided into three building columns: input, transform, and output. These building columns represent the end-to-end transformation process.

Context-Sensitive Menus

You can use the **Graphical Map Builder** pane context-sensitive menu by right-clicking anywhere in this pane and use the options as follows.

Show All Lines



Works as a toggle and when selected, all of the connecting lines between inputs, transformations, and outputs in the **Graphical DSA/Map Builder** pane are active. When this option is not selected, only the connecting lines for the selected node are active.

Save Image

Allows you to save an image of the Transformation Map as a JPEG file.

Highlight Lens

Colorizes (or highlights) each transformation node that specifies a data lens or lens group to use the output of the transformation. You select the data lens that you are interested in for highlighting from the list of specified data lenses. It is not necessary to use the **Un-Highlight Lenses** option before repeating the use of the highlight, though you may not discern a change if the newly selected data lens also uses the transformation because it will already be highlighted.

Un-Highlight Lenses

Removes the highlighting from the nodes that became colorized with the **Highlight Lens** option.

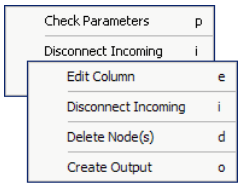
Create/Delete Grid Line

Creates a yellow vertical grid line where the insertion point was resting when you right-clicked. This menu option changes to **Delete Grid Line** when you hover over a grid line that you have created so that it can be removed. You can differentiate the lines you have created and can remove those that are created by the Application Studio by the color; only yellow lines can be removed not black lines.

Create Output for All Inputs

Creates an output column node for each input column node. For more information, see "Creating Output Nodes from Input Nodes" on page 4-20.

Check Parameters



Allows you to view the input data and SQL parameter counts of database transformation nodes and identifies any mismatches of these values.

Edit Column

Allows you to change the display name and column id of the selected input column node.

Disconnect Incoming

Removes the connection between the selected node and any that precede it in the map.

Delete Node(s)

Removes one or more selected nodes from the map. You can use the **Ctrl** key to select more than one node out of sequence; use the **Shift** key to select nodes in a sequence.

Create Output

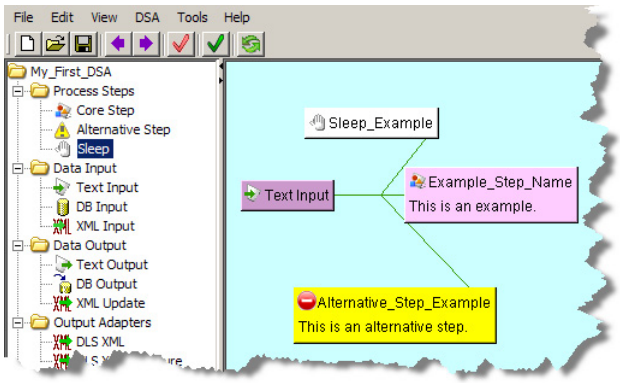
Creates an output column node for the selected input column node.

Create Outputs for Selected Inputs

Creates an output column node for each of the selected input column nodes. You can select multiple nodes as previously described. This option is active when several input column nodes are selected. For more information, see "Creating Output Nodes from Input Nodes" on page 4-20.

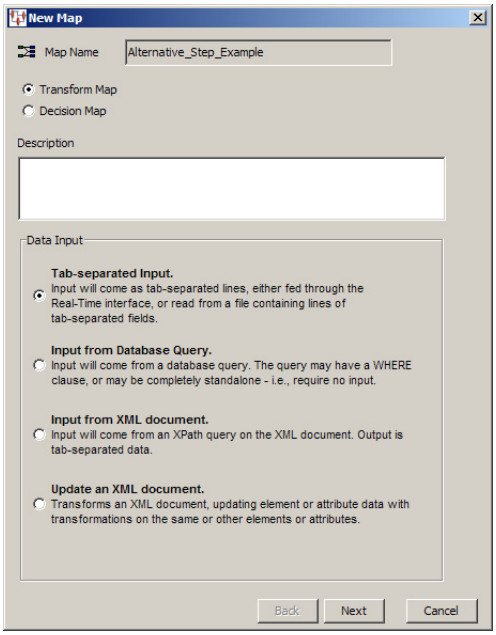
Creating Transformation Maps

The creation of a Transformation Map depends on the existence of a processing step (core or alternative) in the calling DSA as in the following:



For more information, see "Processing Steps Folder" on page 2-2.

To define a Transformation Map for a step (or open subsequently open one), you must double-click a step icon in the **Graphical DSA Builder** pane.



The name of the Transformation Map defaults to the name assigned to the calling step and *cannot* be changed.

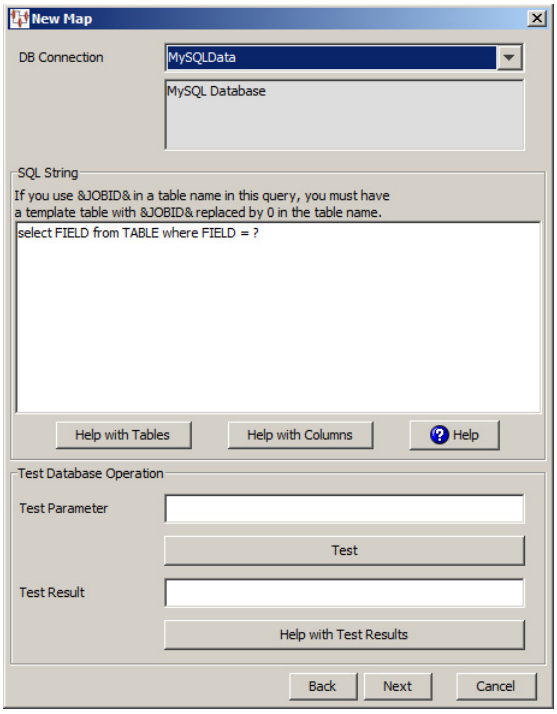
Use the **New Map** wizard as follows:

1. Select the type of map that you want to create using one of the options.
2. Enter a description for the new Transformation Map.
3. Select one of the four types of data input you want to use for this map.
4. Click **Next** to continue.

The next step varies depending on the type of map that you have chosen. Selecting **Tab-separated Input** results in a **Finish** dialog so you can click **Finish** to complete the wizard or **Back** to modify your choices. The remaining three data input options are described in the following sections.

Database Query Data Input

When you select the **Input from Database Query** data input type, you must create the database query within the **New Map** wizard.



The **New Map** dialog box appears. Use this dialog box as follows:

DB Connection

You must select the type of database connection that you want to use. The list of database connections is populated based on those that you are configured in the Oracle DataLens Server. If the type of database connection is not listed, you must configure it in the Oracle DataLens Server so that it is available for selection when creating Transformation Maps.

SQL String

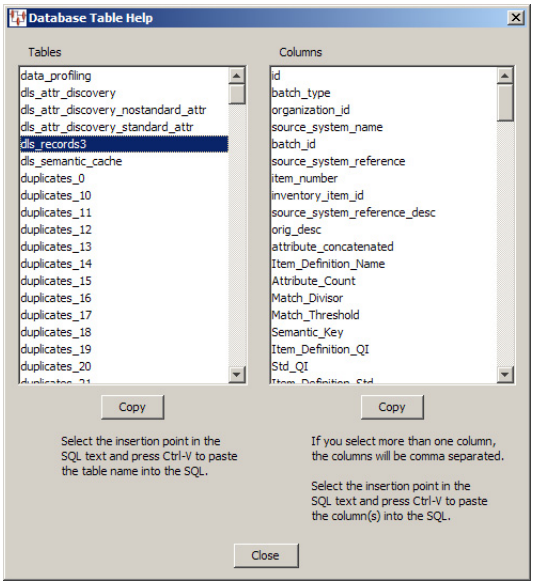
Use the **SQL String** section to construct your database query with standard SQL query statements and syntax. For example, to determine if the inventory part numbers in your input data are an exact match to a standardized set of manufacturing part numbers you could use the following SQL statements:

```
SELECT 'Exact Match Mfg Std PN'||'|'|| inventory_item_id as match_type
FROM xyz_mfg_part_numbers_all_v
WHERE upper(mfg_part_num) = upper(&?3&)
AND upper(manufacturer_name) = upper(&?2&)
AND organization_id = &?1&
AND end_date is null;
```

This `select` clause must be compatible with your database. Optionally, you can use a question mark (?) in the `select` clause. At run-time, the question mark character is replaced with the transformation input data. This allows you to create a database transformation that varies with the content of the record being processed. Additionally, when database transforms are used to aggregate data fields from several different data sources, common access key information can be used across all data sources.

Help with Tables

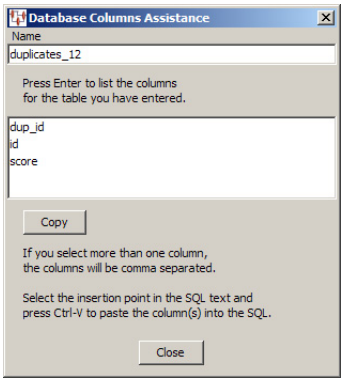
Use this button for assistance with the tables that are being used for the DB transformation.



The **Database Table Help** dialog appears. You can select a table name in the **Tables** list and view the columns for that table in the **Columns** list. You can copy information from either list to paste into the **SQL String** field. One table can be copied in the **Tables** list; one or multiple columns can be selected in the **Columns** list. Click the appropriate **Copy** button for the information you want to copy to the clipboard and click **Close**. When you are returned to the **Database Lookup Transformation** dialog box, use **Ctrl-V** to insert the copied information into the **SQL String** field at the insertion point.

Help with Columns

Use this button for assistance with the columns that are being used for the DB transformation.



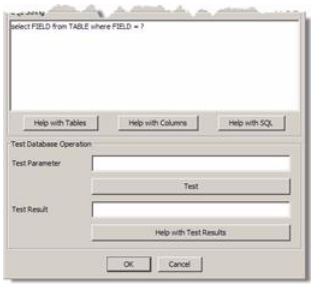
The **Database Columns Assistance** dialog appears. You can enter a table name in the **Name** field, click **OK**, and view the columns for that table. Then you can copy one or multiple columns names to the clipboard using the **Copy** button. Click **Close** to return to the **Database Lookup Transformation** dialog box and use **Ctrl-V** to insert the copied information into the **SQL String** field at the insertion point.

Help with SQL

Select this button to view brief SQL query explanations and examples.

Test Database Operation

The Test Database section is provided to help you test the setup of the data source connection. You can test the data source definition by typing a data parameter value into the **Test Parameter** field and clicking the **Test** button. The test data value you entered is substituted for the question mark character(s) in the query string, and then the query is executed. The query results are displayed in the **Test Result** field.



When the query returns more than one record, the database transformation uses only the first record returned.

If there is an error in the connection definition, an information error message appears. Click the **Help with Test Results** button for help in interpreting error messages.

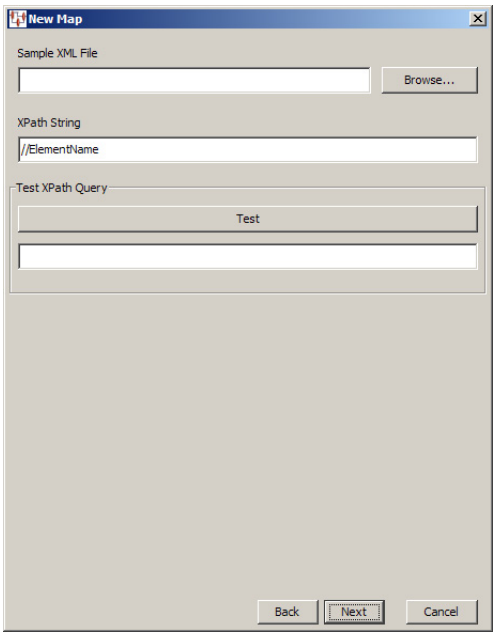
Note: The following message indicates that the database connection tested good and is not an error message:

```
Please check your database now
```

If an error message is received and there are no results in the **Test Result** field, the connection is good and the query is valid, but no matching results were returned.

XML Document Data Input

When your input data is stored in an XML file, you should select the **Input from XML Document** option so that you can use XPath to retrieve the data.



This allows you to locate and specify the XML file containing the input data. Then you enter an XPath expression to retrieve data from the specified XML file, which is then stored as tab-delimited output data.

You can test your XPath expression using the **Test** button; the results are displayed in the field below the button. For example, if your XML file contains 100 elements named `description`, you could select all of these elements with this expression:

```
//desc
```

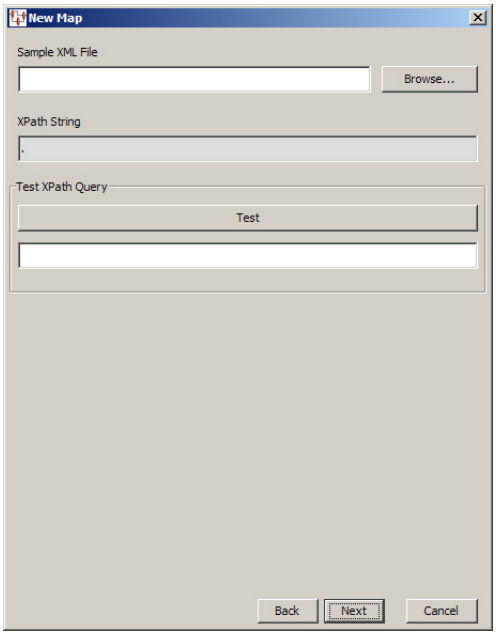
Testing this expression would result in the following:

```
100 entries found
```

> **Tip:** Delete the contents of the test result field under the **Test** button to make it easier to differentiate between repeated testing attempts.

XML Document Update Data Input

You can use the **Update an XML Document** data input option when you want to update an existing XML file to update element or attribute data for transformations that use this same data.



The functionality is the same as described in "XML Document Data Input" on page 4-9 though you cannot specify the XPath expression, which is set to select the parent element thus selecting all elements. When you have selected the XML file and completed any testing, click **Next** to continue the wizard. To complete the new map, click **Finish** and the **Transformation Map Builder** opens.

Transformation Map Builder Creation Components

All of the components (nodes and widgets) needed to design and maintain a transformation map are contained in the **Map Component Tree** pane and are described in this section.

Transformations Nodes and Widgets

The Transformations folder contains the nodes and widgets that are added to the Transformation Column to convert input data and ready it for output. The following sections describe each of these nodes and their associated functionality.

Container Folders

The following folders contain the Transformation Maps (by category) that you have created within the open DSA as follows:

- Lens Transforms
- Item Definition Transforms
- Web Services
- DB Transforms

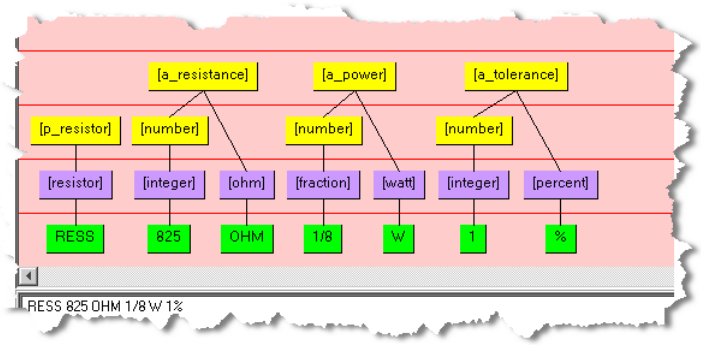
Attributes& Fields Widgets

This section describes the various attributes and field widgets that can be used in your Transformation Maps. For more information about using these controls in your map, see "Transforming Data Using Attributes and Fields" on page 4-46.

Lens Attributes

Use with a defined lens transformation to extract the standardized data spanned by a phrase or term rule. The Lens Attribute widget uses one of the phrases or terms found in the selected data lens transformation.

The following is an example of the rules created in a data lens:



If a Lens Attribute is defined to extract text using the `[a_resistance]` phrase, then the resulting text is 825 OHM; no standardization is applied.

Item Definition Attribute

Use with a defined lens transformation that provides attributes to allow you to extract standardized data based on an Item Definition. The attributes are listed within their defined Item Definition for review and selection.

DB Field

Use with a defined database transformation to allow you to incorporate several fields from the results of the database transformation query. The DB Field widget uses one of the fields found in the query of the selected database transformation.

Lens Classification

Use with a defined lens transformation that provides classification information to allow you to define the Classification Code and Name and the Level of Classification to be used. Values can be Category and Level 1 - 5.

Quality Index (QI)

Use to retrieve the quality index from the transformation.

Process Control Widgets

This section describes the various process control widgets that can be used in your Transformation Maps. For more information about using these controls in your map, see "Transforming Data Using Processing Controls" on page 4-48.

Match

Allows you to filter the results of a transformation based on a string match that you specify.

Arithmetic Match

Allows you to use numerical comparisons to identify the data to be output.

Select First Good

Allows you to select the first acceptable quality result, which is the first transformation result that meets or exceeds its quality indices, from several transformations. The data transformed by the selected transformation is passed on to the next step in the transformation process.

Convert Exception

Provides you with additional flexibility when dealing with record exceptions. Record exceptions occur when a record does not meet or exceed one or more of the transformation quality indices. The **Convert Exceptions** widget allows you ignore certain exceptions by setting a conversion text string. The default data output from this widget is an empty string.

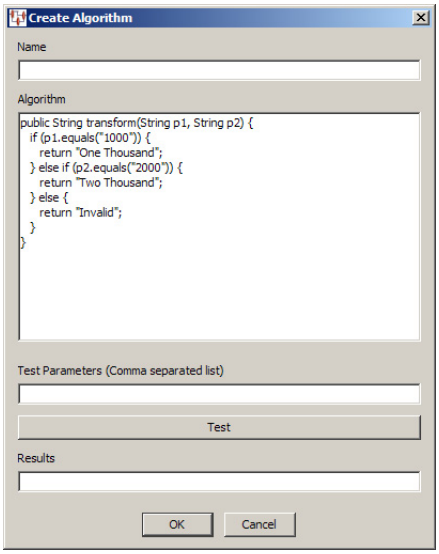
Null -> Exception

Allows you to identify empty data records or database null values to force exception processing.

Algorithm

Allows you to process data using an algorithm that is Java-based processing, which you devise and enter. The algorithm can only include Java scripting employing the use of standard Java classes.

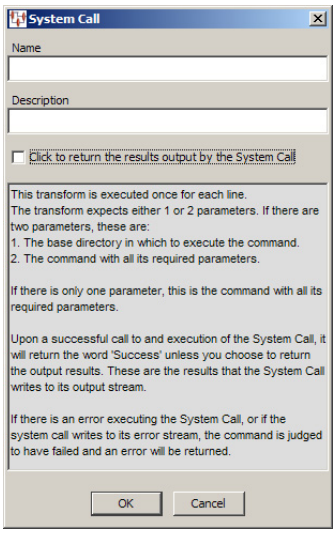
The **Create Algorithm** dialog box prompts you to enter a name for the widget and the algorithm. You can test the validity of your algorithm by entering testing parameters and clicking **Test**; the results appear in the **Results** field.



System Call

Allows you to run Windows or Linux system calls to use external processing in your Transformation Maps. This can be useful to use your custom scripts to process data that cannot be otherwise processed using the available transformation widgets. For example, you could run a shell script or command on Linux or a batch file or command on Windows.

The **System Call** dialog box prompts you to enter a name for the widget and a description of the process being called. You can select the check box to have the output that results from the system call returned or by default, the word Success is returned.



The parameters necessary for this node are the directory path of the system process you want to call and the command itself. These are provided using database input, literal string, text or other input nodes. Likewise, the Output nodes receive the results of the system call to be passed along in the processing.

Strings Widgets

This section describes the various string widgets that can be used in your Transformation Maps. For more information about using these controls in your map, see "Transforming Data Using String Operations" on page 4-49.

No Change

Allows you to pass the data through the map without any changes.

Literal String

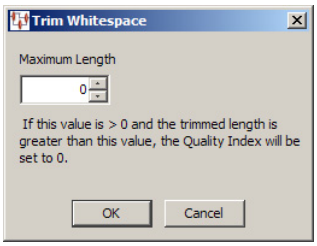
Allows you to insert static text strings into your processing flow. This operator can be used to provide default text or labels for defined outputs.

Substring

Allows you to select a portion of an input string for processing. You select the starting position in the string, the number of characters to extract, and if any whitespace is automatically trimmed.

Trim

Allows you to trim leading and trailing spaces from the field it processes by selecting the **Maximum Length** the string will be after it is trimmed. If the data string is zero (0) and the maximum length is something other than zero then the Quality Index is set to zero.



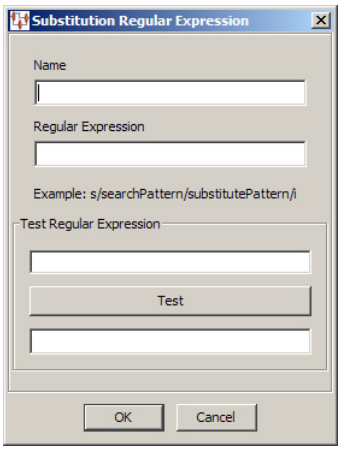
Convert Case
Allows you to convert the case of the input into one of three different case control types: lower, upper, or proper.

Replace
Allows you to search for a string that you specify and replace it with the replacement string that you designate.

Replace Null
Allows you to search for strings that are empty (or null) or contain only spaces, and then replace it with the replacement string that you designate.

Logical Replace
Allows you to compare string one with string two and can apply replacements for both the true and false results.

Regular Expression
Allows you to use regular expressions to match and replace strings using standard Regular Expression syntax. Enter a name and the regular expression that you want to use. You can test the validity of your regular expression by entering the testing parameters and clicking **Test**; the results appear in the **Results** field



For more information, see "Regular Expressions" on page B-1.

Concatenate
Allows you to merge together two or more strings for further transformation. The strings are merged with a single space separating each input string

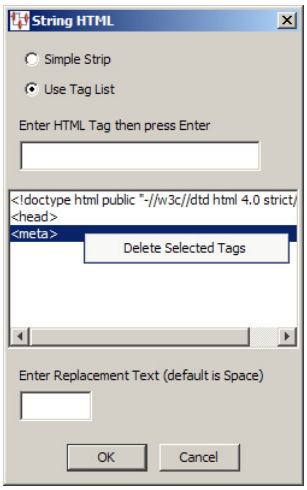
Extract
Allows you to extract the data in a specific field. Enter a name for the widget, the character that separates the input data fields, and select (or enter) the number of the field that contains the data to be extracted.

Splitter
Allows you to prepare the input field to be split into two or more split fields; this widget is used in conjunction with the **Split Field**.

Split Field
Use with the results of a **Splitter** widget to extract one or more characters intelligently respecting white space boundaries.

Strip HTML

Allows you to build a regular expression to strip HTML tags or data and replace it with values you provide or a space.



You can use a **Simple Strip** to replace anything between the less than (<) and greater than (>) symbols, or **Use Tag List** to provide the exact HTML tag you want to replace.

If you use the **Use Tag List** option, then you can enter a tag that you want to replace and press **Enter**. Repeat for each tag you want to replace. Selecting a tag in the list and right-clicking on it allows you to delete the tag from the replacement list.

Enter any replacement text or use the default replacement, which is a space.

Math Widgets

All the arithmetic functions produce record exceptions if the input values are not appropriate for the function. Each can be set to a desired decimal precision in which to return the results. For more information, see "Transforming Data Using Math Operations" on page 4-54.

Arithmetic

This widget allows you to select one of four arithmetic operations: **Add**, **Subtract**, **Multiple**, or **Divide** two values. You can assign the number of decimal digits the results will be calculated to or accept the default of zero.

Rounding

This widget allows you to select one of three rounding operations, Round Up, Round Down, or Round to Nearest. You can assign the number of decimal digits the results will be calculated to or accept the default of zero.

Minimum/Maximum

Allows you to select whether the data is set to the minimum or maximum of the numerical input value.

Exception Information Widgets

These widgets are used to receive information about exceptions from previous maps in the map flow. When used in a map, these widgets receive information about the Transformation Map that received the exception, the Transformation inside the map that received the exception, and the data lens involved in the Transformation (if any) that received the exception.

Exception Map
This widget receives the Transformation Map that created the exception upstream in the map flow.

Exception Transform
This widget receives the name of the data lens, DB, Web service, or processing step inside the Transformation Map that created the exception received by this map.

Exception Lens
Receives the name of the data lens inside the Exception Transform (if any) that created the exception received by the map.

From Decision Map Widgets
These widgets pass results from a parent Decision Map to a child map. For more information, see "Decision Map Builder" on page 5-1.

Decision Data
Passes decision data results. This eliminates the need to transform fields in the same record multiple times.

Decision QI
Passes data quality index results to the next widget. This allows for analysis of results from parent decisions maps so that you can identify why the decision was made.

Decision Item Definition
Passes all of the Item Definition attribute information from a previous decision map. This allows attributes to be passed from one map to another and provides all attributes defined within an Item Definition.

Add-In Functions Widgets
These widgets provide specialized functions for your use.

Oracle AU XML Parameterizer
Integrates an Oracle database with Oracle Product Data Quality. To activate this functionality you must contact Oracle Consulting Services.

Get Field
Allows you to retrieve one specific field from an input data record. The widget that you connect the **Get Field** widget to must pass the specified string from a field. The field index, field separator, and default value are specified in the fixed parameters. Typically, this is used in a logic decision in a map.

New Input/Output Nodes
The nodes in this folder vary depending on the type of map that you have selected, text, database, or XML. The following is a comprehensive list of all input nodes included in the Application Studio:

Input from Map
An input node is a placeholder that indicates that the input data is from another map.

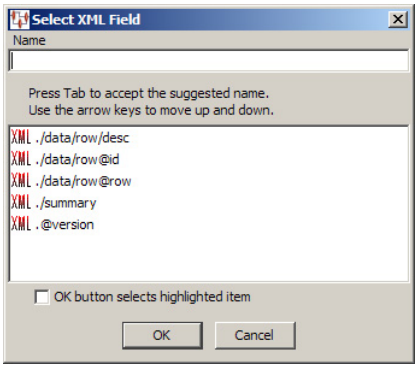
Input Column
An input column node is a placeholder that indicates the type of input data to be used for the map.

Output Column
An output column node receives data from an input or transform node to pass to another DSA step.

Item Definition Output
An Item Definition node receives attribute data from an Item Definition transform node to pass to another DSA step.

DB Field Input
A database field input node receives data from the fields in the input database to pass to transformation or output column nodes.

XML Field Input
An input node that receives data from a field in an XML input data file.



Enter a name for the node. As you type, the letters are matched against the XML Fields in the list. If a match is found, it is automatically selected and you can press **Tab** to accept the selection. If you click **OK**, your entry in the **Name** field is used unless the check box is selected, in which case the automatically selected XML Field is used rather than the text you entered.

Additionally, you can select the XML field containing the data using the arrow keys, and then press **Tab** to select that field.

XML Output Column
An output column node that passes data from a transformation node to a field in an XML data file. The same **Select XML Field** dialog box is used as previously described.

Database Updates Widgets

These widgets enable you to find and match data records, or create an output table.

Attribute Match and Match2
This widget is described in Advanced Mapping and DSA Concepts on page 153.

Attribute Find
This widget is described in Advanced Mapping and DSA Concepts on page 153

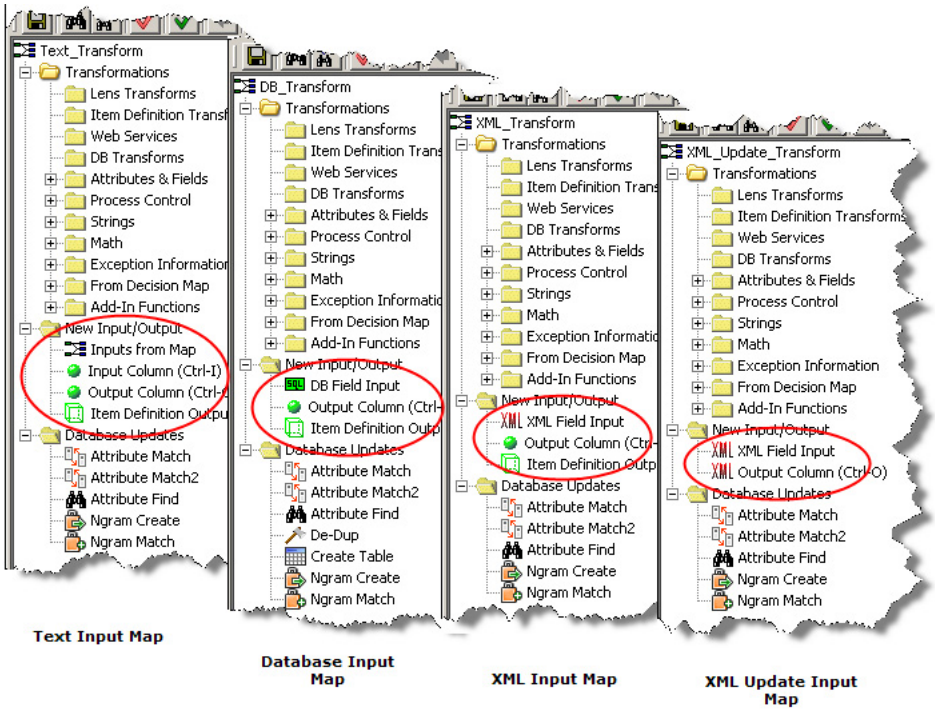
De-Dup
This widget is described in Advanced Mapping and DSA Concepts on page 153.

Defining the Input Column

Define the input column by adding input nodes to the map to be connected to a transformation column node or an output column node. Input nodes are essentially placeholders that indicate the type of data input.

Adding Input Column Nodes

The nodes that appear in the New Input/Output folder of the **Map Component Tree** pane vary depending on the type of map input that you selected when creating the map as described in "Creating Transformation Maps" on page 4-5.

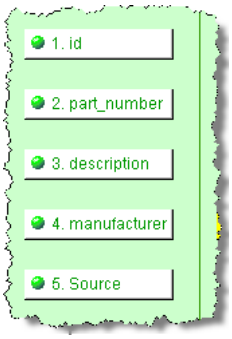


The Application Studio intuitively changes the types of nodes (input, output, and database update) needed for each map and only those nodes are available for ease of use.

You can add nodes to the input column of your map by double-clicking on the **Input Column** node in the **Map Component Tree** pane, or dragging a node and dropping it in the **Graphical Map Builder** pane, or by using **Ctrl-I**.

Enter a descriptive name for the node and click **OK**.

The new input node is now ready to be connected to a transformation node to supply data to it. The number of input nodes that you create is solely dependent on your input data though typically there are several input nodes to effectively process data.

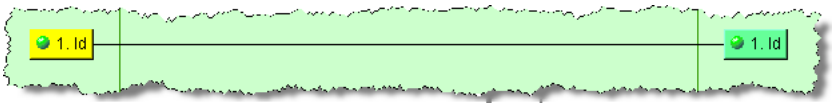


Tip: All unconnected nodes in Transformation Maps are colored white, connected nodes are green, and selected nodes are yellow.

Creating Output Nodes from Input Nodes

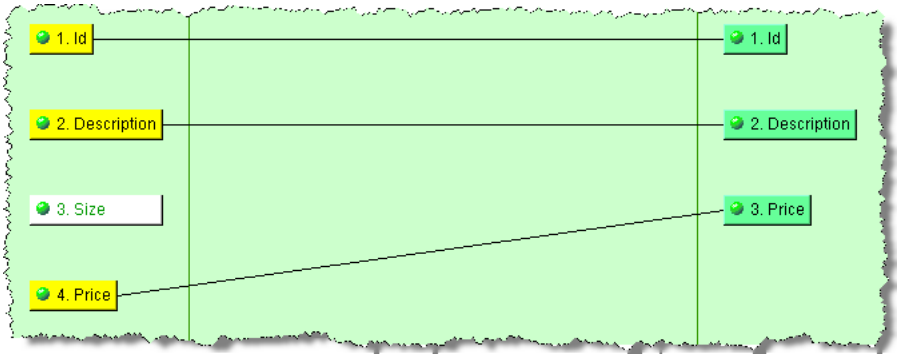
You can automatically create and connect output nodes for one or more input nodes.

To create an output node for a single input node, right-click on the input node, and then select **Create Output**.



The output node is created with the same name as the input node and the nodes are connected.

You can select multiple input nodes and then create output nodes using use the **Shift** (selecting continuously) or the **Ctrl** (selecting discontinuously) keys while selecting the nodes, right-clicking on one of them, and then selecting **Create Outputs for Selected Inputs**.



Output nodes are created for the selected input nodes appear and are logically connected to the input data.

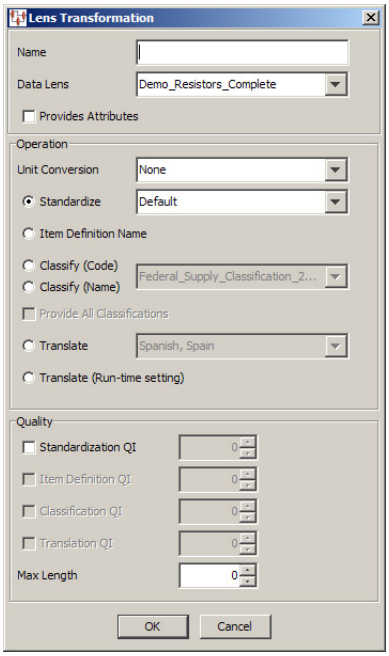
Defining the Transformation Column

The following sections explain how to define the nodes in the Transformation Column of your Transformation Maps.

Defining Lens Transforms

Lens transformations are defined by the data lenses that have been created using the Knowledge Studio.

To create a Lens Transform, from the **Edit** menu, select **Create Lens Transformation** or use the button of the same name.



Complete the **Lens Transformation** dialog as follows:

Name

Use this field to name your transformation. This name is displayed in the **Map Component Tree** pane and in the **Graphical Map Builder** pane.

Data Lens

Use this list to select the data lens to use for this transformation. This list of available data lenses is built from the data lenses selected into the Oracle DataLens Server.

Provides Attributes

Use this check box when the Lens transformation will be used to extract terms and phrases defined within the data lens. When this check box is selected, only dialog fields associated with standardization are enabled. Lens Attribute controls in the **Process Control** folder of the **Map Component Tree** pane only work with data lens transformations defined to provide attributes.

Operation Section

Select one of the following operations for this lens transform:

Unit Conversion

Use this list to transform the input data using the selected unit conversion. The list of available unit conversions is defined by the selected data lens project.

Standardize
Select this option if the input data is to be standardized, and then select one of the available standardization types from the list. This list is populated by the standardization types contained in the selected data lens.

Classify (Code) or Classify (Name)
Select one of these options if the input data is to be classified, and then select one of the available classification types from the list. This list is populated by classification types contained in the selected data lens. For more information, see "Classification Outputs Tab" on page 4-31.

Translate
Select this option if the input data is to be translated to a target locale, and then select the translation language. The list of available translation languages is populated by the selected data lens.

Translate (Run-time setting)
Select this option if the input data is to be translated to a target locale, and you want the target locale to be selected at run-time. This function allows you to use the same map to translate the input data set into multiple languages by selecting the target locale for each transformation job.

Quality Section Modify this section as follows:

Standardization QI
When this check box is selected, the value in the adjacent value field is used to control the quality of the output records. Records that meet or exceed the standardization quality index are routed to the output records. Records that do *not* meet the standardization quality index setting are routed to the exception records file. The Standardization QI can range from 0-100.

The Standardization QI is very valuable in determining the overall quality of the transformation result. Because of this, the Standardization QI is always available no matter what transformation operation is selected in the **Operation** section. The Standardization QI value encodes how much of the input text is recognized by the data lens. When the Standardization QI is very high, 80 to 100, the data lens used in the transformation recognizes most or all of the input data. When a data lens recognizes all of the input data, its transformation will be highly accurate. The Standardization QI is often used in conjunction with the other quality indices to ensure the highest possible quality results.

Classification QI
When this check box is selected, the value in the adjacent value field is used to control the quality of the output records. Records that meet or exceed the classification quality index are routed to the output records. Records that do *not* meet the classification quality index setting are routed to the exception records file. The Classification QI can range from 0-100.

Translation QI
These controls are only active if a translation operation is selected. When the **Translation QI** check box is selected, the value in the adjacent value field is used to control the quality of the output records. Records that meet or exceed the translation quality index are routed to the output records. Records that do *not* meet the translation quality index setting are routed to the exception records file. The Translation QI can range from 0-100.

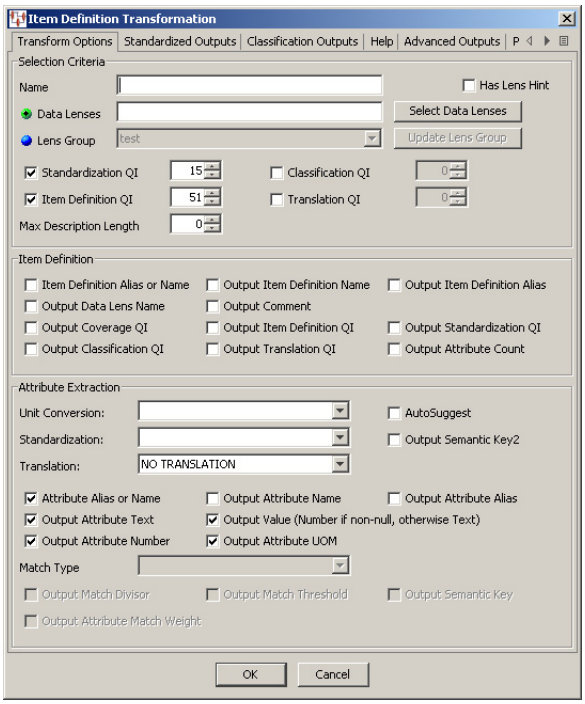
Max Length

This field is used to control the length of the transformation result. Records that are less than or equal to the Max Length value are routed to the output records. Records that are greater than the Max Length value are routed to the exception records file. A Max Length of zero (0) means that no length limitation applies.

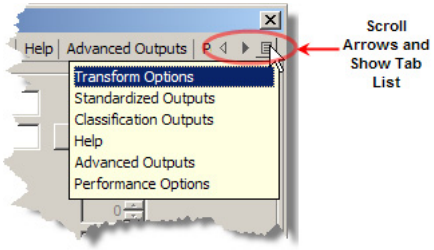
Defining Item Definition Transforms

Item Definitions provide the Item Definition name and attributes based on the Name, Text, Value, and Number for use with both the DB Update and Item Definition Output nodes.

To create an Item Definition Transform, from the **Edit** menu, select **Create Lens Transformation** or use the button of the same name.



The **Item Definition Transformation** dialog box appears. To navigate the tabs, you can click the left and right **Scroll Arrows** at the right to scroll to the right or back to the left. In addition, you can click the **Show Tab List** button, as shown in following figure, to see a list of all tabs to select them individually:



Use this dialog box as described in the following sections.

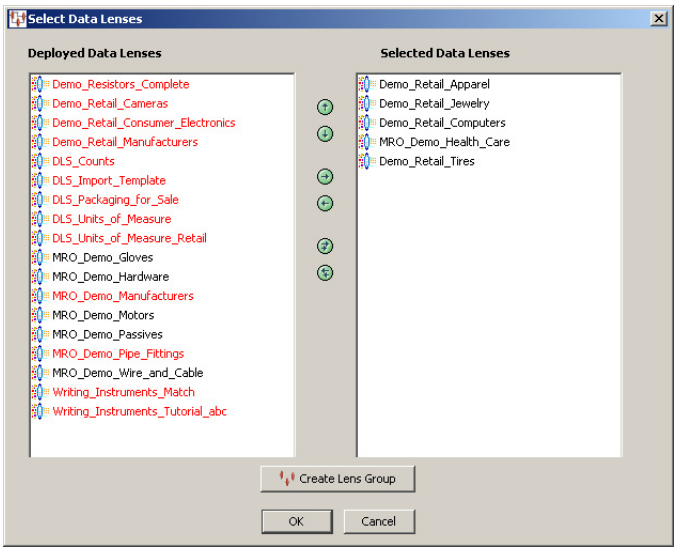
Transform Options Tab

Selection Criteria Section Modify this section as follows:

Name
Enter a name for this transformation. This name is displayed in the **Map Component Tree** pane and in the **Graphical Map Builder** pane.

Has Lens Hint
Select this check box to set the data lens that will be used first in the transformation. This feature optimizes the processing of data when your Lens Transformation has multiple data lenses or a Lens Group to use during processing by setting the data lens that is used first. This data lens is the one most likely to process the data successfully therefore the remaining data lenses do not have to be used for processing. The data lens that is first in the selected list of multiple data lenses (created using the **Select Data Lens** button) is set as the lens hint.

Data Lenses
Enter the data lens to use for this transformation. To use multiple data lenses or a lens group, use the **Select Data Lens** button to select data lenses.



The **Select Data Lenses** dialog box appears. The list of available (deployed) data lenses is populated based on the data lenses deployed into the DataLens Administrator.

Data Lenses are moved between list boxes using the right and left arrows or by double-clicking on a data lens. The up and down arrows are used to change the data lens processing order. Any deployed data lenses that appear in red are not compatible with the selected data lens because the Unit Conversion, Standardization, or Match types are different.

If the **Has Lens Hint** check box is selected, the first data lens in the list of **Selected Data Lenses** is used first to process the data.

Creating a Lens Group You can create a specific group of data lenses for use with all Item Definition transformations in your DSA instead of selecting them individually in each transform. A lens group can be used in one or multiple DSA steps and in multiple DSAs

Lens groups allow you to update the lens group in one transformation that automatically updates all other transformations using the lens group, which avoids the need for each Transformation Map throughout the DSA when you want to add or

remove a data lens. This feature greatly increases consistency and reduces errors in DSAs.

To create an Item Definition transformation lens group:

1. Click the **Select Data Lens** button.
2. Add the first of the data lenses that you want to group to the **Selected Data Lenses** list.
3. Click **OK** to return to the **Item Definition Transformation** dialog box. This allows the Application Studio to determine the other data lens that are compatible with your selection.
4. Click **Select Data Lens**.
5. Add the remaining data lenses that you want to include in this lens group.
6. Click **Create Lens Group**.

7. Enter a descriptive name for the new lens group and a description that indicates its purpose. Both of these fields must be completed.
8. Click **OK**. An informational message appears to alert you that the new lens group has been created.
9. The Oracle DataLens Server is updated to include this data lens. You can view and delete all lens groups from the Oracle DataLens Server though you must edit them in the Application Studio.
10. Click **OK** to return to the **Item Definition Transformation** dialog box.
11. The Lens Group field is updated to display the lens group that you just created though it is not active.
12. Click the **Lens Group** option to activate this field and the **Update Lens Group** button.

> **Note:** To use this lens group in other Item Definition transformations, you must edit each transformation and select it using the **Lens Group** controls.

Lens Group

Select this option to activate the field and the **Update Lens Group** button. You can select one of the listed lens groups to use for this transformation. If there are no lens groups listed, you can create them as previously described.

- **Update Lens Group**

You can edit your lens group by editing any one of the Item Definition transformations that use it. Updating a lens group in one transformation permeates throughout the DSA in one simple action.

Clicking the **Update Lens Group** button displays the **Select Lenses for Lens Group:** ***Lens Group Name*** dialog box. This dialog box operates identically to the **Select Data Lenses** dialog box previously described. You can add or remove data lenses to reconfigure the lens groups.

Note: Lens groups can only be deleted using the Oracle DataLens Server. For more information, see *Oracle Product Data Quality Oracle DataLens Server Administration Guide.*

Standardization QI

When this check box is selected, the value in the adjacent value field is used to control the quality of the output records. Records that meet or exceed the standardization QI are routed to the output records. Records that do *not* meet the standardization quality index setting are routed to the exception records file. The Standardization QI range that can be defined is 0-100.

Item Definition QI

When this check box is selected, the value in the adjacent value field is used to control the quality of the output records. The Item Definition QI encodes how much of the input text is recognized by the data lens based on the Required and Scoring Attributes that have been defined within each of the Item Definition values. The range that can be defined is 0-100.

Maximum Description Length

When this field is used, the transformation verifies that the Description Length, based on either the Non-Item Definition Standardized Description or the Item Definition Standardization Description, does not exceed the value defined in this field. The Maximum Description Length that can be defined is 0-500.

Classification QI

When this field is used, the transform provides an understanding about whether that record has been classified. The value is 100 if classified once or 100 divided by the number of classifications found. The range that can be defined is 0-100.

Translation QI

When this field is used, the transform provides an understanding of how many attributes have been translated correctly. The range that can be defined is 0-100.

Item Definition Section The following tables describe the Item Definition dialog box output check boxes and provide brief descriptions, the output column order, and the output name. The tables are in a top to bottom, then left to right order. The order is based on all of the outputs being selected. If there are fewer outputs selected then the order will be adjusted accordingly.

Check box	Description	Output Column Order	Output Name
Item Definition Alias or Name	Select to output either the Item Definition Alias or Name. The Alias is output if present; otherwise, the Name is output. You can select this check box or the **Output Data Lens Name** check box not both.	1	`Item_Definition_Name` `Item_Definition_Alias`
Output Data Lens Name	Select to output the data lens name. You can select this check box or the **Item Definition Alias or Name** check box not both. This option must be set to use the attribute search functionality provided by the **Attribute Find** control.	2	`Lens_Name`
Output Coverage QI	Select to output the Coverage QI, which is the percentage of non-white space characters that are recognized (covered) and placed into attribute in an Item Definition. This includes optional attributes.	9	`Coverage_QI`
Output Classification QI	Select to output the Classification QI.	12	`Cls_QI`
Output Item Definition Name	Select to output the Item Definition name. This option must be set to use the attribute search functionality provided by the **Attribute Find** control.	1	`Item_Definition_Name`
Output Comment	Select to output the Item Definition description.	3	`Comment`
Output Item Definition QI	Select to output the Item Definition QI; it must be at least 51.	10	`Item_Definition_QI`
Output Translation QI	Select to output the Translation QI.	13	`Trn_QI`
Output Item Definition Alias	Select to output the Item Definition alias.	1	`Item_Definition_Alias`

Check box	Description	Output Column Order	Output Name
Output Standardization QI	Select to output the Standardization QI.	11	`Std_QI`
Output Attribute Count	Select to output the Attribute Count.	4	`Attribute_Count`

Attribute Extraction Section

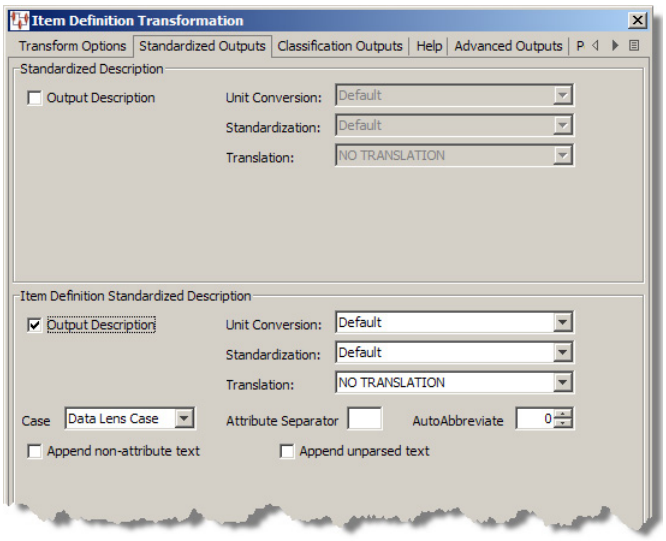
Control	Description	Output Column Order	Output Name
Unit Conversion	Select the appropriate Unit of Measure type for use with unit of measure conversions for standardized values for the attributes that are being transformed.	N/A	N/A
Standardization	Select the appropriate Standardization for the attributes that are being transformed.	N/A	N/A
Translation	Select the appropriate Translation for the attributes that are being transformed.	N/A	N/A
AutoSuggest	Select to output suggestions as an XML structure in the **Attribute Text** field where a suggestion is available. The output is generated in a format for use in the Governance Studio to display in AutoSuggest tabs.	18	`Att_`*`Number`*`_Text`
Output Semantic Key2	Select to output the Semantic Key2.	8	`Semantic_Key2`
Attribute Alias or Name	Select to output the Attribute Name.	15	`Att_`*`Number`*`_Name`
Output Attribute ID	Select to output the Attribute ID.		`Att_`*`Number`*`_ID`
Output Attribute Text	Select to output the Attribute text.	18	`Att_`*`Number`*`_Text`
Output Attribute Number	Select to output the Attribute number.	19	`Att_`*`Number`*`_Number`
Output Attribute Name	Select to output the Attribute name.	15	`Att_`*`Number`*`_Name`
Output Attribute Group ID	Select to output the Attribute Group ID.		`Att_`*`Number`*`_Group_ID`
Output Value (Number if non-null; otherwise Text)	Select to output the Attribute value.	17	`Att_`*`Number`*`_Value`

Control	Description	Output Column Order	Output Name
Output Attribute UOM	Select to output the Attribute Unit of Measure.	20	`Att_Number_UOM`
Output Attribute Alias	Select to output the Attribute alias.	16	`Att_Number_Alias`
The following matching controls are not active unless a Match Type has been defined in your data lens.			
Match Type	Select the appropriate Match type for the attributes that are being transformed.		N/A
Output Match Divisor	Select to output the Match Divisor.	6	`Match_Divisor`
Output Attribute Match Weight	Select to output the Match Weight. This option must be set to use the attribute matching functionality provided by the **Attribute Match** controls.	21	`Att_Number_ Weight`
Output Match Threshold	Select to output the Match Threshold. This option must be set to use the attribute matching functionality provided by the **Attribute Match** controls.	7	`Match_Threshold`
Output Semantic Key	Select to output the Semantic Key.	8	`Semantic_Key`

Standardized Outputs Tab

The Standardized Outputs tab is no longer actively used and will be deprecated in a future release. For further information, contact Oracle Consulting Services.

Use the **Standardized Outputs** tab as follows.



The Standardized Description section of this tab is no longer actively used and will be deprecated in a future release. For further information, contact Oracle Consulting Services.

Control	Description	Output Column Order	Output Name
Output Description	Select to output the Item Definition Standard Description.	13	`Item_Definition_Std`

The Output Description is based on the selection of the following controls:

Unit Conversion
Select the appropriate Unit of Measure type that should be used for unit of measure conversions for standardized values.

Standardization
Select the appropriate Standardization type to be used for the Standardized Description.

Case
Select the correct case to be used with the standardized description. You can select one of the following options: DataLens case, uppercase, lower case, or proper case. Setting a standard case in the transform overrides the case settings in a data lens.

DataLens case is the case that is defined within the data lens itself and can be viewed in the rewrite rule for the attribute in the Knowledge Studio on the **Standardize Items** tab, on the **Standardize Attributes** sub-tab.

Attribute Separator
Enter a character value when a separator is required between each of the attributes defined within the Order Attributes for each of the Item Definitions.

AutoAbbreviate
Enter a value for the number of characters to be shown in the standardized description. The Application Studio uses an algorithm to shorten the description to the desired length while maintaining readability.

Append non-attribute text
Select this check box when the standard description should include the defined attributes within the Item Definition and any other attributes that have been identified within the data lens.

Append non-parsed text
Select this check box when the standard description should include the defined attributes within the Item Definition and all other unparsed text.

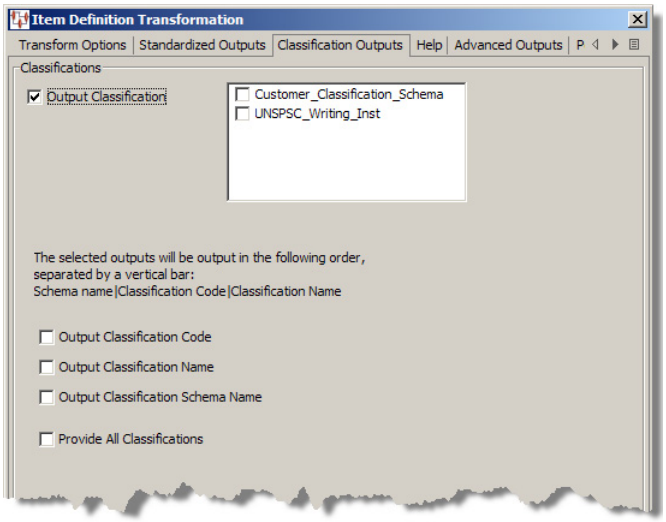
> **Note:** When selected, both of these appending options place the values at the end of the attributes defined within the Item Definitions for the specific Standardization Type selected.

Classification Outputs Tab

This tab is not active unless a Classification Type has been defined in your data lens.

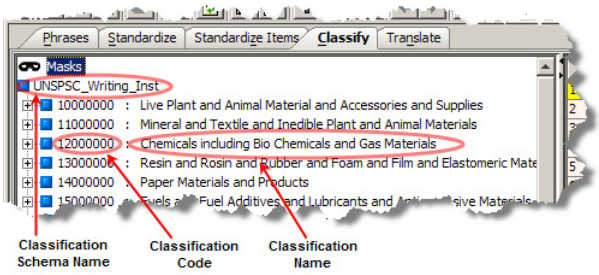
You can classify data using multiple secondary classification schemas in one Lens Transform, which simplifies your DSAs.

Use the **Classification Outputs** tab as follows.



Control	Description	Output Column Order	Output Name
Output Classification	Select this check box, and then the classification schemas you want to use to classify your data from the list. This list is populated by the Classification Types that are defined in the data lenses or Lens Group.	14	Classification

The output is in a single output field that is vertical bar (|) delimited based on your selection of the check boxes. The following figure shows the types of classification information you can output:



Output Classification Code

Select to include the code used to classify the input data.

Output Classification Name

Select to include the name of the classification code used to classify the input data.

Output Classification Schema Name

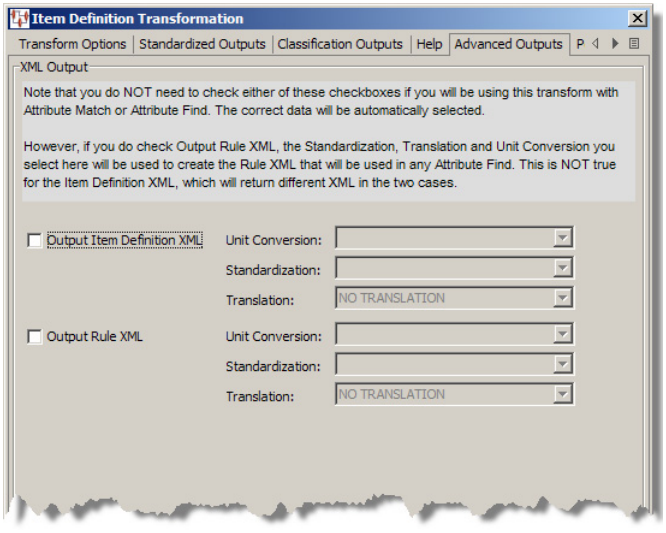
Select to include the name of the selected classification schemas used to classify the input data.

Provide All Classifications

Select this to include all of the preceding classifications.

Advanced Outputs Tab

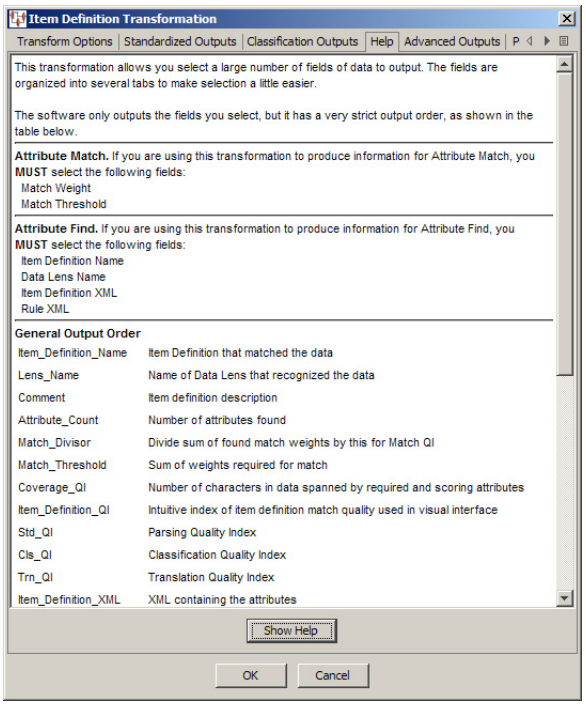
It is not necessary to select any XML outputs on this tab as this data is automatically generated. The automatic generation of the XML outputs allows you to use the attribute search functionality provided by the **Attribute Find** control.



If you want to use this tab, it is recommended that you contact Oracle Consulting Services for assistance because this functionality is applicable only in special circumstances.

Help Tab

By clicking on this tab, and then clicking the **Show Help** button, you can view the order in which fields are output.



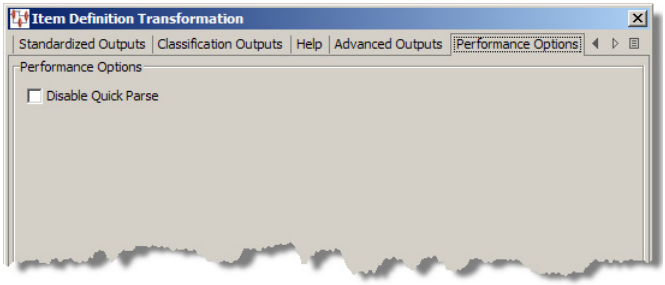
The information remains active until you close the **Item Definition Transformation** dialog. In other words, if you close the dialog and reopen it, you must click **Show Help** to activate the information for viewing.

Performance Options Tab

By default, the Quick Parse functionality is enabled. Quick Parse optimizes the parsing speed by parsing only the required attributes of all data lenses that are participating in the classification of the input data. This is applicable to both Lens Groups and when multiple data lenses are applied.

When your input data text contains no spaces or punctuation to aid parsing (conjoined), the possibility exists that some of the data may not be parsed because there are no logical text breaks so the Quick Parse function does not parse data preceding a match that could further classify the data. If your input data is conjoined, you should create an identical Lens Transform step with Quick Parse disabled that immediately follows the first to ensure that all data that was not parsed initially is parsed. You can still realize the parsing optimization even though there are two parsing steps in your DSA.

Select the **Disable Quick Parse** check box to stop the Application Studio from using Quick Parse.



Resulting Data Output Format

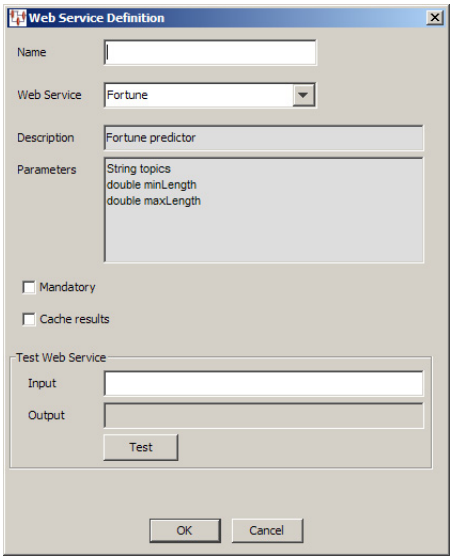
The fields of data that you have configured to be output from an Item Definition transformation when your DSA is used to process input data are provided in the following order:

Item_Definition_Name	Item Definition that matched the data
Lens_Name	Name of Data Lens that recognized the data
Comment	Item definition description
Attribute_Count	Number of attributes found
Match_Divisor	Divide sum of found match weights by this for Match QI
Match_Threshold	Sum of weights required for match
Coverage_QI	Number of characters in data spanned by required and scoring attributes
Item_Definition_QI	Intuitive index of Item Definition match quality used in visual interface
Std_QI	Parsing Quality Index
Cls_QI	Classification Quality Index
Trn_QI	Translation Quality Index
Item_Definition_XML	XML containing the attributes
Rule_XML	XML containing the parsing rules
Non_Item_Definition_Std	Standardized text (non-attribute-based)
Item_Definition_Std	Standardized text (attribute based)
Classification	Classification code or name
Att_1_Name	Name of first attribute
Att_1_Value	Attribute value, number if numeric; otherwise text
Att_1_Text	Attribute text (includes number & UOM if numeric)
Att_1_Number	Numeric value of attribute, if numeric
Att_1_UOM	Unit of measure of attribute, if numeric
Att_1_Weight	Relative importance of attribute
Att_2_Name	Name of second attribute, and so on for all subsequent attributes.
Att_2_Value	Second attribute value, and so on for all subsequent attributes.
Att_2_Text	Second attribute text, and so on for all subsequent attributes
Att_2_Number	Second attribute number, and so on for all subsequent attributes
Att_2_UOM	Second attribute unit of measure, and so on for all subsequent attributes
Att_2_Weight	Second attribute relative importance, and so on for all subsequent attributes

Defining Web Service Transforms

Web services are defined to support inoperable machine-to-machine interaction over a network.

To create a Web Service Transform, from the **Edit** menu, select **Create Web Service** or use the button of the same name.



The **Web Service Definition** dialog box appears. Use this dialog box as follows:

Name

Enter a name for your transformation. This name is displayed in the **Map Component Tree** pane and in the **Graphical Map Builder** pane.

Web Service

Select the Web service that you want to use for this transformation. This list is populated with the RPC Web Services that are configured in the Oracle DataLens Server.

Description

Populated with the description of the selected Web service from the Oracle DataLens Server.

Parameters

Populated with the parameters of the selected Web service from the Oracle DataLens Server

Mandatory

Selecting this option results in an empty return from the Web service being considered an error condition.

Cache Results

Selecting this option results in the Web service caching the results of calls to it. If the same result is required, the result is retrieved from cache and an additional call to the Web service is not made. The cache remains in effect for the duration of the DSA job run.

Test Web Service

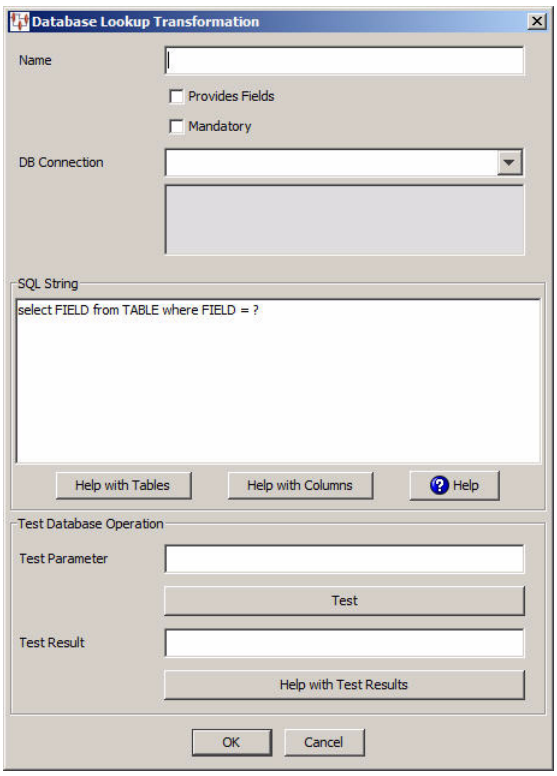
The **Test Web Service** controls are provided to help you test the Web service connection. After adding a name for the transform, you can test the data source definition by typing a data value into the **Input** field then clicking **Test**. The test input should be entered with spaces separating multiple parameters. If the parameter is embedded, spaces use double quotes around the parameter. The results are displayed in the Output field.

Defining DB Transforms

Database transforms are defined by the database used. The Transformation Map Builder supports input from existing Oracle, MySQL, PostgreSQL, or SQL Server databases. Coupled with Lens and Item Definition transforms, this capability allows multiple data sources to be appropriately aggregated.

Transformation Maps can access information stored in a database and use that information to transform data. This database information can also be merged into the transformation flow, which allows you to aggregate several separate data sources.

To create a DB Transform, from the **Edit** menu, select **Create DB Transformation** or use the button of the same name.



The **Database Lookup Transformation** dialog box appears. Use this dialog as follows.

Name field

Enter a name for your transformation. This name is displayed in the **Map Component Tree** pane and in the **Graphical Map Builder** pane.

Provides Fields

Select this check box to indicate that this database query returns multiple fields to the map. When selected, the **Mandatory** check box is not active because the fields that are returned are determined by those in the database.

Mandatory

Select this check box to define the quality of the database transformation. At run-time this transformation is success if the query returns a result, otherwise the record being processed is routed to the exception file.

DB Connection

You must select the type of database connection that you want to use. The list of database connections is populated based on those that you are configured in the Oracle DataLens Server. If the type of database connection is not listed, you must configure it in the Oracle DataLens Server so that it is available for selection when creating Transformation Maps.

SQL String

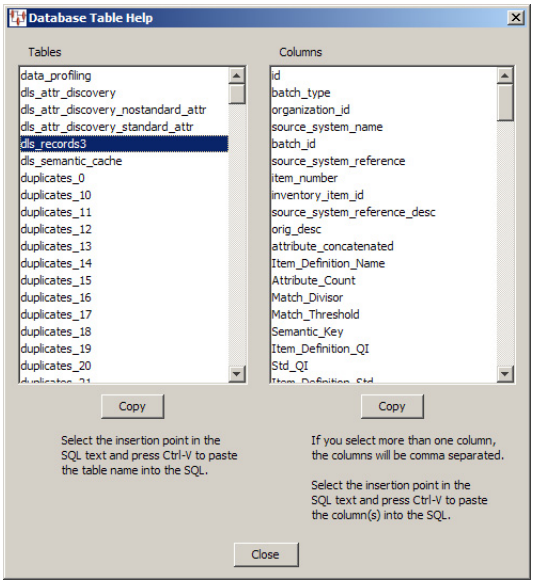
Use the **SQL String** section to construct your database query with standard SQL query statements and syntax. For example, to determine if the inventory part numbers in your input data are an exact match to a standardized set of manufacturing part numbers you could use the following SQL statements:

```
SELECT 'Exact Match Mfg Std PN'||'|'|| inventory_item_id as match_type
FROM xyz_mfg_part_numbers_all_v
WHERE upper(mfg_part_num) = upper(&?3&)
AND upper(manufacturer_name) = upper(&?2&)
AND organization_id = &?1&
AND end_date is null;
```

This `select` clause must be compatible with your database. Optionally, you can use a question mark (?) in the `select` clause. At run-time, the question mark character is replaced with the transformation input data. This allows you to create a database transformation that varies with the content of the record being processed. Additionally, when database transforms are used to aggregate data fields from several different data sources, common access key information can be used across all data sources.

Help with Tables

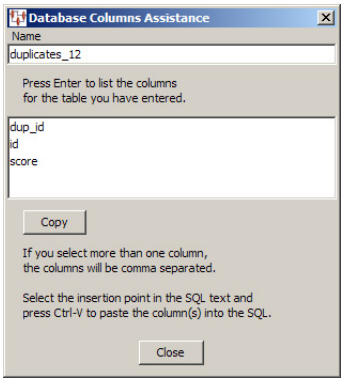
Use this button for assistance with the tables that are being used for the DB transformation.



The **Database Table Help** dialog appears. You can select a table name in the **Tables** list and view the columns for that table in the **Columns** list. You can copy information from either list to paste into the **SQL String** field. One table can be copied in the **Tables** list; one or multiple columns can be selected in the **Columns** list. Click the appropriate **Copy** button for the information you want to copy to the clipboard and click **Close**. When you are returned to the **Database Lookup Transformation** dialog box, use **Ctrl-V** to insert the copied information into the **SQL String** field at the insertion point.

Help with Columns

Use this button for assistance with the columns that are being used for the DB transformation.



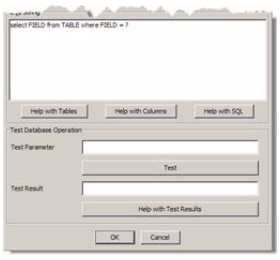
The **Database Columns Assistance** dialog appears. You can enter a table name in the **Name** field, click **OK**, and view the columns for that table. Then you can copy one or multiple columns names to the clipboard using the **Copy** button. Click **Close** to return to the **Database Lookup Transformation** dialog box and use **Ctrl-V** to insert the copied information into the **SQL String** field at the insertion point.

Help with SQL

Select this button to view brief SQL query explanations and examples.

Test Query

The Test Query fields are provided to help you test the setup of the data source connection. After you have entered a name, you can test the data source definition by typing a data value into the **Test Parameter** field and clicking the **Test Query** button. The test data value you entered is substituted for the question mark character(s) in the query string, and then the query is executed. The query results are displayed in the **Test Result** field.



When the query returns more than one record, the database transformation uses only the first record returned.

If there is an error in the connection definition, an information error message appears. Click the **Help with Test Results** button for help in interpreting error messages.

Note: The following message indicates that the database connection tested good and is not an error message:

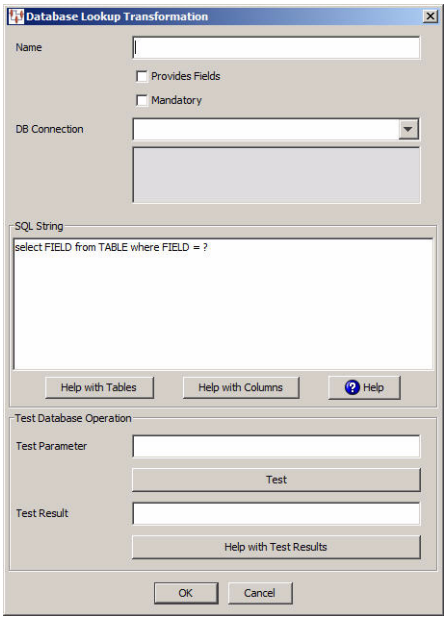
```
Please check your database now
```

If an error message is received and there are no results in the **Test Result** field, the connection is good and the query is valid, but no matching results were returned.

Example Single-Field Database Transformation

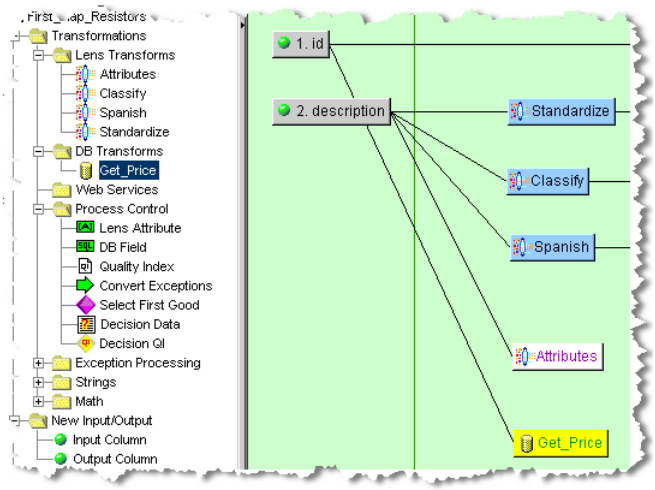
This example shows how to configure a DB Transformation whose query returns only one field. It selects the price of an item from a database based on its ID. This DB transformation example is named `Get_Price`. It uses the data source named Pricing, and a query string that selects the base price from a database using the input ID column. The input ID column is represented in the query string as a question mark (?). At run-time, Application Studio uses the ID value from the record being process to access the database per the Map definition.

1. Create a DB Transform, right-click in the **Map Component Tree** pane, select **Create DB Transformation**.



2. Click **OK**.

   This DB Transform is placed in the Map, and then connected to the input ID column by dragging it and dropping it on a node in that column.

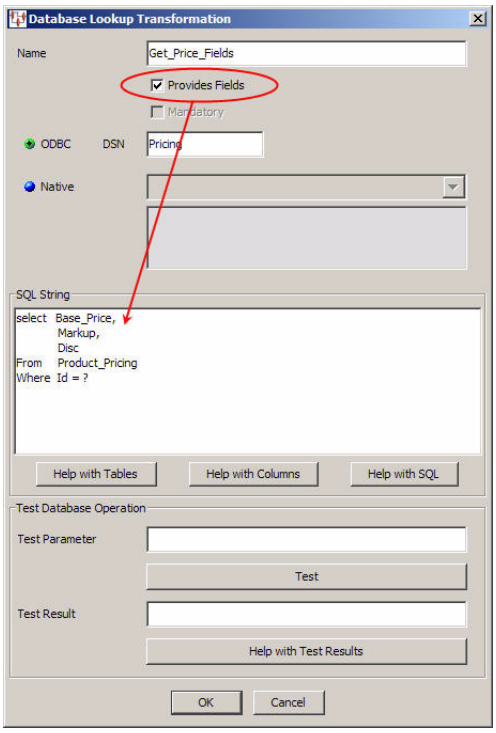


3. To connect single-field database transformations to the input column, you simply drag-and-drop the database transformation onto the target node in the Map to use the field output from the database query.

Example Multiple-Field Database Transformation

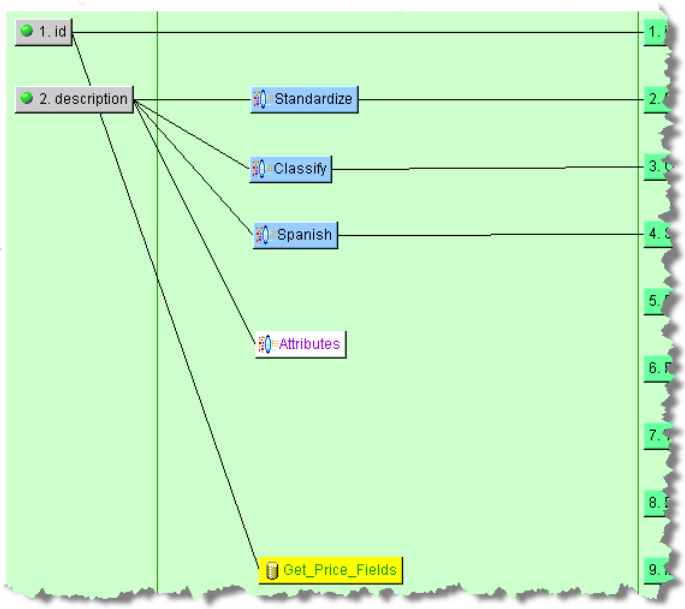
You can use several fields from a database transformation by simply selecting the **Provides Fields** check box as previously described. You must setup your query string to select the required multiple fields. This example retrieves the base price, markup, and discount from a pricing database.

1. Create a DB Transform, right-click in the **Map Component Tree** pane, select **Create DB Transformation**.
2. Select **Provides Fields**.



3. Click **OK**.

   The resulting database transformation is placed in the map and connected to the **ID** input node.

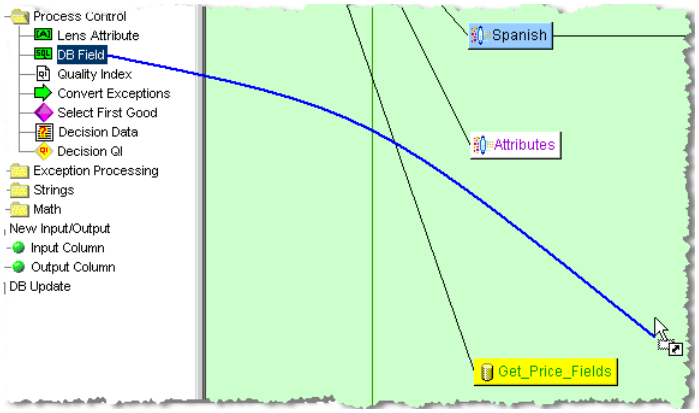


   The **DB Field** control is used to extract the query results from the database transformation.

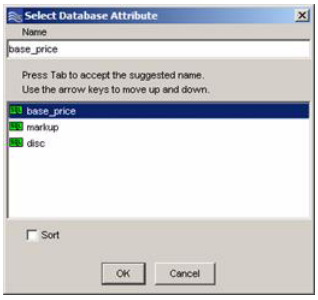
4. To add **DB Field** controls, you must first select the database transformation that will be automatically connected to it by clicking on it.

   In this case, the **Get_Price_Fields** transform is selected in the **Graphical Map Builder** pane and is colored yellow.

5. Drag and drop the DB Control from the **Map Component Tree** pane into the **Transformation** column of the **Graphical Map Builder** pane.



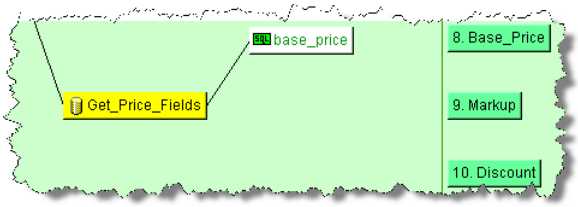
The **Select Database Attribute** dialog appears listing all of the database fields available from the define query.



Note: Because Application Studio attempts to read the list of fields directly from the defined database connection and query, an error occurs if your database connection and query are not correctly setup. If you are having trouble with your database connection, contact your IT professional.

6. Creating connections between the database fields and the data source is done by selecting the attribute in this dialog and clicking **OK**.

   The selected DB node is automatically connected to its data source. In this example, the **base_price** node is automatically connected the **Get_Price_Fields** DB node.



All DB Fields can be connected to other nodes in your map in this manner.

Defining the Output Column

The Output column is the final step in defining a Transformation Map and defines the data that will be passed to the other applications within Oracle Product Data Quality.

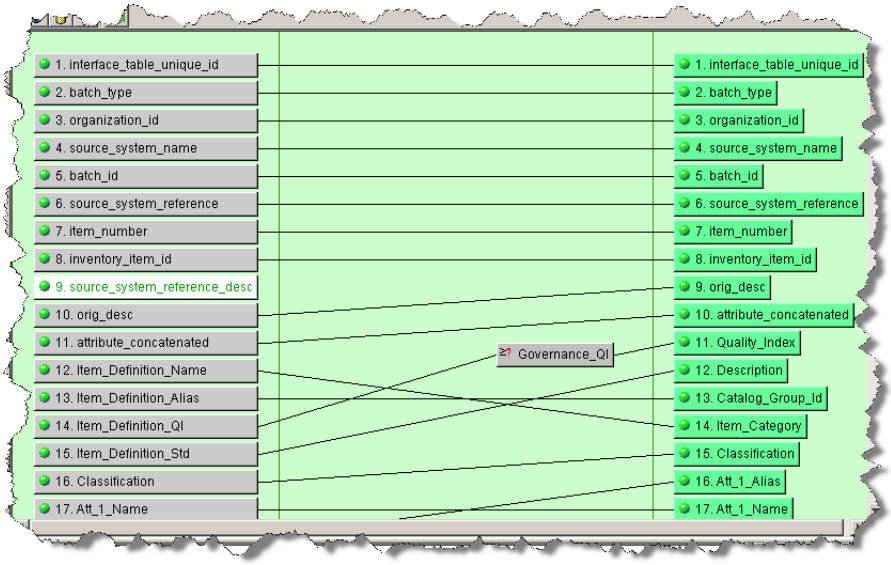
One of the ways you can create an output node is using an input node as the basis. For more information, see "Creating Output Nodes from Input Nodes" on page 4-20. These output nodes contain a single piece of data in either a text or SQL database field.

Alternatively, you can manually create output nodes by dragging the Output Column node from the **Map Component Tree** pane into the Output column of the **Graphical Map Builder** pane or pressing **Ctrl-O** anywhere in the map. The output type is selected automatically based on the type of map: text, database, or XML. For information about how use Item Definition output nodes, see "Creating Item Definition Output Nodes" on page 4-44.

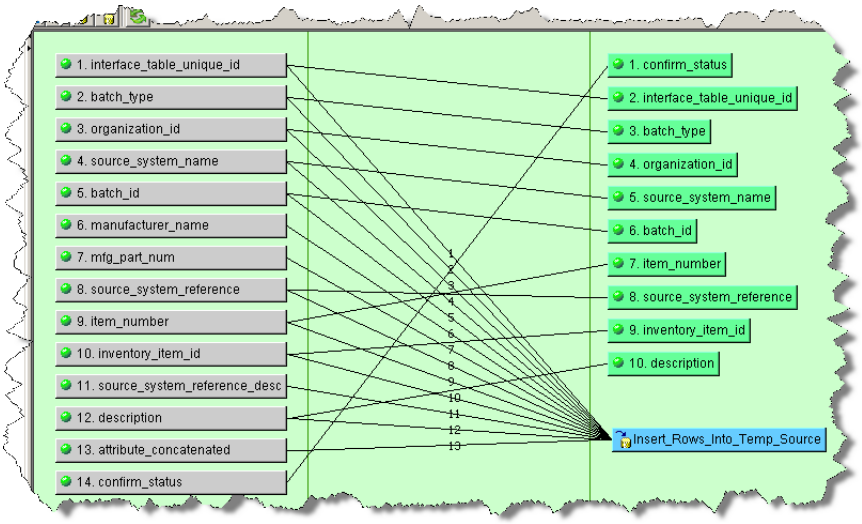
You are prompted for a name for the node and when you click **OK** the new node appears in the Output column of your map ready to be connected to an input or transformation node.

> **Note:** Similar to other naming restrictions, output column names cannot contain spaces or special characters.

Once you have created an output node, you connect it to an input or a transformation node by dragging one of these nodes and dropping it onto the output node.



When you connect two or more nodes to one node, the connection lines are sequentially numbered. This allows you to discern the connections between nodes easily, which is particularly useful when there are numerous nodes in your map.

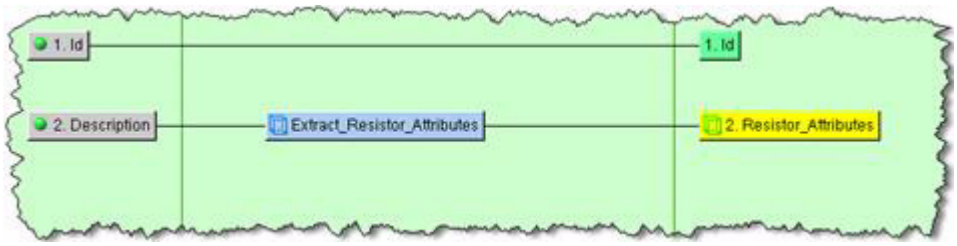


Creating Item Definition Output Nodes

To create an Item Definition Output, expand the **New Input/Output** folder in the **Map Component Tree** pane, then drag and drop **Item Definition Output** node onto the Transformation Map in the Output column of the **Graphical Map Builder** pane.

Enter a name for the output node, select (or enter) the number of attributes that this output node will create, and then click **OK**. You should enter a value in this field that exceeds the maximum number of attributes defined within the Item Definition. If only the first attribute should be created, enter a 1 as the attribute count. It is not necessary for Output node names to match Input node names.

Once connected, the Item Definition Output node might look like the following:

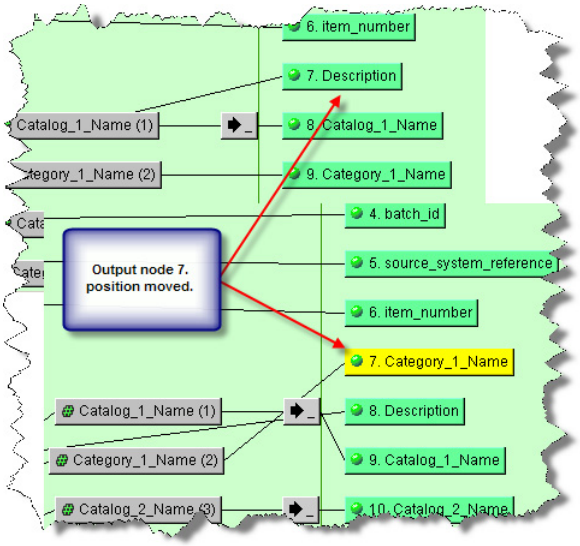


Modifying Transformation Map Column Nodes

The nodes that you have added to the transformation map columns can be modified using the functions described in this section.

Moving Column Nodes

Column nodes are placed in the map in the order in which they are created (automatically or manually) and can be moved by dragging and dropping a node to a different position in the column. This does not change the connections to other nodes; these connections remain intact and the connection lines are moved accordingly.

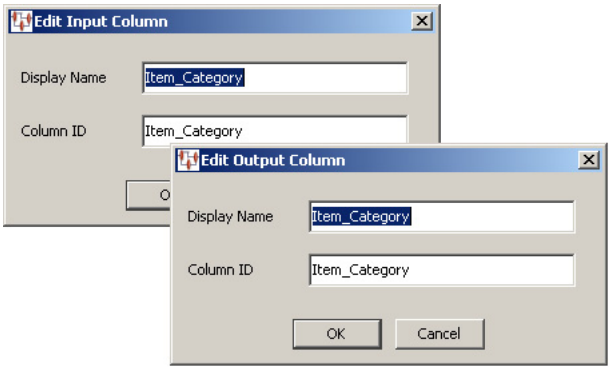


Though the example depicts moving output nodes, you can move any of the nodes in any column. The position number of the effected nodes changes to reflect the repositioning of a moved node.

Editing Column Nodes

You can edit the name and the field position number of existing input or output column nodes.

You can edit a node in any column, right-click on it, and then select **Edit Column**.



You can change the name that is displayed in the column using the **Display** field. The position label number can be changed using the **Name** field. Click **OK** to effect the changes and update the map column.

Disconnect Incoming

The connections to input and transformation nodes can be removed so that you can connect a transformation or Output node to another node.

You can disconnect a node from its incoming nodes, right-click on the node, and then select **Disconnect Incoming**. The lines connecting the selected node to any incoming nodes disappear, as well as the connections to the incoming data.

Deleting Nodes

You can select a node or nodes and delete them if they are not required on the Transformation Map.

Delete one node by right-clicking on it, and then select **Delete Node(s)**.

To select more than one input node, use the **Shift** (selecting continuously) or the **Ctrl** (selecting discontinuously) keys while selecting the nodes to be deleted. When all nodes are selected, right-click on one of the selected nodes and select **Delete Node(s)**. The nodes are removed from the map.

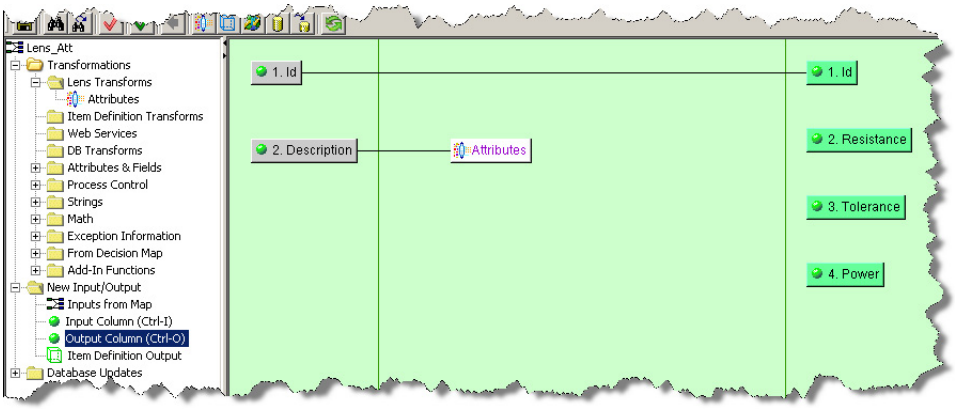
Transforming Data

This section details various examples of how you can process your data using some of the numerous transformation operations available.

Transforming Data Using Attributes and Fields

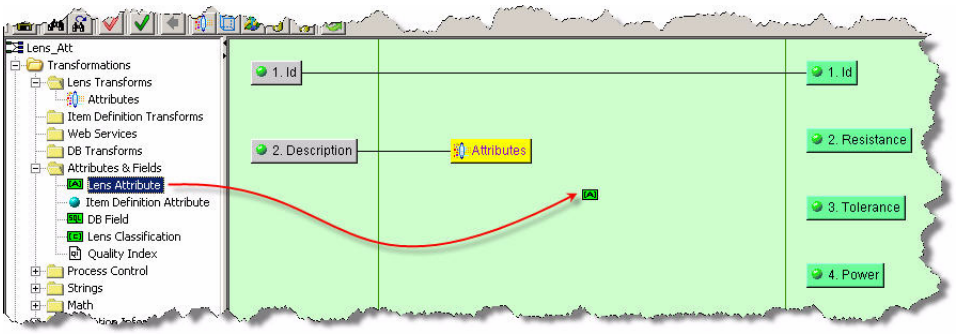
The various attribute and field widgets are described in Attributes& Fields Widgets. These widgets are directly connected to a particular Lens Transform to process the data.

In this example, a Lens Transform named **Attributes** has been defined with the **Provides Attributes** option set to populate the Resistance, Power, and Tolerance output column nodes. Data lens attributes are directly associated with Lens Transforms, unlike DB Fields that are associated with a defined database transformation.

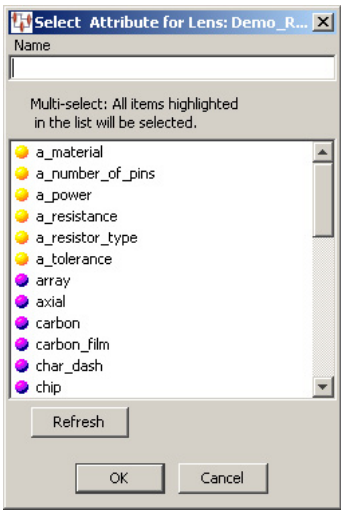


The next step in extracting the attribute text into the correct output columns is to add data lens attribute controls to the map. When adding data lens attribute controls, you must first select the data lens transformation associated with the data lens attribute control.

To make this association, click the **Attributes** transform in the **Graphical Map Builder** pane to select it; it changes color to yellow. Next, from the **Map Component Tree** pane, drag-and-drop the **Lens Attribute** node into the transformation column of the **Graphical Map Builder** pane.



The **Select Attribute for Lens** dialog box appears. This dialog lists all of the phrases and terms found in the data lens used by the selected Lens Transform. In this case, the **Attributes** transformation attributes are listed.

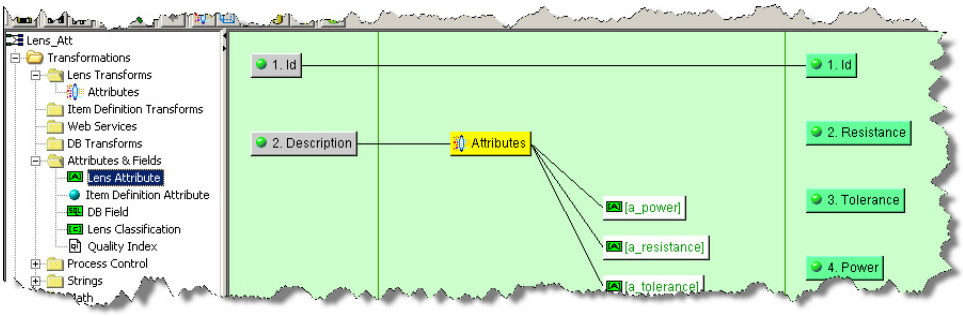


As you enter text in the **Name** field, the Application Studio automatically selects the closest phrase or term match from the list and takes you to that item. You can use the **Tab** key to complete the phrase or term name you have begun in the Name field.

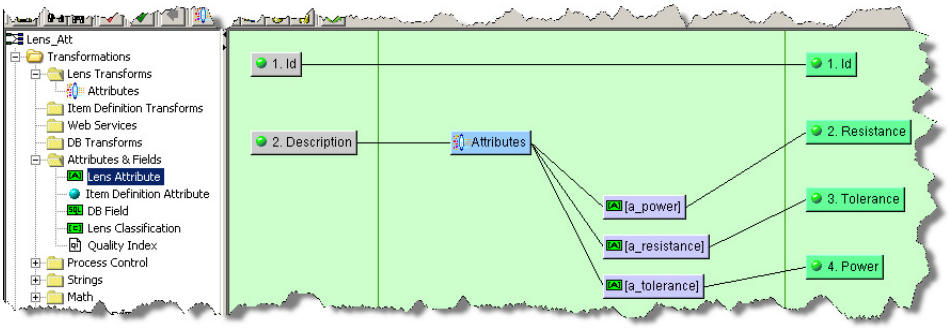
You select one or more items from the list to add attributes to your map; selecting multiple items creates a **Lens Attribute** node for each.

> **Tip:** If the data lens has been updated and selected into the DataLens Administrator since you started your Transformation Map Builder session, you may need to use the Refresh button on this dialog to update your list of phrases and terms supplied by the data lens.

Click **OK** to add the selected phrase or term rules to your map as a data lens attributes and connect them to the selected lens transformation.



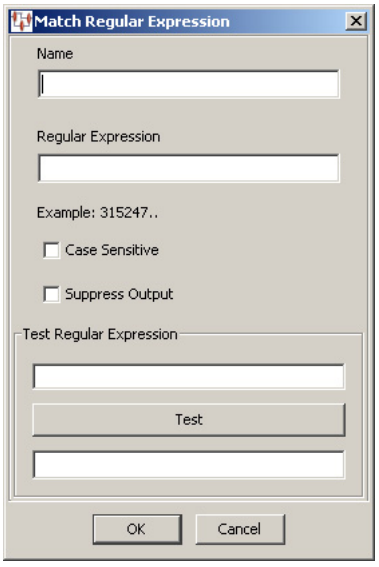
The final step is to connect these attributes to their respective output columns as previously described, and then to save your map.



Transforming Data Using Processing Controls

The various processing control widgets are described in Process Control Widgets. One of the most frequently used processing operations is to match data. For example, you may have data that you want to confirm whether it was found to be a match for a phrase rule. To do this you would use a match processing operation, which uses regular expressions to determine matching data.

From the **Map Component Tree** pane, drag and drop a **Match** widget into the transformation column of the **Graphical Map Builder** pane.

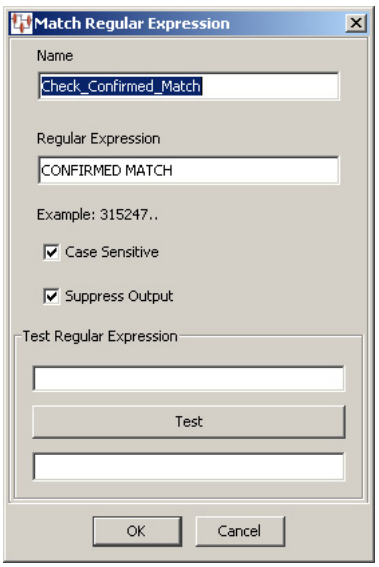


The **Match Regular Expression** dialog box appears. You provide the widget name, regular expression that you want to use for matching. Additionally, you can set the

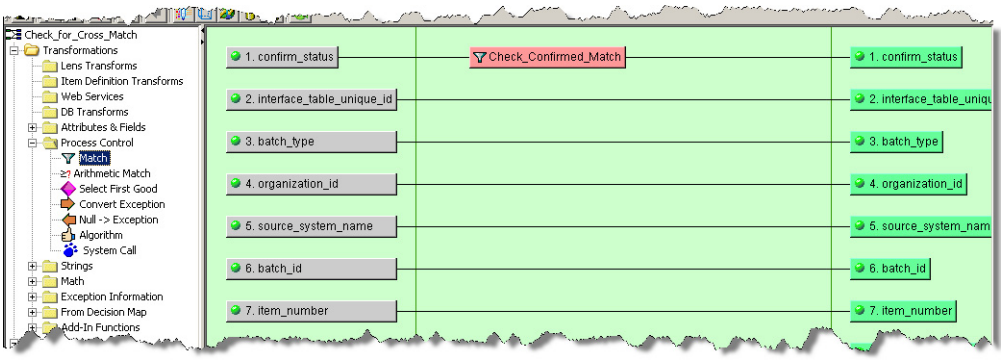
matching operation to be case sensitive or suppress any output results from being stored using the check boxes.

After you have entered defined the options, you can test the regular expression by entering a data value into the **Test Regular Expression** field and clicking **Test**. The test results are displayed in the **Test Result** field.

In the following example, the `Check_Confirmed_Match` **Match** widget is being created that identifies the data as a match if the uppercase expression `CONFIRMED MATCH` is encountered. It is set not to retain output data because the input data is processed by a subsequent DSA step.



Once the **Match** widget is connected to the input and output nodes, it looks similar to the following:



Transforming Data Using String Operations

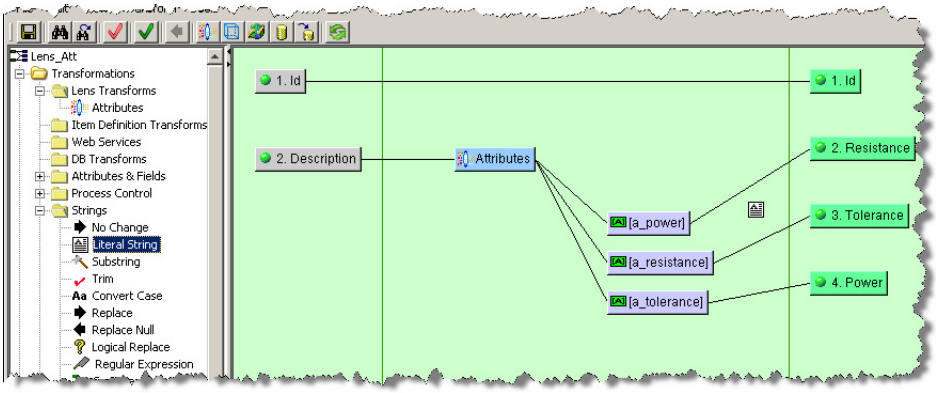
The various string operators are described in Strings Widgets on page 76. This section provides some examples of how you might use string operators in your Transformation Maps.

Literal String Operation

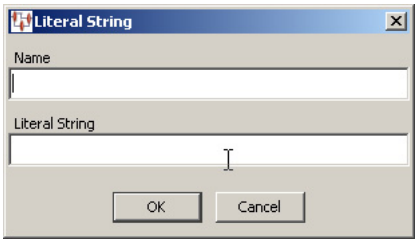
Literal Strings are used to add static text to your map transformation process. For example, if you want to add a label field for an attribute that was extracted by a Lens

Transform, as in the Transforming Data Using Attributes and Fields example, you would use a **Literal String** widget and a corresponding **Output** node.

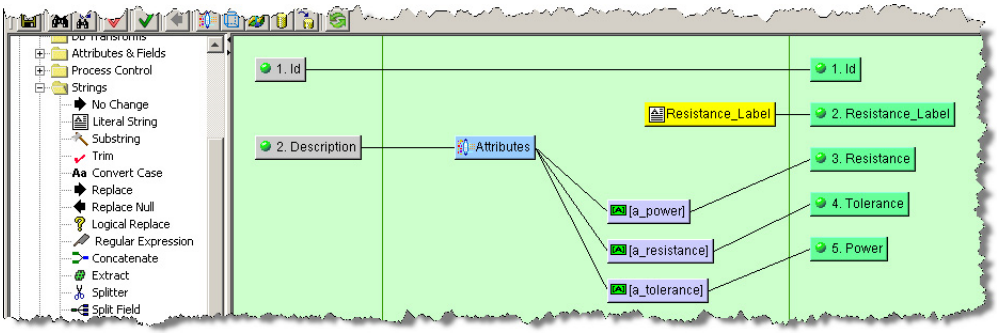
From the **Map Component Tree** pane, drag and drop a **Literal String** widget into the transformation column of the **Graphical Map Builder** pane.



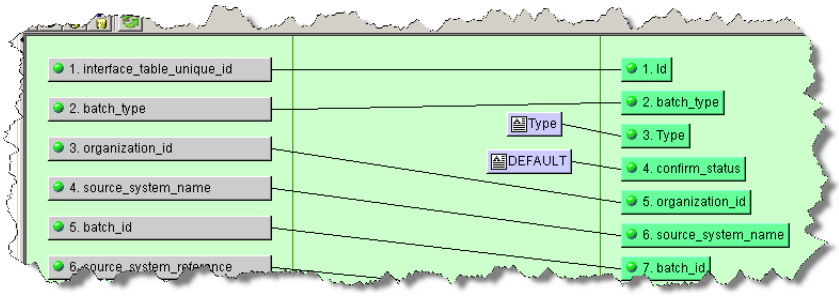
The **Literal String** dialog box appears.



You are prompted for a **Name** and the static text that you want used. Next, you add an **Output** node to the Output column of the map and connect the two. In the following example, a label was created for the **Resistance** attribute column:



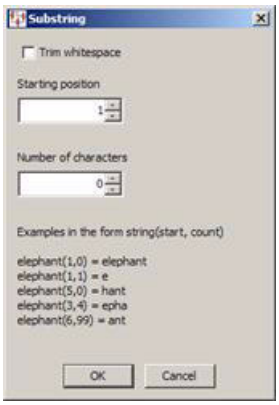
Another way to use the **Literal String** widget is simply to assign a specific string of static text to an Output node. The process is the same and may look similar to the following:



Substring Operation

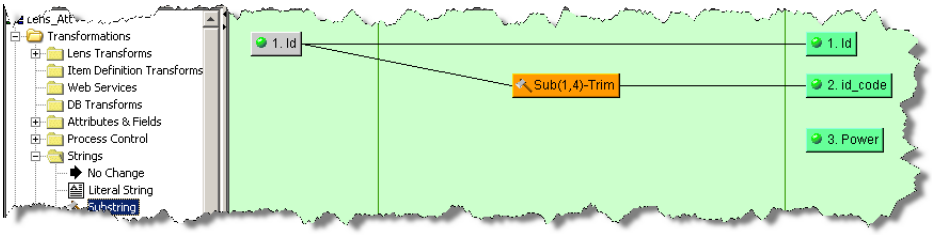
Substrings are used to extract a specific sub-set of characters from the input string. For example, if the first four characters of your ID column contain an important code that needs to be routed to a separate output column, you would use the substring operation to accomplish this.

From the **Map Component Tree** pane, drag and drop a **Substring** widget into the transformation column of the **Graphical Map Builder** pane.



The **Substring** dialog box appears. Select the start position for the substring and number of characters you want to be in it. You can also trim any whitespace using the check box. Click **OK**, and then connect the Substring widgets to both the input and output nodes.

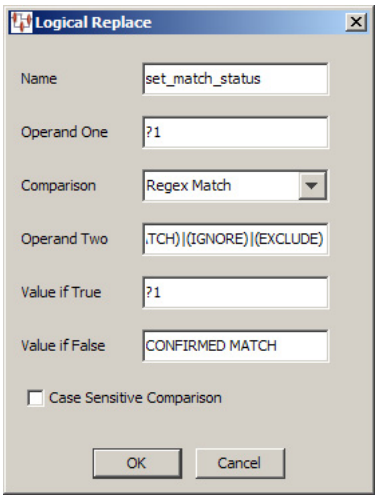
In the following example, the substring that will be created will start with the first character, contain four characters, and will not contain any whitespace if it is present within the four characters:



Logical Replace Input Operation

You can use this widget to replace any text using a logical expression. It is a powerful tool, particularly when using regular expressions.

From the **Map Component Tree** pane, drag and drop a **Logical Replace** widget into the transformation column of the **Graphical Map Builder** pane.



The logical replace compares value of **Operand One** against the value of **Operand Two** using the selected comparison. If the result is true, the value entered into the **Value if True** field is use as the output of this widget. Conversely, if the result is false the **Value if False** value is the output. These values can:

- be literal (string or number) so are replaced with the entered values, or
- indicate an input parameter in the form of `?1, ?2, ?3` and so on. This syntax forces the replace to use the first, second, and third inputs to the transformation during the comparison and generating the output. The `?number` syntax can be used in any of the operand or value fields.

The comparison is a numeric comparison if both operands are numeric, or a string comparison if either operands are non-numeric. If a **Regex Match** comparison is selected, **Operand Two** is expected to be a valid regular expression, or the second run-time input must be a regular expression.

> **Tip:** Tooltips are available when hovering over one of the operand or value field names.

In this example, **Operand One** indicates that the input data is going to be compared against the value of **Operand Two** using a regular expression. Because this is a regular expression comparison, the following delimited list was entered to see if the input data matched anything in the list:

```
(CONFIRMED CROSS REFERENCE)|(CONFIRMED NEW)|(CONFIRMED
MATCH)|(IGNORE)|(EXCLUDE)
```

If the value is false, it is replaced with `CONFIRMED MATCH`; otherwise, the value remains unchanged so that it can be processed with perhaps a subsequent transformation column node.

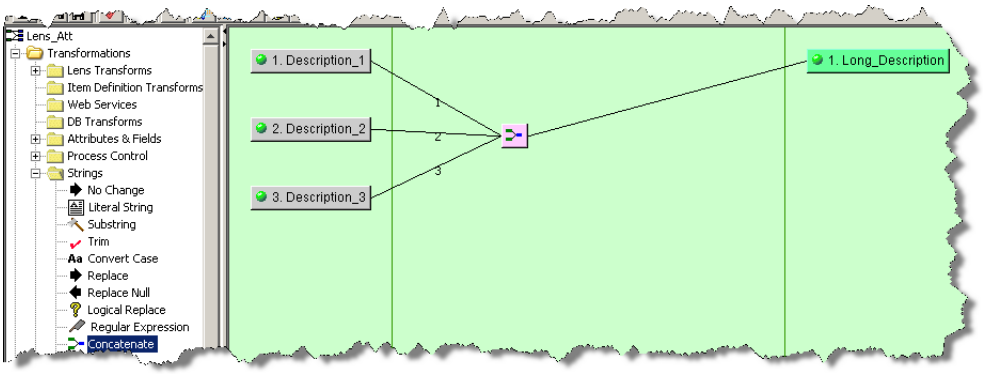
For more information, see "Regular Expressions" on page B-1.

Concatenate Inputs Operation

The **Concatenate** widget is used to merge multiple input strings to create one output string. The connected string inputs are merged top to bottom with a space between each string.

From the **Map Component Tree** pane, drag and drop a **Concatenate** widget into the transformation column of the **Graphical Map Builder** pane.

Connect the **Concatenate** widget to one or more input nodes and just one output node to create a concatenated string as in the following example:

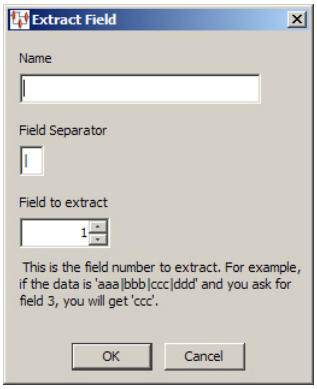


Extract, Splitter and Split Field Inputs Operation

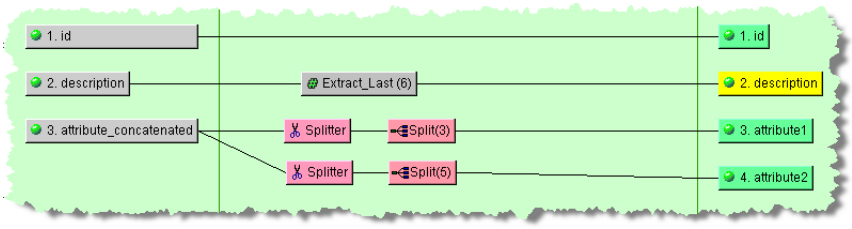
The **Extract** widget can be used to retrieve a specific field to pass as output data.

From the **Map Component Tree** pane, drag and drop an **Extract** widget into the transformation column of the **Graphical Map Builder** pane.

The **Extract Field** dialog box appears.



Enter a name for the widget, the symbol used to separate the fields, and the number of the field that you want extracted. Click **OK** to complete and add the Extract widget to the map, and then connect it to an output node to complete its use.



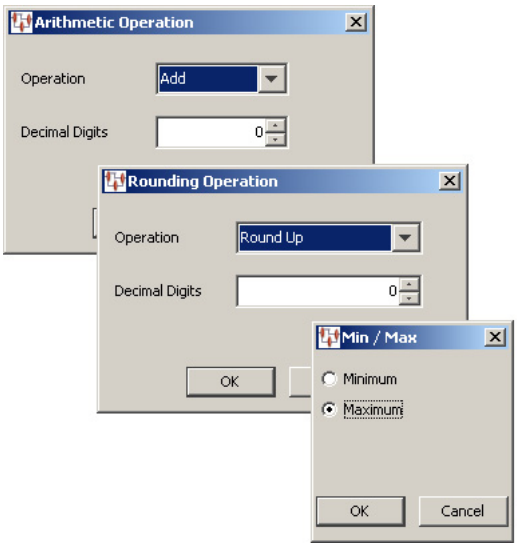
As in the preceding example, you can use the **Splitter** and **Split Field** widgets in conjunction to split the input data into chunks for multiple output nodes. They must be used in pairs for the output to operate correctly. You add the **Splitter** widget first then the **Split** widget, which allows you to select the number of characters of data that will be retrieved. These nodes are connected together, and to the input node. Each of the Split widgets is connected to the appropriate output nodes.

Note: The count selected for each of the Extract and Split widgets, the number of fields, and number of characters respectively, appear next to the widget name in parenthesis for quick identification.

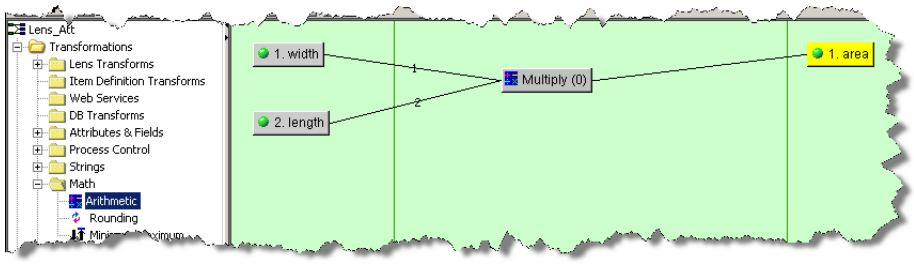
Transforming Data Using Math Operations

The various string operators are described in Math Widgets on page 79. All of the math operators function in a similar manner by transforming the input numbers to a different output number using a designated math operator.

From the **Map Component Tree** pane, drag and drop a Math widget into the transformation column of the **Graphical Map Builder** pane.



The selected math operation dialog box appears. Select the arithmetic operation you want to use and the number of decimal digits that the output will be calculated to, or select minimum or maximum. Click **OK** to add the **Math** widget to the map, and then connect the input and output nodes to it as in the following example:



Testing and Validating Transformation Maps

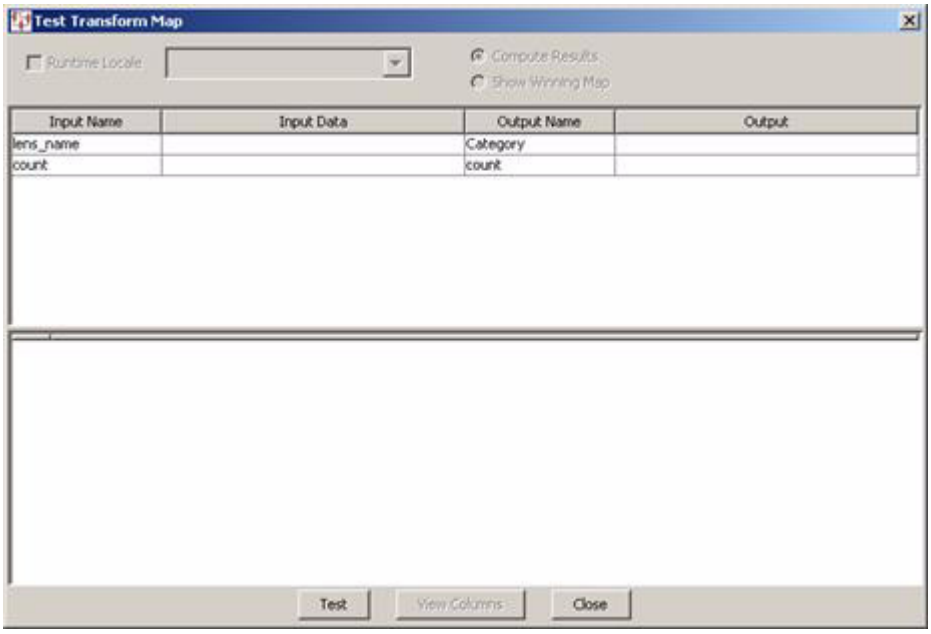
During the design of a map and upon completion, you should test and validate it to ensure that it will operate properly. This section describes these two functions.

Testing a Map

To test the open map, select **Test Map** from the **Transform** menu or click the button of the same name on the toolbar.

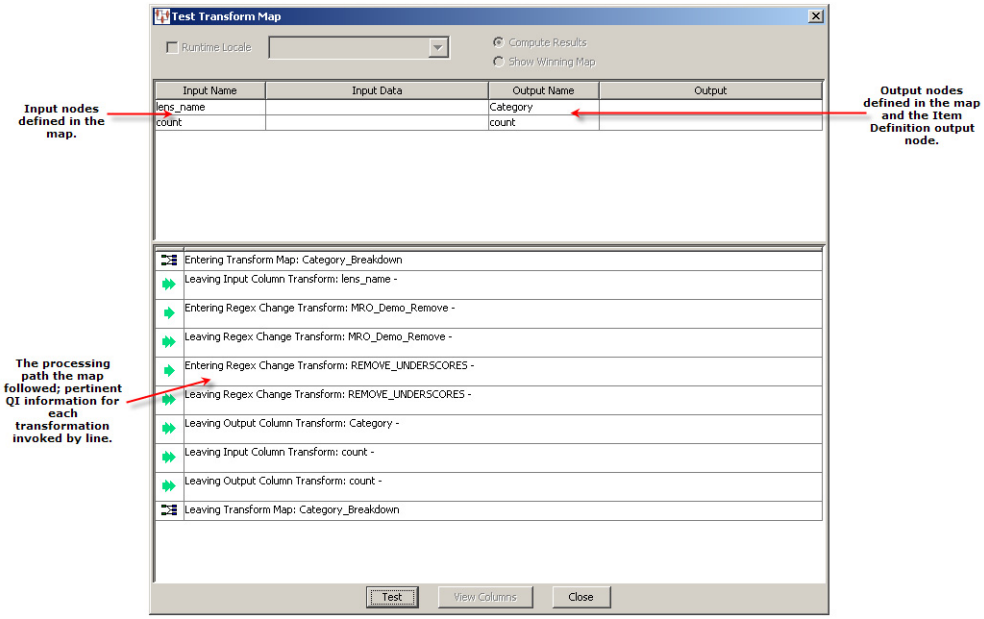
Note: If you have unsaved changes or errors are found when the save is attempted, you are alerted and can choose to correct these errors by clicking **No** or **Cancel**.

Note: Maps that contain decision Item Definitions cannot be tested.



This **Test Transformation Map** dialog box appears.

Click **Test** to review your testing results.



You can enter data into any of the **Input Data** fields then click the **Test** button to view the results in the corresponding **Output** field. This can help you verify the operation of the map and isolate any errors that may exist. Click **Close** to conclude testing.

Validating a Map

Validating a Transformation Map ensures that the map steps and all columns connect properly; this is internal to the map. Any errors that require correction are identified with a brief explanation in an informational pop-up message.

To validate the open map, select **Validate Map** from the **Transform** menu or click the button of the same name on the toolbar. A message is displayed only if there are errors to be corrected in the map.

Importing and Exporting Maps

Transformation Maps that are defined in a DSA can be exported for use in other DSAs. This enables you to reuse a map repeatedly and avoids recreating the same function across various DSAs. For example, you could create one Transformation Map that queries a database for a given set of attributes and reuse it across a set of DSAs that are based on the same input data.

You can export a Transformation Map by selecting it in the **DSA Component Tree** pane and right-clicking on it, and then selecting **Export Transform Map**. The map is automatically saved to your *DataLens*\data\map directory and the file is named the same as the map.

> **Tip:** The Status Field shows you the full pathname of the Transform Map file that was saved.

The **Import Referenced Transformation Map** option tells Application Studio to check its export directory for a transform of the same name. If one is found, that transform is imported to the current map. This is a very handy technique for copying Transformation Maps from one DSA to another: open the source DSA, export the desired transform, then open the destination map and select **Import Referenced Transformation Maps**.

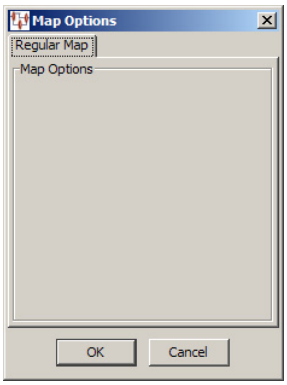
Map Options

The initial options that you set when defining a new Transformation Map may be changed when necessary as described in this section. Though the map options vary for the different Transformation Map types, they are accessed in the same manner.

From the **Tools** menu, select **Map Options**.

Text Input Map Options

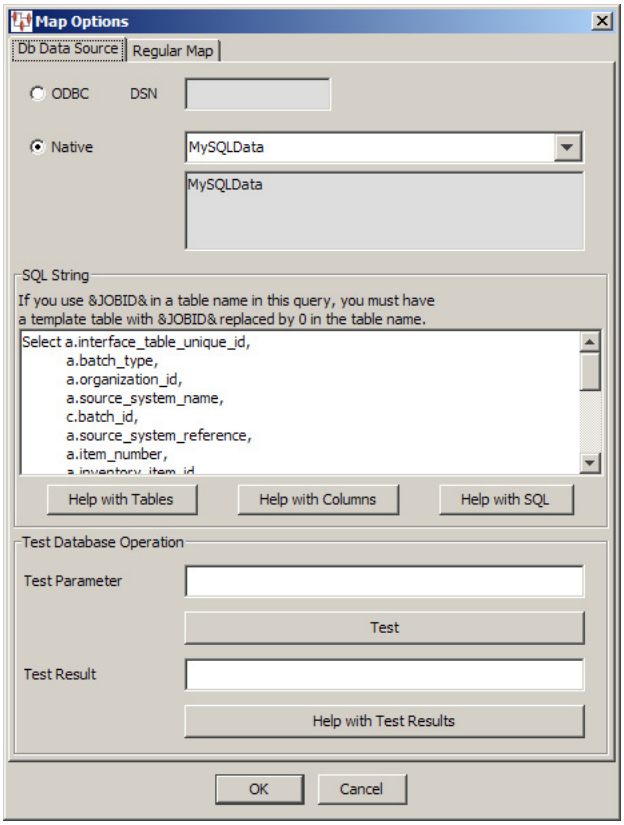
The text input (tab-separated files) Transformation Maps do not have configurable options.



Database Map Options

You can modify the original database query that you configured when creating your database map.

From the **Tools** menu, select **Map Options**.

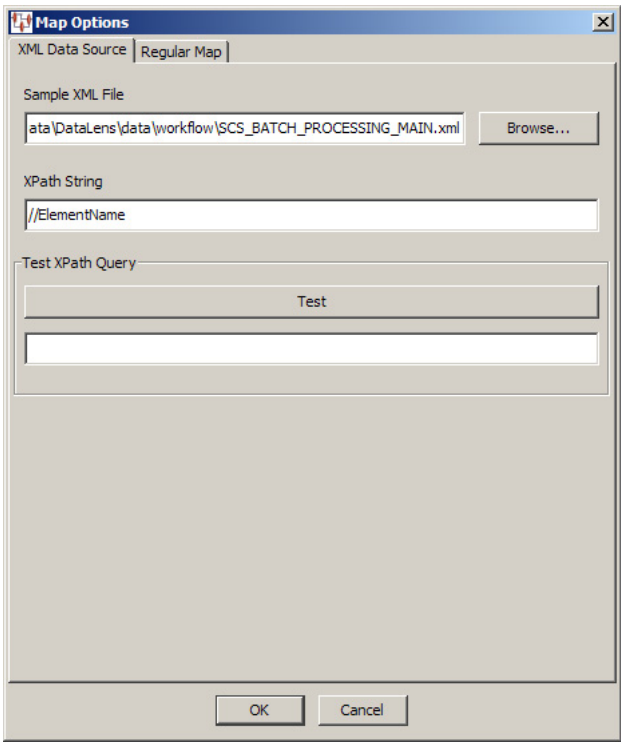


The **Map Options** dialog box is used as described in "Database Query Data Input" on page 4-7. The **Regular Map** tab does not contain any options.

XML Map Options

You can modify the original database query that you configured when creating your database map.

From the **Tools** menu, select **Map Options**.



The **Map Options** dialog box is used as described in "XML Document Data Input" on page 4-9. The **Regular Map** tab does not contain any options.

5

Decision Map Builder

A Decision Map is specialized Transformation Map that has a different capability than Transformation Maps. Unlike Transformation Maps whose primary function is to define the actual transformation process at the record level, Decision Maps function at a higher level of abstraction. Decision Maps define the sequential operation among different data lenses and other data sources (databases and Web services).

Decision Maps enable you to define a business process based on the following operations:

- Identify the data sources
- Create a model for processing record level data at run-time including the mainstream processing and exception handling.
- Identify alternate processing schemes including the use of other data lenses and data sources, based on quality metrics.
- Provides seamless hand-off to Transformation Maps.

The primary advantage of Decision Maps is that they enable the Oracle DataLens Server to encapsulate a semantic understanding of corporate processes.

> **Note:** It is recommended that you create Decision Maps first as part of a corporate methodology for semantic mapping and transformations.

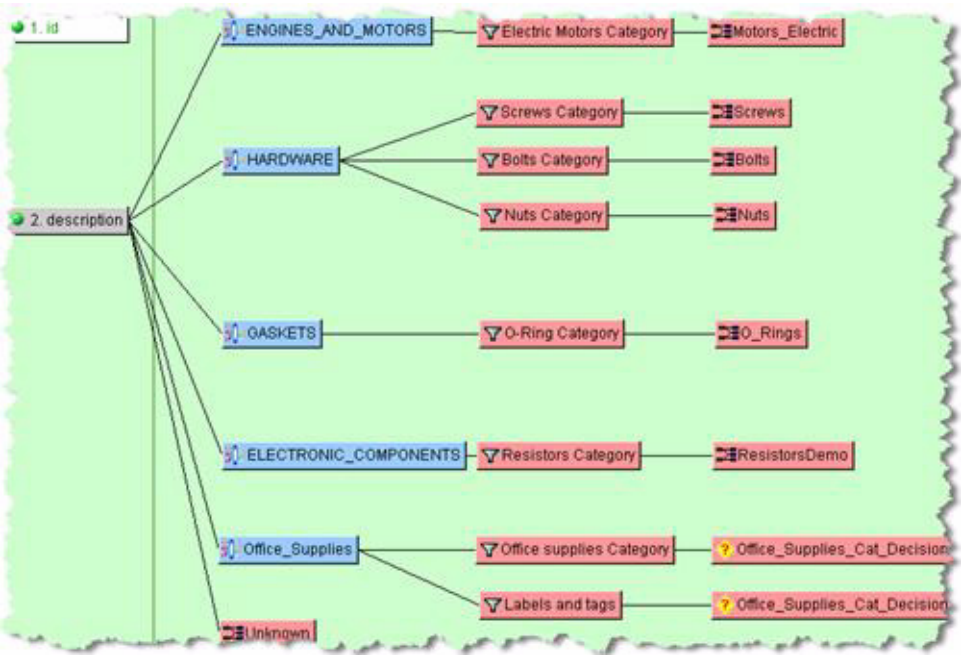
Decision Map Example

For example, the following sample data input spreadsheet excerpt involves a mixture of content:

	A	B
1	ID	Description
2	1	MOTOR , ELEC , 1 - 1/2 HP , 182T FR , 1200 RPM , 230
3	2	O-RING , BODY , BUNA-N , GOES ON CYLINDER BODY
4	3	O-RING , 3/4"ID , 15/16"OD , 3/32" CS , NITRILE-
5	4	MOTOR , ELEC , 1 HP , 143T FR , 1725 RPM , 460 VAC
6	5	MOTOR , ELEC , 1 HP , 143T FR , 1750 RPM , 230/460
7	6	Sheet Metal Screw Slotted Flat Head Zinc Plated #6 x 3/8
8	7	MOTOR , ELEC , 3 HP , 182T FR , 1730 RPM , 230/460
9	8	MOTOR , ELEC , 7 - 1/2 HP , 254TC FR , 1175 RPM ,
10	9	RESP,WW AX , 50 OHM , 5 W , 10% , .88"L
11	10	RESS , TF CH 1206 , 825 OHM , 1 / 8 W , 10%
12	11	Sheet Metal Screw Slotted Flat Head Zinc Plated #10 x 1 1/
13	12	RESS , 1206 TF CH , 0.0 OHM , 1 / 8 W , 5%
14	13	O-RING, 1 - 9/16 "ID , 1 - 3/4 "OD , 3/32" CS , NIT-
15	14	O-RING , PKG , VITON
16	15	RESP , NTWK SIP ISO , 47 OHM , 3 R , 2%, 1W
17	16	MOTOR , ELEC , 5 HP , 184T FR , 1745 RPM , 230/460
18	17	MOTOR , ELEC , 40 HP , 324TS FR , 3560 RPM , 460
19	18	RESP , NTWK SIP 8 P 33 OHM 4 R 2% 1W
20	19	MOTOR , ELEC , 7 - 1/2 HP , 210LP FR , 3600 RPM ,
21	20	Hex Head Cap Screw M27 x 240 Partial Thread Zinc
22	21	RES 0603 1% 169 OHM 3W
23	22	Hex Head Cap Screw M20 x 150 Full Thread Zinc
24	23	MOTOR , ELEC , 2 HP , 145T FR , 1730 RPM , 230/
25	24	MOTOR , ELEC , 7 - 1/2 HP , 213LP FR , 3600 RPM ,
26	25	Hex Head Cap Screw M5-0.8 x 70 Partial Thread Plain
27	26	MOTOR , ELEC , 40 HP , 364T FR , 1175 RPM , 460
28	27	MOTOR , ELEC , 10 HP , 215TCV FR , 3525 RPM , 230/
29	28	MOTOR , ELEC , 3 HP , 182TCV FR , 3515 RPM , 230/
30		OR , ELEC , 15 HP , 254HP 15 RPM

Several product commodity types in the content must be processed to achieve the corporate goal. Each commodity type has its own associated data lens. It is necessary to know which data lens to apply to which line item in order to apply the correct solution. The sample input data contains a random collection of Fasteners, Gaskets, Resistors, and Electrical Motors.

The following Decision Map selects among alternative processing schemes using Transformation Maps that are based on the quality metrics reported by each decision.:



Clearly, Hardware data (fasteners) are better recognized by a Hardware data lens than an Electric Motor data lens. The quality metrics represent the semantic overlap

between the record data and the data lenses. The data lens with the most semantic overlap has the highest quality metric and hence understands the correct routing for the individual record. Ultimately, the chosen Transformation Map is used to classify the record, to standardize the description, and to extract up to four key attributes from the description.

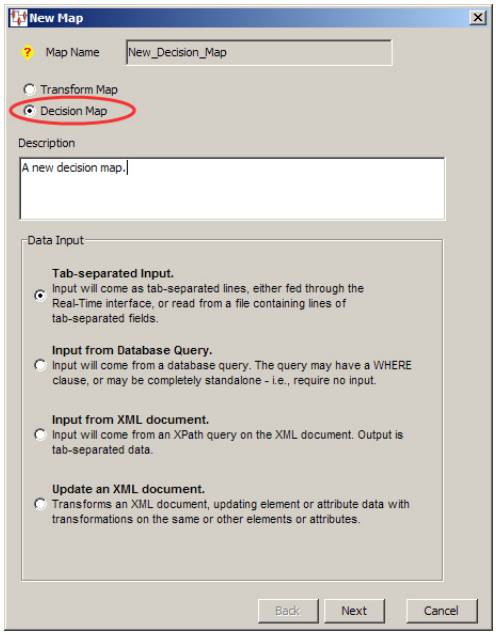
> **Note:** The client workspace is described in Understanding the Transformation Map Client Workspace on page 60.

Creating Decision Maps

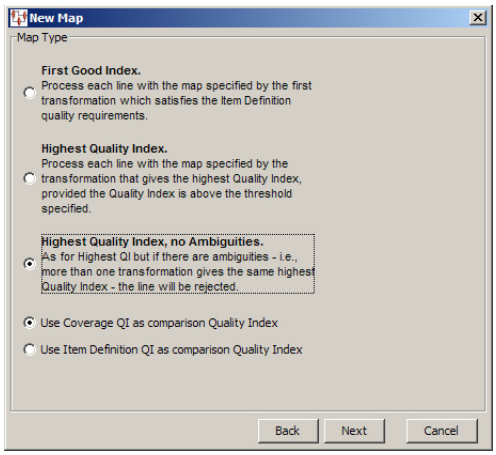
When creating a Decision Map, remember the following:

- Each decision node must have a corresponding output.
- All data that you want to use in downstream processing must be passed through the map even if it is not used in the decision.
- Item Definition outputs

The steps in creating a new Decision Map are similar to that of creating a Transformation Map as described in "Creating Transformation Maps" on page 4-5 though the Decision Map option is selected.



After selecting the type of Decision Map you want to create, you can decide the map's operation.

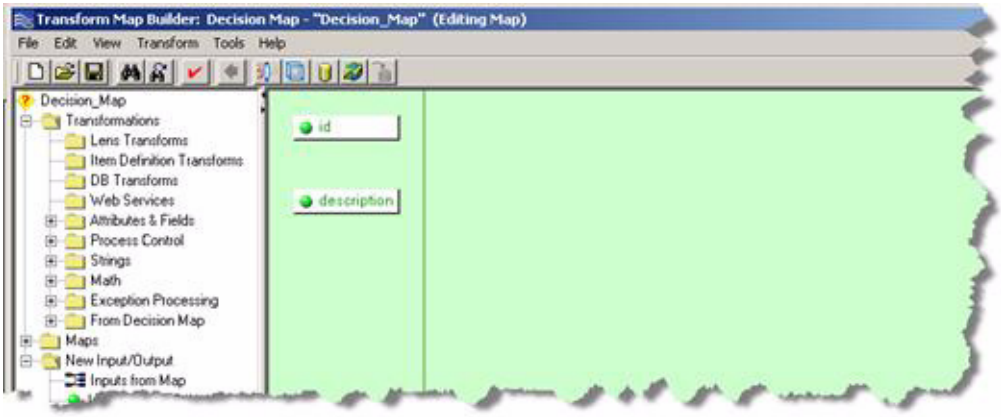


You can route the input record based on the transformation with the highest quality index (best semantic overlap) by selecting **First Good Index**.

Alternatively, you can select one of the highest unique quality index options: **Highest Quality Index** or **Highest Quality, no Ambiguities**. With these options, you set the threshold quality index below which the record is rejected as an exception record or above which the record is routed to the appropriate processing respectively.

Typically, the default **Use Coverage QI as comparison Quality Index** option should be used for all decision maps. It sets the span number to be the tie breaker in determining the highest quality index. However, if the decision map is based on an Item Definition transform, then the **Use Item Definition QI as comparison Quality Index** should be used.

After you complete the Decision Map creation, you add input column definitions as you would in a Transformation Map though the output column node is an additional Transformation Map that processes and defines the set of outputs.



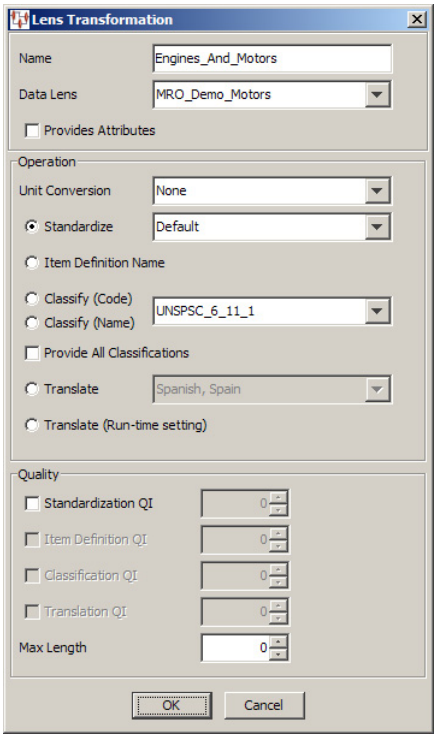
There are two semantic overlap approaches available for you to use to make your processing decisions, Lens Transformations or Item Definition Transformations. Both of these approaches are described in the following sections.

Creating Decision Maps Using Lens Transformations

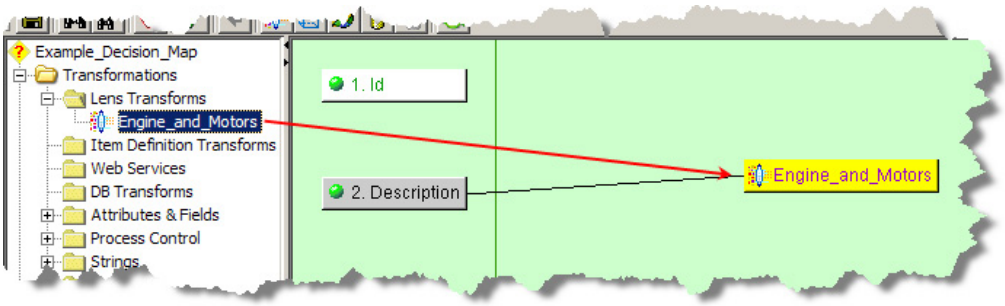
The example Decision Map created in this section to illustrate the use of Lens Transformations in Decision Maps use a data lens that is based on engine and motor data that recognizes records about electric motors (see "Decision Map Example" on

page 5-1). The same classify operation is applied in each of the Lens Transformations. The Lens Transformation that best classifies the record will then determine the Transformation Map to apply to that record. In this example, the `Engines_and_Motors` Lens Transformation should correctly classify any electric motor records, and then route the record to the `Motors_Electric` Transformation Map for processing.

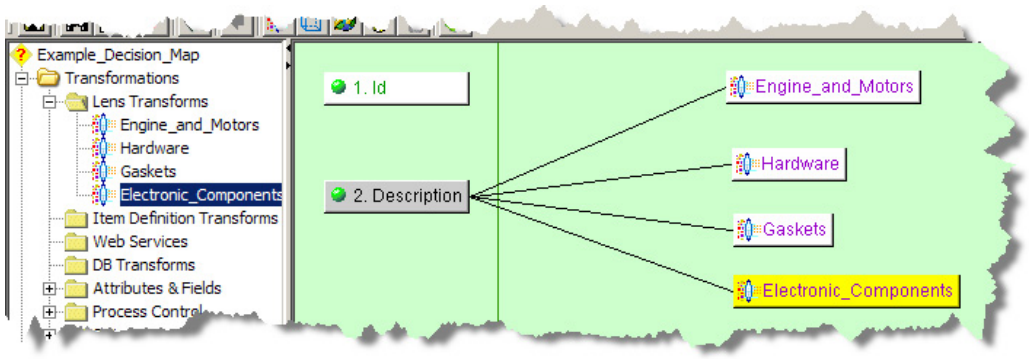
To create a Lens Transform, from the **Edit** menu, select **Create Lens Transformation** or use the button of the same name.



The **Lens Transformation** dialog box is completed to define the classify operation and after completion the new data lens Transformation is placed in the **Map Components Tree** pane. You drag and drop it into the Graphical Map Builder pane and connect it to the input column to create a decision.

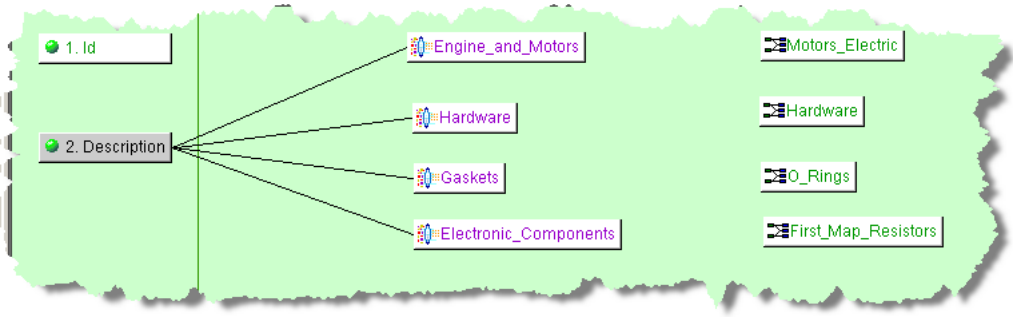


The remaining Lens Transformations are then created to complete all of the necessary engine and motor classification decisions.

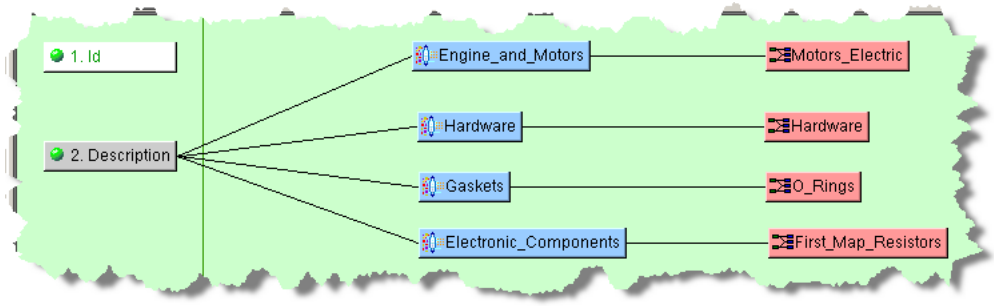


Next, you must route the record to the correct Transformation Map. To do this you can either select an existing Transformation Map from the **Map Component Tree** pane or, if you have not yet built the required transformation, drag a **New Transform Map** node onto your map and name it.

Building your Decision Map in this top-down manner creates any new Transformation Maps for you when you save your Decision Map.



Finally, connect each decision to its corresponding Transformation Map by dragging the appropriate Lens Transform and dropping it onto a Transformation Map. This completes all of the decisions in the map.



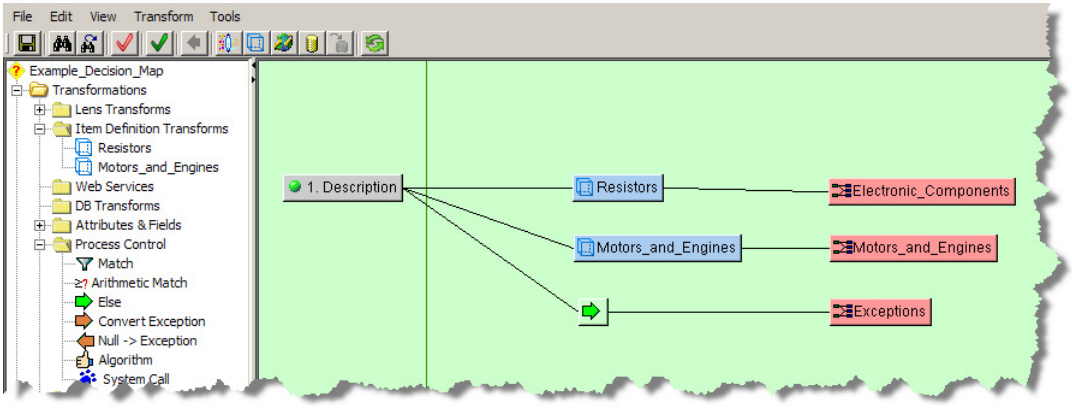
You can validate and test your Decision Maps, after saving it, in the same manner as a Transformation Map. For more information, see "Testing and Validating Transformation Maps" on page 4-54.

Decision Maps can route records to other Decision Maps to as many levels as necessary. The use of Decision Map nesting is complex, contact Oracle Consulting Services for more information.

Creating Decision Maps Using Item Definition Transformations

Item Definition Transformations are used in Decision Maps to determine the selection of which transformation map is used to process the data based on the Item Definition Quality Index, as defined within the Item Definition Transformation.

A Decision Map based on an Item Definition is created in a similar fashion as described in "Creating Decision Maps Using Lens Transformations" on page 5-4. The following example uses the same input data and Item Definition transformations are used to route the data to the appropriate output map:



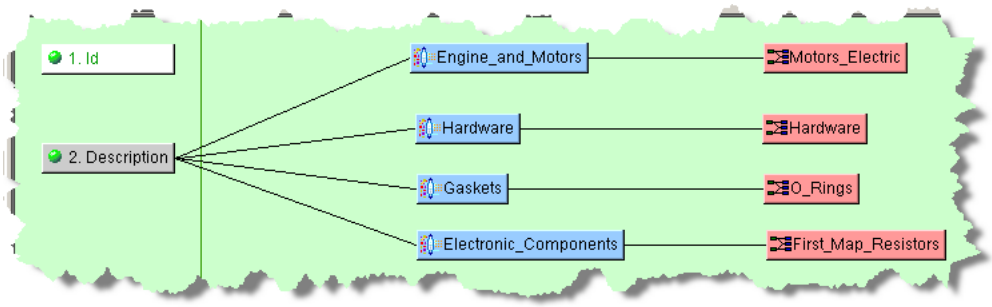
The map processes each line of data to determine if the data meets or exceeds the defined QI, and then it passes the data to the appropriate map for standardization of the description and attribute extraction. The resistor line of data is set to meet or exceed a QI of 80%, which routes to the `Electronic_Components` Transformation Map when true. This Item Definition Transformation then standardizes the description and extracts the attributes. If the data does not meet the QI it is passed to any remaining Item Definition transformations, `Motors_and_Engines` in this case. Any data that cannot be processed is passed to the Else (exception) widget to pass the data to the Exceptions map.

You can validate and test your Decision Maps, after saving it, in the same manner as a Transformation Map. For more information, see "Testing and Validating Transformation Maps" on page 4-54.

Navigating Decision Maps

An important feature of Decision Maps is that they allow you to navigate between Maps during creation. If you double-click on a map node, you are advanced to that Transformation Map.

For example, if the following map was open and you double-click the Hardware map, then that map opens:



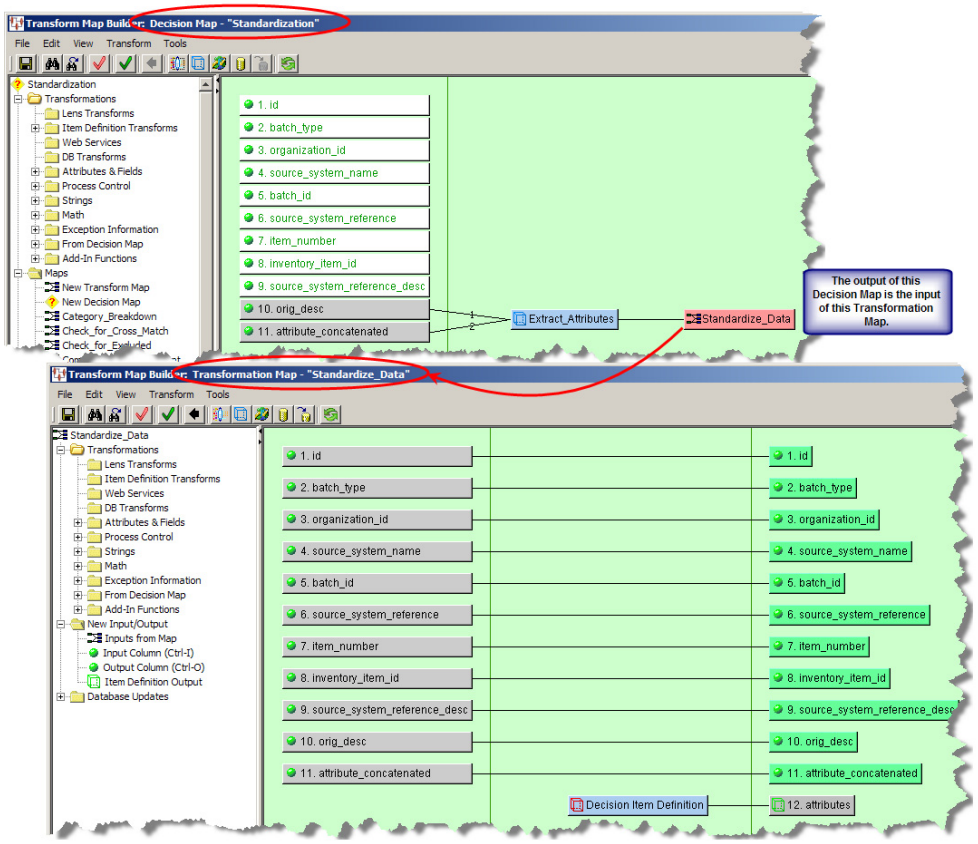
If this is a new Transformation Map, you can create the transformations and output columns you need as previously describe. The input columns are defined for you based on the columns defined in the Decision Map.

You can navigate back to your main Decision Map using the Previous arrow button on the toolbar, which is active only when you have opened a child map.

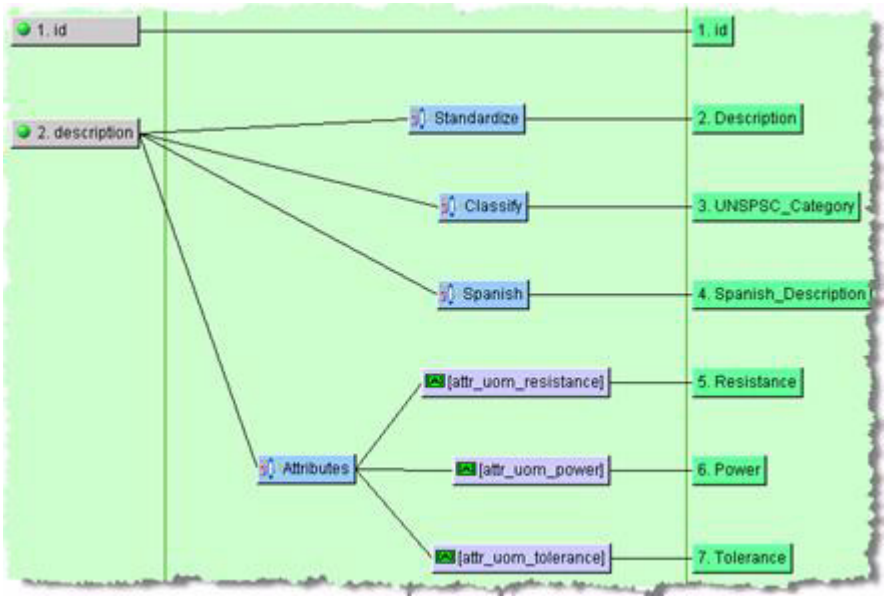
The **Maps** folder, in the **Map Component Tree** pane, appears when you are creating a Decision Map. It contains nodes and widgets to create new maps and any Transform and Decision Maps that have been created so that you can use them in your map design.

Decision Map Example

The following `Standardize` Decision Map creates a `Standardize_Data` Transformation Map to extract the description, and extracts then concatenates up to six key attributes to be standardized:



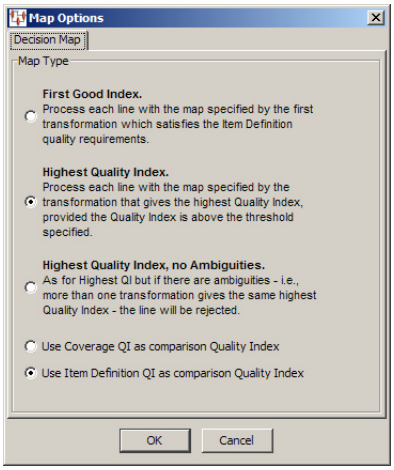
The following Decision Map creates a commodity Transformation Map to classify the record, standardize the description, translate the description to Spanish, and extract up to three key attributes from the description:



Decision Map Options

The map options for Decision Maps are unique and dissimilar to the options that can be set for Transformation Maps. Decision Map options allow you to control how the data input routing decisions are made.

From the **Tools** menu, select **Map Options**.



Select one Map Type processing decision options from the **Map Options** dialog box as follows:

First Good Index

Allows the decision transformation that first meets its quality criteria to route the record to a child map for further processing. The transformations are tried from top to bottom per the map layout.

Highest Quality Index

Allows the decision transformation with the highest quality index to route the record to a child map for further processing. A minimum quality index must be met, as set in the Quality Index field. If no decision transformation meets the minimum quality

index then the record is considered an exception. If two or more decision transformations exceed the quality threshold, have an equal quality index, and their indices are the highest in the map, the first decision transformation will route the record to the associated child map. The first decision transformation is the first transformation node as displayed in the **Graphic Map Builder** pane top-to-bottom.

Highest Quality Index, no Ambiguities
Allows the decision transformation with the highest quality index to route the record to a child map for further processing *or* identifies exceptions. This option is similar to the **Highest Quality Index** option. However, if two or more decision transformations exceed the quality threshold, have an equal quality index, and their indices are the highest in the map, the record is rejected as an exception because the routing is considered ambiguous.

Select one of the remaining two options as follows:

Use Coverage QI as comparison Quality Index
Use to set the span number to be the tie breaker in determining the highest quality index. This default is typically used.

Use Item Definition QI as comparison Quality Index
Used if the decision map is based on an Item Definition transformation.

Click **OK** to set the new options.

6

Advanced Mapping and DSA Concepts

This chapter explains how to implement advanced concepts that you can use to enrich your DSAs, Transformation Maps, and Decision Maps.

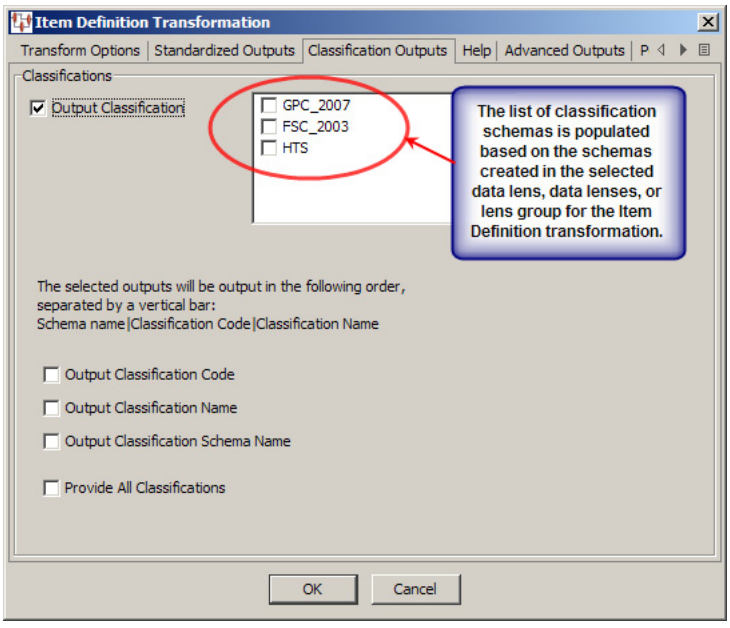
Defining Multiple Classification Transformation Maps

You can define an Item Definition transformation to classify your input data automatically based on several classification schemas rather than just one. This simplifies the classification process because a single step is required rather than several successive steps to use all of the classification schemas available for use in the DSA.

The use of Item Definition transformations, including the **Classification** tab, is described in Defining Item Definition Transforms on page 89.

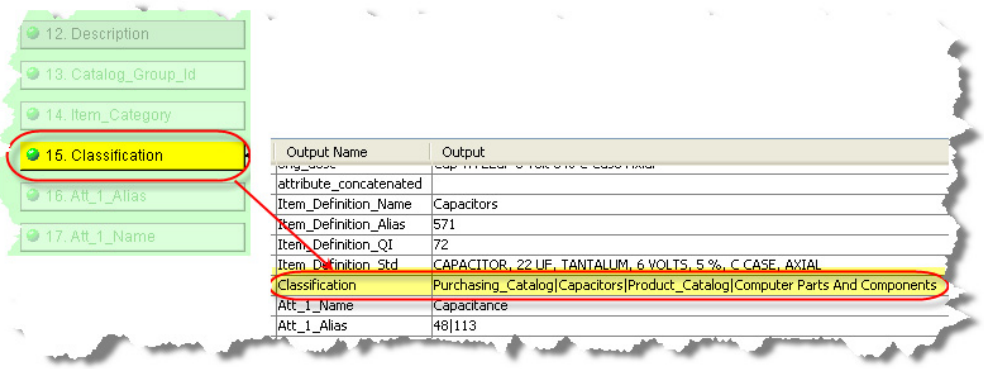
Use the following steps to create a multi-classification Transformation Map:

1. Open the DSA and Transformation Map that you want to be processed with multiple classification schemas.
2. Open an existing Item Definition transformation or create a new one.
3. Select the **Classification** tab.
4. Ensure that the **Output Classification** check box is selected, which activates the list of classification schemas.

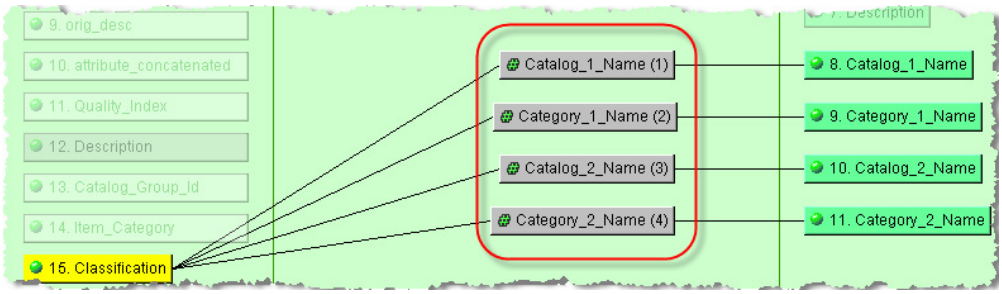


5. Select all classification schemas that you want to apply to your input data from the list of check boxes.
6. Select the items of classification data that you want to output or **Provide All Classifications** for all classification data.
7. Click **OK** to save these changes in the Item Definition transformation.
8. Save and close the map.
9. Select the next step in the DSA and open the associated map.

   The step that follows the multiple classifications Transformation Map must use the **Extract** widget to obtain the information from each of the selected classification schemas. The output from the Item Definition transformation is an overloaded field named Classification by default. It contains the results of the multiple classification request and the values are delimited by a vertical bar (|) as in the following figure:



10. Add an **Extract** widget for each item of classification data, as described in "Strings Widgets" on page 4-14. Ensure that you use the vertical bar as the field separator.
11. Connect each of the **Extract** widgets to the **Classification** input node and individual output nodes as follows:

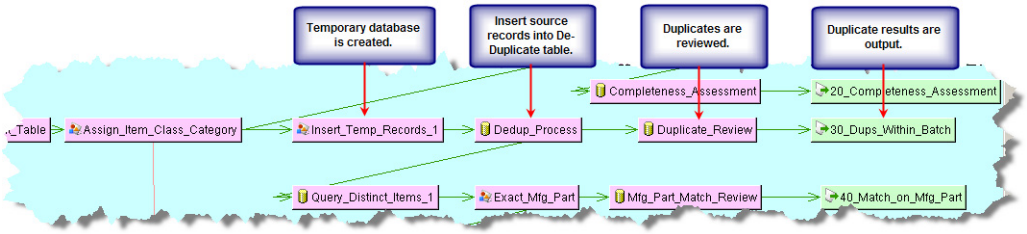


> **Note:** If you edit the **Classification** tab in the Item Definition transformation, you must ensure that you edit the extraction Transformation Map accordingly.

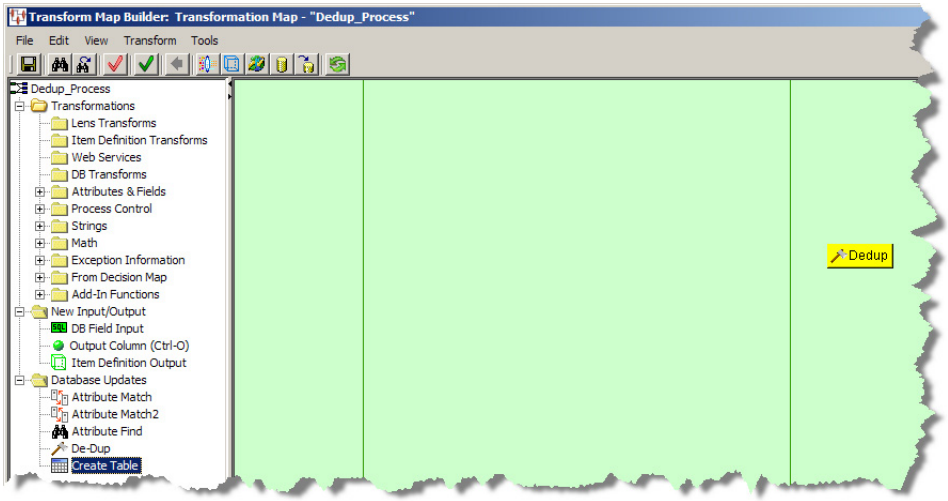
12. Save and close the map.

Using De-Duplicate

The de-duplicate functionality enables you to quickly and easily sort through your input data to find duplicate records of data to prevent them from being included in the output results. It is based on the selection of the Match Results (Multiple) output type, for use in Governance Studio projects, in the DSA output step. The following DSA example illustrates the use of this feature:

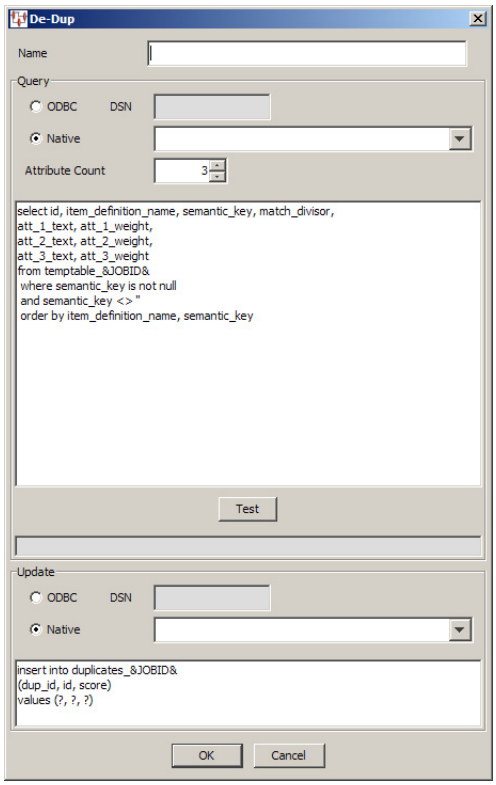


The de-duplicate function is created using a processing step that calls a Transformation Map that contains only the **DeDup** widget.



To create de-duplicate processing in your DSA, follow this process:

1. Open your DSA.
2. Ensure that you have a processing step that is supplying the data records that you want searched for duplicates in a data table.
3. Create or edit the next processing step so that it contains a database Transformation Map. For more information, see "Core Processing Steps" on page 2-2 and "Creating Transformation Maps" on page 4-5.
4. From the **Map Component Tree** pane, drag and drop the **Dedup** widget to the output column.

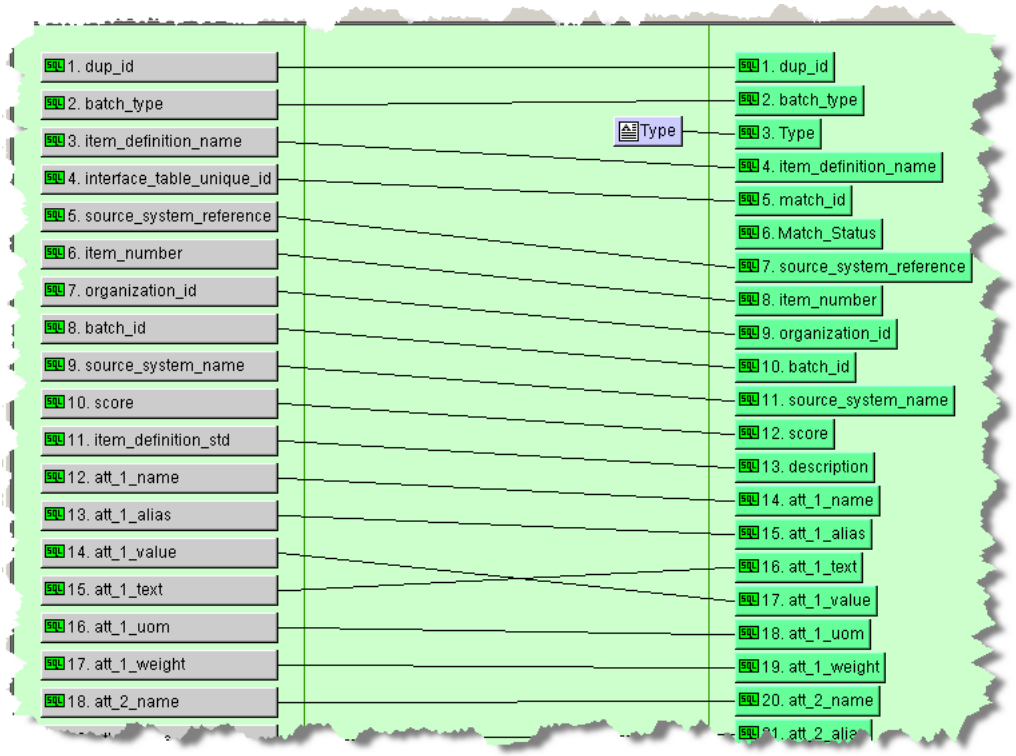


5. Enter a name for the widget.
6. Select the type of database connection.
7. Select the number of attributes to be received.
8. Modify the example SQL statements to match the number of attributes, the name of your data table, etc. Ensure that the number of attributes matches the number of attribute statements to avoid errors. The following example SQL statements generate a `dup_id` and score to create the duplicate groups based on the `item_definition_name`, `semantic_key`, and `match_divisor` attributes.

```
select interface_table_unique_id, item_definition_name, semantic_key, match_
divisor,
att_1_text, att_1_weight,
att_2_text, att_2_weight,
att_3_text, att_3_weight,
att_4_text, att_4_weight,
att_5_text, att_5_weight,
att_6_text, att_6_weight,
att_7_text, att_7_weight,
att_8_text, att_8_weight,
att_9_text, att_9_weight,
att_10_text, att_10_weight,
att_11_text, att_11_weight,
att_12_text, att_12_weight,
att_13_text, att_13_weight,
att_14_text, att_14_weight,
att_15_text, att_15_weight,
att_16_text, att_16_weight,
att_17_text, att_17_weight,
att_18_text, att_18_weight,
att_19_text, att_19_weight,
```

```
att_20_text, att_20_weight
from temp_records&JOBID&
where semantic_key is not null
and source_system_name <> 'Product Information Management Data Hub'
order by item_definition_name, semantic_key
```

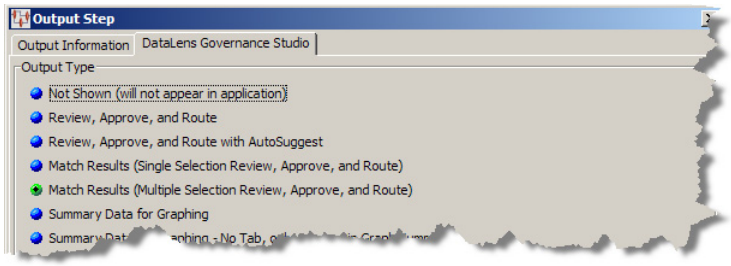
9. Select the database connection in the **Update** section.
10. Modify the example SQL statements in the **Update** section with the name of the table you are going to update with the results, the correct attribute names, and ensure that the number of question marks exactly matches the number of attributes you entered in the list in parentheses.
11. You can use the **Test** button to make sure that your SQL query and database connection operate correctly.
12. Click **OK**.
13. Save and close the map.
14. Create or edit the next DSA processing step so that it receives all of the duplicate attribute data that resulted from the de-duplicate processing in the previous DSA step as in the following example:



Note: You must make sure that `match_id` is listed in the correct output order because all output nodes above `match_id` appear in the top pane in the Governance Studio while output nodes below `match_id` appear in the lower pane

15. Create or edit the text output step to forward the results to the Governance Studio project for review. For more information, see "Text Output Nodes" on page 2-7.

Ensure that the **Match Results (Multiple Selection Review, Approve, and Route)** Output Type is selected on the Governance Studio tab.



16. Connect the de-duplicate DSA processing steps.
17. Save the DSA and check it in so that it can be used by the Governance Studio project.

Matching Attributes

A matching application is a set of two Data Service Applications (DSAs) that you use to find appropriate data records that match pre-specified criteria. A matching process is built upon a data lens (or set of data lenses), used to recognize items by their attributes and to rank attributes in order of importance. This type of matching functionality is flexible and can assemble exact matches or close matches, depending on pre-specified Match Type criteria configured in your data lens. Since you control the Match Types in your data lenses, you can make a change or add a new match type in the data lens and use the changed or added match type in the DSA.

Matching data using a data lens enables you to process your data rapidly in real-time. A matching application can easily handle input data containing thousands or millions records.

Further, that your customers submit frequent purchase requests that you need to process to determine whether you can fulfill those requests from the products contained in your input data. To meet this need, you can construct a matching application that will process (match) the incoming requests against an existing set of records in the source input data known as an *Item Master*.

At a high level, the following shows an example of a matching application, this has the following elements:

1. Creates an attribute cache that includes a Semantic Key to enable matching
   - Passes the original data through one or more data lenses to produce a set of attributes and a Semantic Key per record.
   - Outputs standardized attributes of the data along with the Semantic Key into a database *attribute cache*, which can be stand-alone or connected to the Item Master.
2. Processes a set of records against the attribute cache to determine if there is a match
   - Passes the new, incoming source data through the data lenses.
   - Creates a set of standardized attributes and Semantic Key
   - Checks for matches, using the Semantic Key between the new data and previously created Item Master data held in the attribute cache. A row is considered a match when the Semantic Keys match exactly.

- Presents the matches in the format required for the downstream application.

Attribute Function Data Recognition and Matching

Attributes serve as the foundation of the matching application so it is important to distinguish between the function and configuration of attributes for data recognition and those attributes used in a matching application. You define the attributes in the data lens, and for a matching application, attributes are also weighted.

Attributes for Recognition

Attributes for recognition in the data lens can be one of the following three types:

Required

Necessary to define an item. Without Required attributes, you cannot categorize the designated item

Scoring

Form/fit/function attributes that increase the likelihood that the description describes the designated item

Optional

Ancillary to defining the item-the description does not stand or fall in any way on having this type of attribute included.

For example, for the following data record:

MTR, 3/4HP, 1725RPM, 115/208-230V, 56

You might use the following Required, Scoring and Optional attributes in your data lens:

Attribute Type	Function	Example
Required	Necessary to define item	Motor
		Voltage
Scoring	Horsepower	3/4 HP
	RPM	1725 RPMs
	Mounting Type	
Optional	Brand, Model #	Harley-Davidson
		MN-1528-FX2

Only the Required and Scoring attributes participate in the Item definition quality index, which is the scoring mechanism within the data lens.

Attributes for Matching

For matching, all types of attributes can participate numerically in the matching process, including Optional attributes; this makes the matching application extremely flexible and effective while leveraging the power of the data lens to correctly identify items and standardize text.

In a matching application, you must provide some additional information about the attributes to the data lens. This additional information specifies how important each attribute is in determining whether another data record is a suitable match for downstream processes, such as suggesting alternative products to a purchaser. You can do this by creating one or more standardization types in the data lens that specify

which attributes should participate in the match and the weight to be given each attribute. You can specify and weight attributes of any type (required, scoring, or optional) as a match type.

Components of a Matching Application

For matching the attributes of incoming source records against the attributes contained in an Item Master, you must have the following components:

- an Item Master that holds uniquely identified product descriptions,
- a data lens or lenses that recognize descriptions and output standardized attributes,
- a database that holds the attribute output from the attribute cache,
- all records in the Item Master (and thus attribute cache) are identified by a `match_id`, and
- the incoming source records are uniquely identified by a `request_id` to match against the Item Master using the attribute cache.

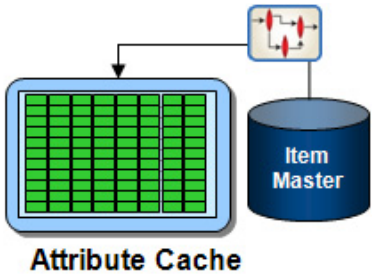
For example, suppose electric motors data is being matched based on the following criteria:

120 Volts

3100-3150 RPMs

The matching process includes two steps. The following figure illustrates the first step in the matching process where you create a match type using a set of attributes:

Match_ID	Item Definition Name	Attr1	Attr2	Attr3
1	Electric Motor	300	120V	3450
2	Electric Motor	250	120V	3150
3	Electric Motor	200	120V	3450
4	Electric Motor	300	120V	3450



In this example, Attr2 (volts) and Attr3 (rpm) are a part of the match type.

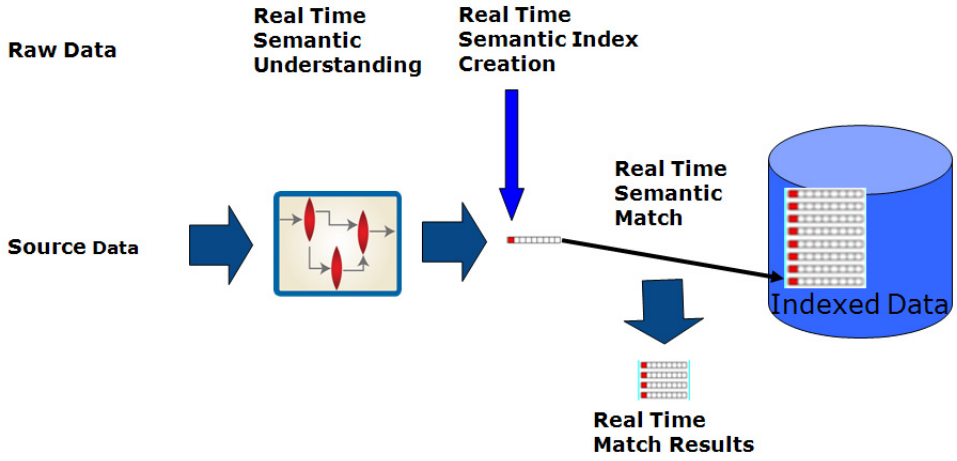
The second step evaluates the incoming record attribute values against the existing records. Any incoming record that has the same attribute values will be considered a match.

There are two types of matching functions and both match data using a Semantic Key cache. A Semantic Key is an obfuscated string of text that is created for each data record based on the required matching attributes set in your data lens using a semantic matching algorithm. The Item Definition Transform generates the Semantic Key cache. The Semantic Key cache (or index) is container of data that contains the Semantic Key, id, Item Definition, attributes, attribute weight, and text for each data record. The two matching functions are as follows:

Attribute Match (Semantic Key1)

This matching function uses a static Semantic Key cache, consisting of a fixed set of matching attributes and weights. This Semantic Key 1 cache can be used for matching

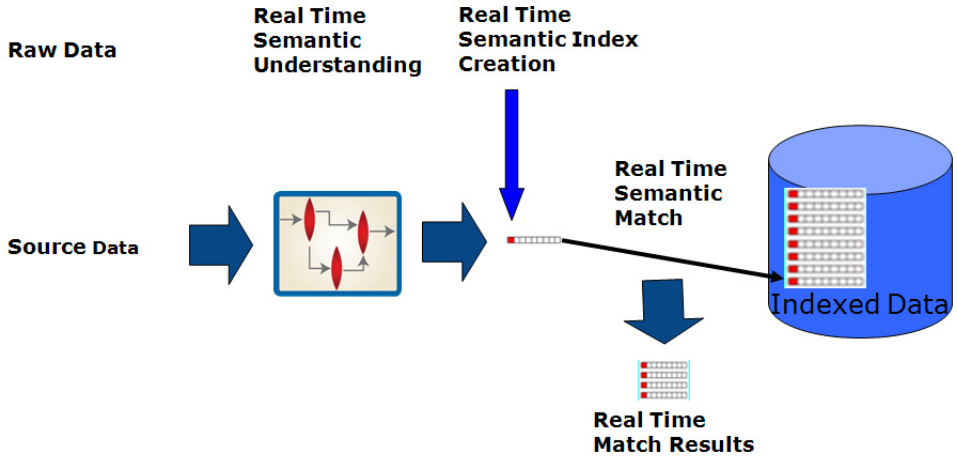
until the required matching attributes or weights are changed. When this occurs, the Semantic Key must be rebuilt. Each data lens Match Type requires an individual Semantic Key.



Attribute Match2 (Semantic Key2)

This matching function uses a Semantic Key2 cache that is built independently of the match weights, allowing you to use the same Semantic Key2 cache for multiple match types without the need either to have multiple caches for each match type or to rebuild the cache if the match weights change as is the case for Semantic Key1. The Semantic Key2 operates with multiple Match Types that provide the match weights to the match algorithm in real time. In addition, Semantic Key2 is created based on the Unit Conversion and Standardization Types selections in the Lens Transformation. The cache only needs to rebuilt if new attributes are added to the set.

The Semantic Key2 also uses a memory cache to improve performance. The memory cache is synchronized with a Semantic Key2 table that dynamically ensures that the Semantic Key cache is current.



For more information about configuring Match Types and match attributes, see *Oracle Product Data Quality Knowledge Studio Reference Guide.*

Note: Both Attribute Match Widget types (Attribute Match - Semantic Key1 and Attribute Match2 - Semantic Key2) requires at least two data inputs: id and an Item Definition Transformation Output. Furthermore, it must be the output of a Decision Map if you want to use multiple lenses.

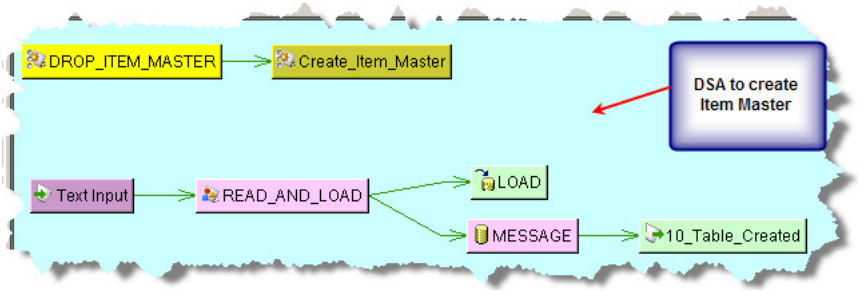
Using Semantic Key1 for Matching

You can construct a matching application that processes the incoming requests and presents the records in the Item Master that match the requests. The recommended best practice is to design a DSA that creates an attribute cache from the Item Master, and then a DSA that processes the matching in this order as in the following two sections.

For more information about creating DSAs, see "Creating or Opening a DSA" on page 2-1.

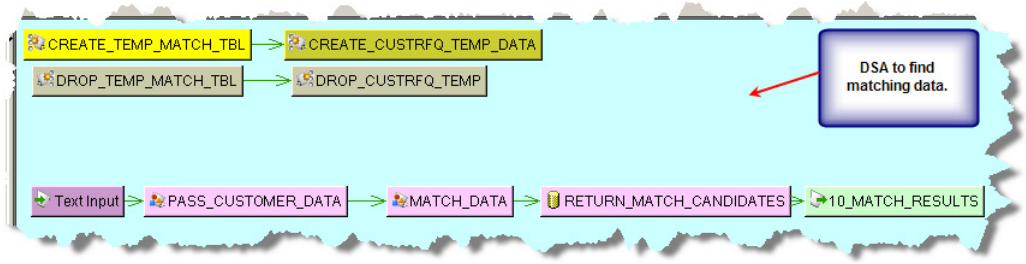
Step 1: Creating an Attribute Cache (Semantic Key Cache) DSA

Create your Item Master DSA and ensure that you include all attributes, even those that are not required. You must have the Match Type, attribute name, attribute text, `match_threshold`, `match_divisor`, and `match_weight` values. This is the data that you will match your input data against. Ensure that you create pre-processing steps to create the actual Item Master table. In this example, the LOAD database output step creates the attribute cache.



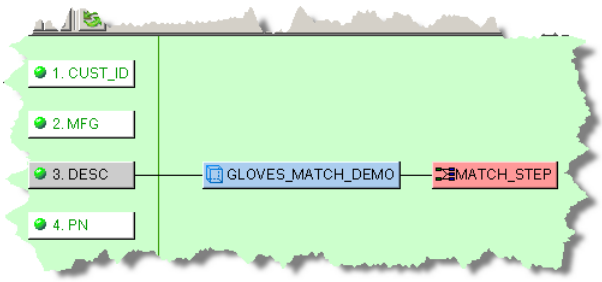
Step 2: Creating an Attribute Matching Application Using the Attribute Cache

Create the DSA to process your input data to look for matches. You should create a table that contains your input data and a table to contain the matches found during processing. The Transformation Maps created for the `PASS_CUSTOMER_DATA` processing step that passes the source input data is as follows:

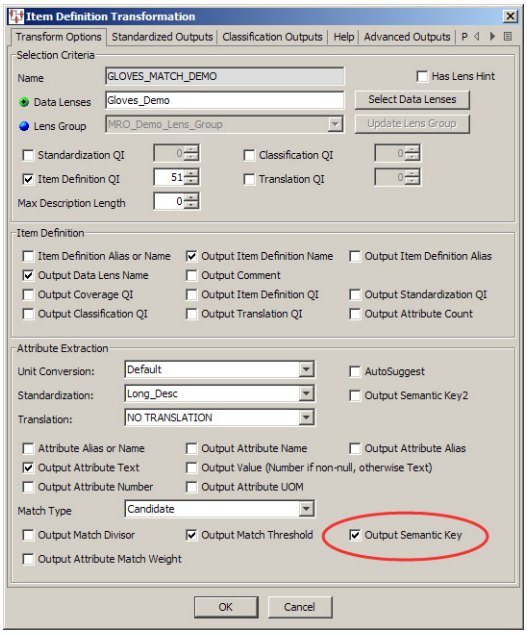


The `MATCH_DATA` processing step includes a Decision Map that includes a Lens Transformation and a decision output node to pass all of the attribute data.

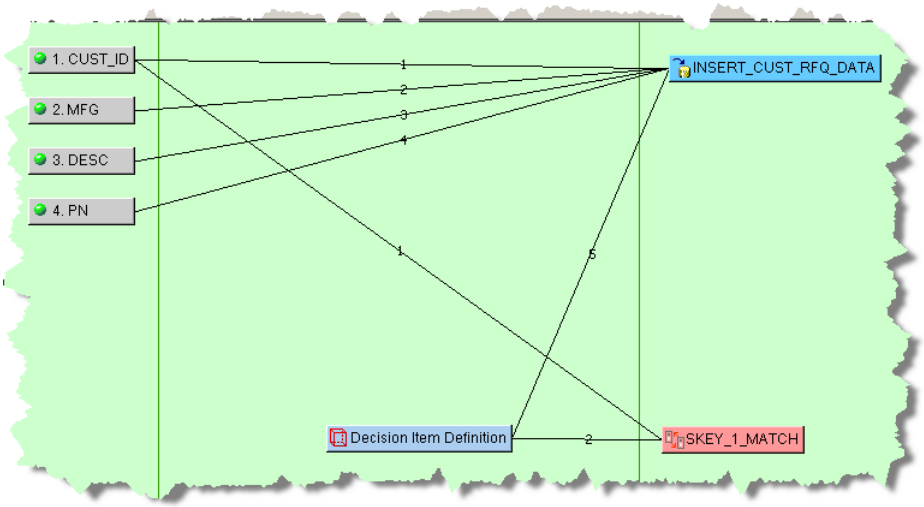
Note: The match relies on the same lenses and match types for both the cache creation and match.



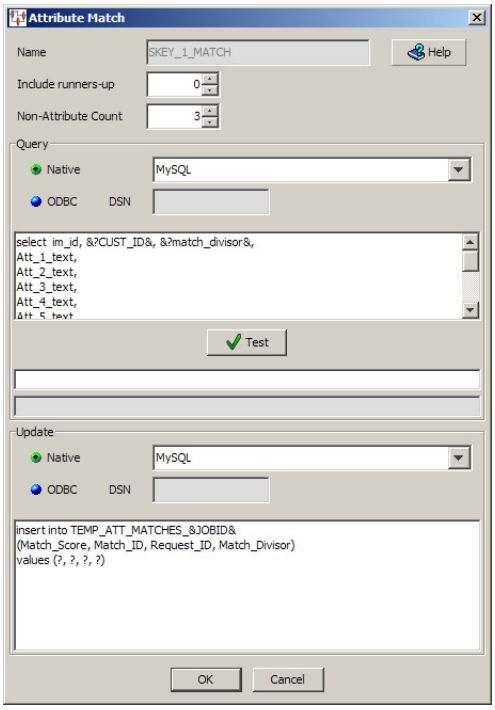
The Lens Transformation is configured with the appropriate Lens Group and attributes as follows.



The MATCH_STEP Decision Map that actually processes the data searching for matches contains the required attributes, the **Attribute Match** widget, SKEY_1_MATCH Output column node, Item Definition processing decision, and table Output node to pass the data is as follows:



When you add an **Attribute Match** widget to the output column of a Transformation Map, you must provide the attribute matching counts, SQL statements that you want to use to search for matches, and the SQL statements to update the table that contains the matching data. The `SKEY_1_MATCH` Attribute Match query consists of the following information:



This example does not search for any runners-up matching data so those records that were a close match are not included in the output because the **Include runner-up score** list is zero (0). The **Non-Attribute Count** list must match how many non-attributes are in the SQL `select` clause.

In the **Query** section, the SQL statements must adhere to the following rules:

- The first variable in the `select` clause *must* be the Match ID. The Match ID is the unique id for the match.

- The second and third variables *must* be the Request ID and Match Divisor.
- All non-attributes *must* come *before* the attributes listed in the `select` clause so that the attributes are looped through to add the match weights preset in the data lenses for the matched attributes.

The keywords that can be used to get data from the Item Definition are as follows. Remember that to use them you need to wrap them with `&?` and `&`. For example, `&?Att_3_Value&`.

Non-attribute values

- `Item_Name`
- `Item_QI`
- `Attribute_Count`
- `Match_Divisor`
- `Match_Threshold`
- `Semantic_Key`

Attribute values

Using `attribute` 3 as an example:

- `Att_3_Value`
- `Att_3_Text`
- `Att_3_Number`
- `Att_3_Units`
- `Att_3_Name`

The following SQL statements are used in the example **Attribute Match** widget:

```
select im_id, &?CUST_ID&, &?match_divisor&,
Att_1_text,
Att_2_text,
Att_3_text,
Att_4_text,
Att_5_text,
Att_6_text,
Att_7_text,
Att_8_text,
Att_9_text,
Att_10_text
from GLOVES_ATTR_CACHE
where semantic_key=&?semantic_key&
```

The Match ID is `im_id`, the Request ID is `CUST_ID`, and the Match Divisor is `match_divisor`. You must modify the sample statements so that it matches your input nodes, table names, etc. Since there are three non-attribute values in the `select` clause, the **Non-Attribute Count** is set to 3.

It is a good practice to include the same number of attribute placeholders in your SQL statements for all of your attributes even if you do not use them in matching to avoid errors. This also allows you to use any attribute in a subsequent processing step.

You can review the help topics using either of the help buttons. Your matching SQL statements can be tested by supplying a value for each parameter using a vertical bar as a delimiter in the active field, and then clicking **Test**. The database connections, as

well as the entered statements are tested and the results appear in the field below the values you enter.

The **Update** section of the **Attribute Match** dialog box is used to update the table you created to contain your match output data.

The Match Score (computed by the Application Studio) must be the first variable in the `insert` or `update` SQL statement. The `insert` or `update` clause is automatically populated with the values retrieved in the `select` clause and is in the order specified by in the `select` clause.

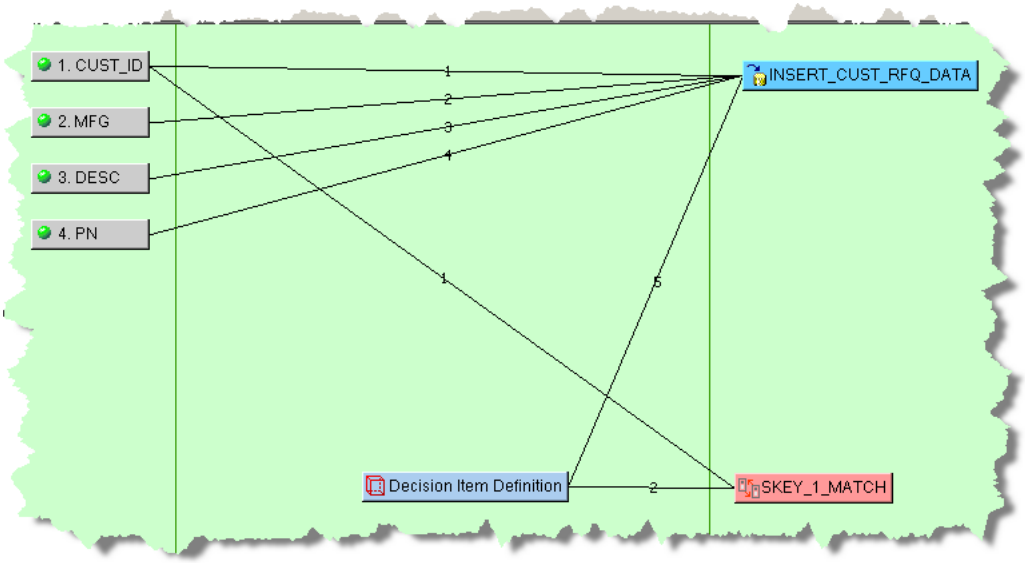
> **Tip:** To avoid errors, ensure that your `values` clause has one question mark (?) for every node (column) in the insert or update statement.

In this example, the table update statements are as follows:

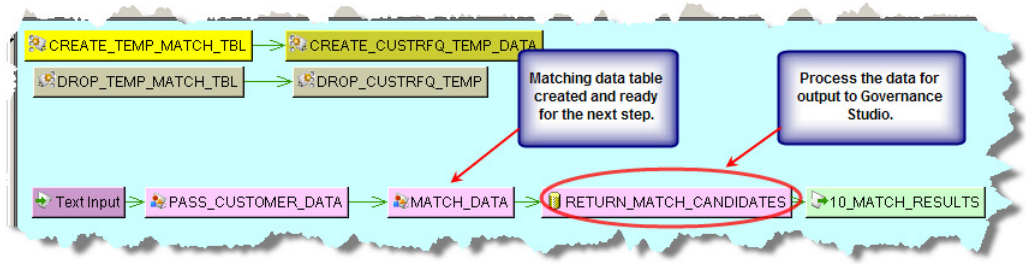
```
insert into TEMP_ATT_MATCHES_&JOBID&
(Match_Score, Match_ID, Request_ID, Match_Divisor)
values (?, ?, ?, ?)
```

The Match Score, match ID, Request ID, and Match Divisor values will be inserted into the matching table.

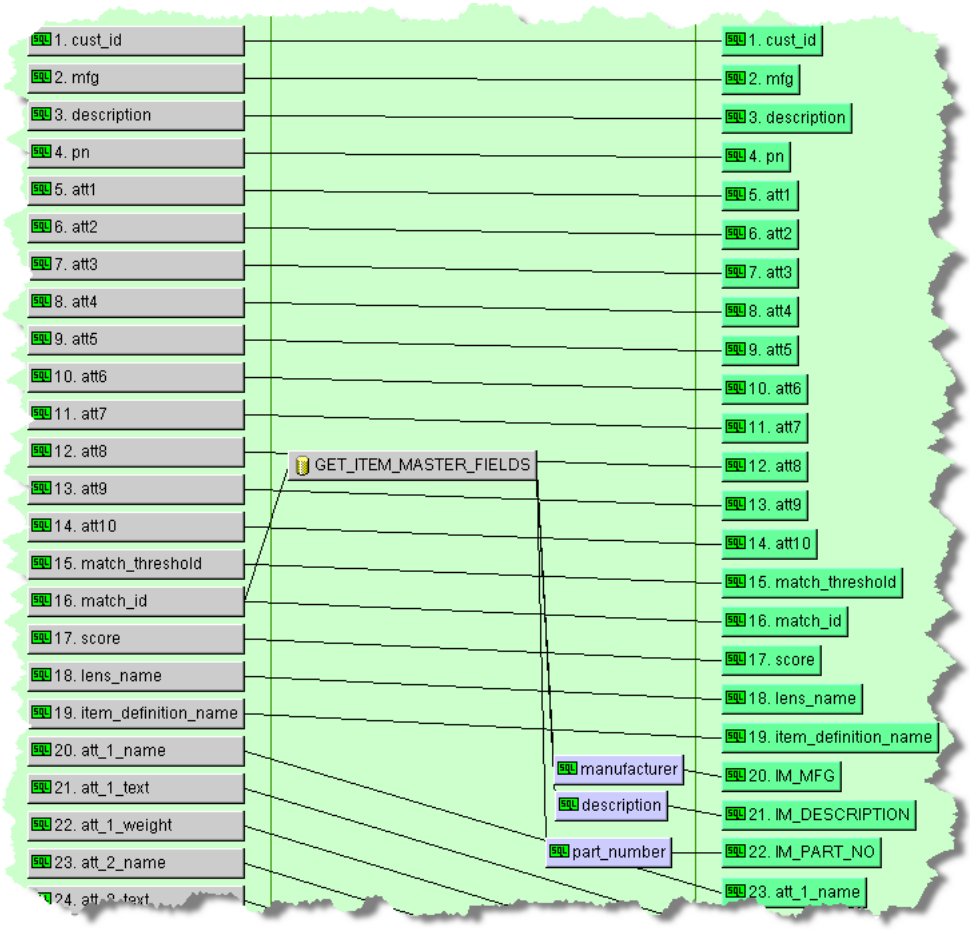
The Attribute Match widget, `SKEY_1_MATCH` is connected to the `CUST_ID` and Decision Item Definition input nodes as is necessary.



The Transformation Map Output node, `INSERT_CUST_RFQ_DATA` contains the data from the matching table to pass from the `MATCH_STEP` Decision Map to the `RETURN_MATCH_CANDIDATES` processing step Transformation Map in the DSA as follows:



The RETURN_MATCH_CANDIDATES Transformation Map is configured as follows:



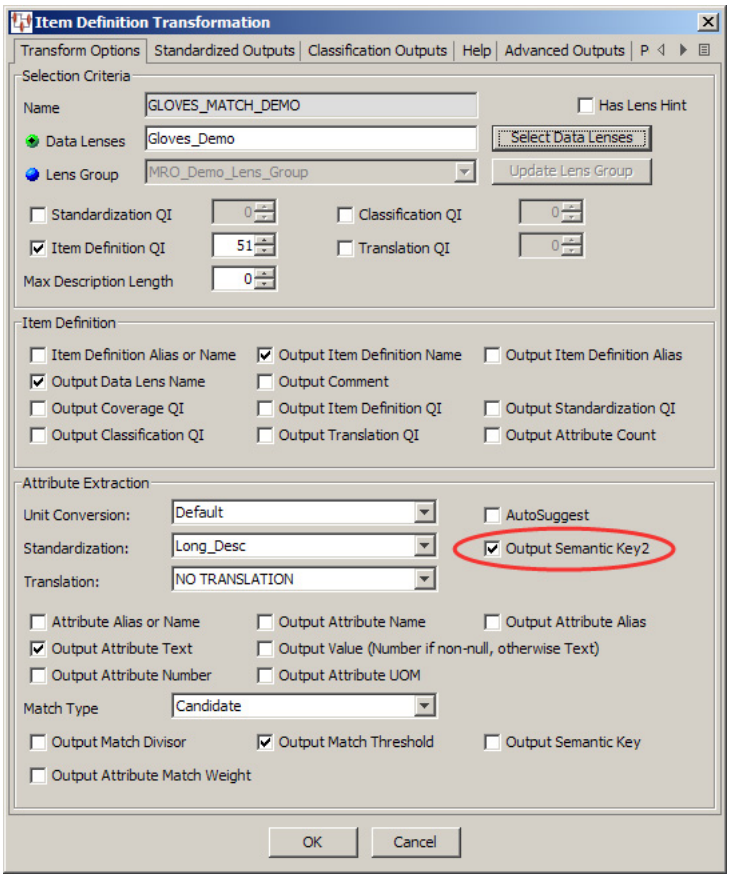
This map inserts the match data, with all of the attributes, into the appropriate output nodes in preparation for the DSA output step.

The output step of the matching DSA is configured based on how you want view the matching data in the Governance and Knowledge Studios. You must select one of the Match Results output types on the **DataLens Governance Studio** tab. For more information about configuring output steps, see "Text Output Nodes" on page 2-7.

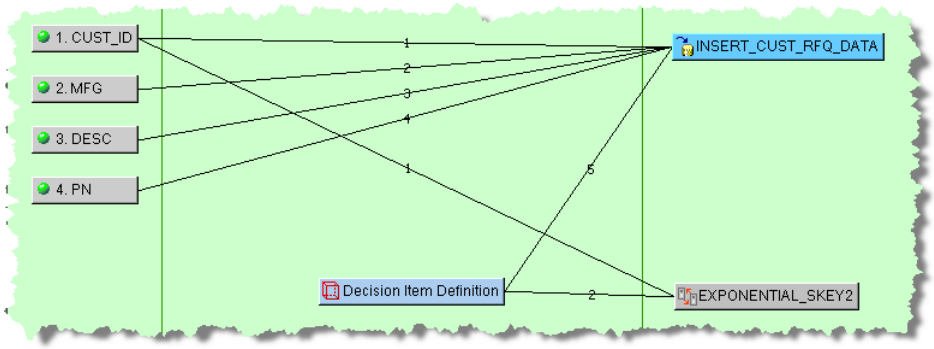
Using Semantic Key2 for Matching

Designing a matching application using Semantic Key2 operates in a very similar manner to Semantic Key1 and its implementation is essentially the same. For more information, see "Using Semantic Key1 for Matching" on page 6-10.

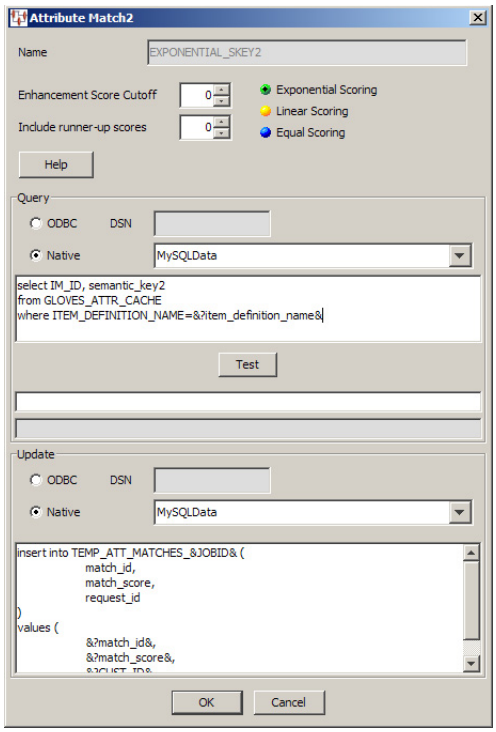
The difference is that the **Attribute Match2** Widget query relies on the Item Definition name and the Item Definition output *must* include the Semantic Key2 as in the following:



The MATCH_STEP Decision Map that actually processes the data searching for matches contains the required attributes, the **Attribute Match2** widget, EXPONENTIAL_SKEY2 output column node, Item Definition processing decision, and table Output node to pass the data is as follows:

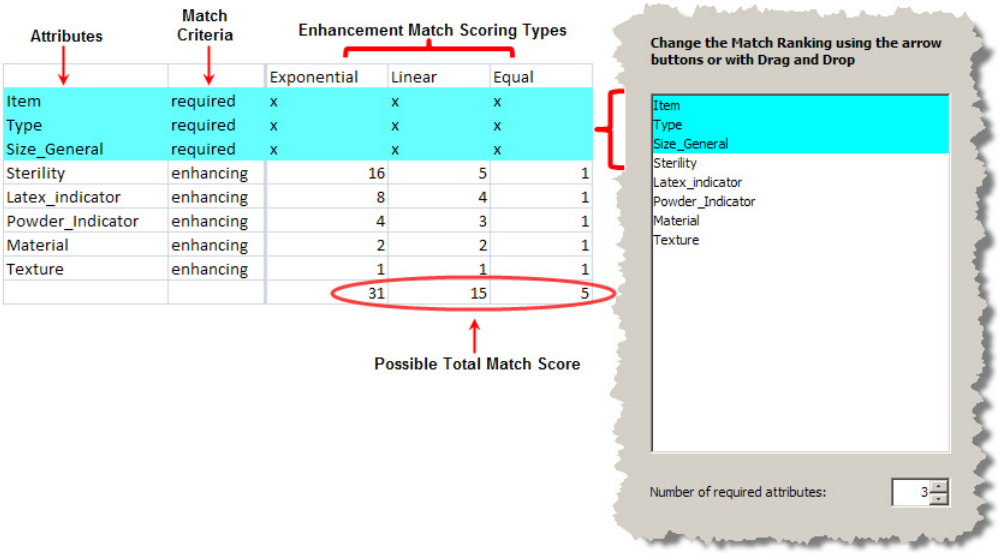


When you add an **Attribute Match2** widget to the output column of a Transformation Map, you must provide the attribute matching counts, SQL statements that you want to use to search for matches, and the SQL statements to update the table that contains the matching data. The EXPONENTIAL_SKEY2 Attribute Match2 query consists of the following information:



This example does not search for any runners-up matching data so those records that were a close match are not included in the output because the **Include runner-up score** list is zero (0).

In addition, when the required attributes are met, you can score matches further using one of the following methods:



Exponential Scoring

Scoring is computed as an exponential power of two; the match score begins with the lowest attribute scoring of 2 or 1 and increases each remaining attribute by a power of two for the total set of attributes.

Linear Scoring
Scoring is computed in a linear manner; the match score begins at the total number of available enhancement attributes and descends in value by one for each selected attribute.

Equal Scoring
Scoring is weighted equally for all selected match attributes.

In this example, the Exponential Scoring option is selected (by default) though additional scoring is not used as indicated by the zero in the **Enhancement Score Cutoff** check box.

In the **Query** section, the SQL statements must adhere to the following rules:

- The first variable in the `select` clause *must* be the Match ID. The Match ID is the unique id for the match.
- The second variable in the `select` clause *must* be the Semantic Key2.
- The `where` clause *must* select matches based on the Item Definition and *may* select on data that is passed into the transformation.

> **Note:** No other data from the matching table is required or used by the matching application.

The following SQL statements are used in the example **Attribute Match2** widget:

```
select IM_ID, semantic_key2
from GLOVES_ATTR_CACHE
where ITEM_DEFINITION_NAME=&?item_definition_name&
```

The Match ID is `IM_ID` and the Semantic Key2 variable is `semantic_key2`. You must modify the sample statements so that it matches your input nodes, table names, etc.

You can review the help topics using either of the help buttons. Your matching SQL statements can be tested by supplying a value for each parameter using a vertical bar as a delimiter in the active field, and then clicking **Test**. The database connections, as well as the entered statements are tested and the results appear in the field below the values you enter.

The **Update** section of the **Attribute Match2** dialog box is used to update the table you created to contain your match output data. The rules for the `insert` clause are extremely flexible. You can insert various pieces of data, using the `&?name&` mechanism. The order is irrelevant as long as the field names and data names correspond correctly. The available data is the following:

From the Query:
```
&?match_id&
```

Computed
```
&?match_score&
```

From the Item Definition Data
```
&?match_score&
&?att_1_value&
&?item_definition_name&
&?item_definition_qi&
```

```
&?attribute_count&
&?match_divisor&
&?match_threshold&
&?semantic_key2&
```

From the Item Definition Attribute Data (using attribute 1 as an example)

```
&?att_1_text&
&?att_1_name&
&?att_1_alias&
&?att_1_uom&
&?att_1_weight&
```

From the Input Data

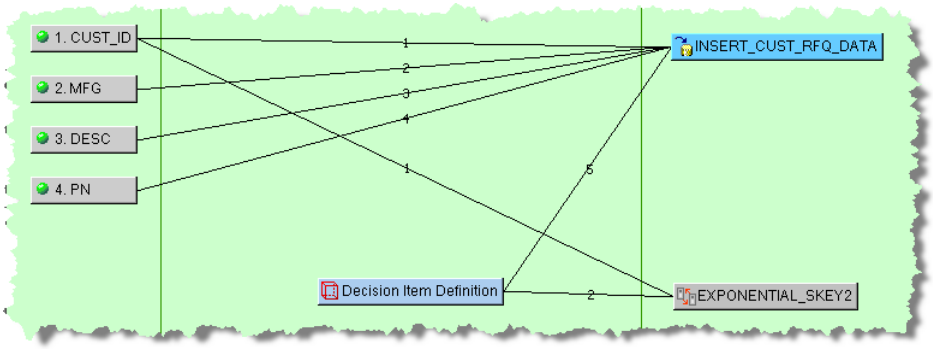
Any node that is connected to the Attribute Match2 transformation, such as a Request ID node, can be named. For example, the data from a node named `request_id` would be specified as `&?request_id&`.

In this example, the table update statements are as follows:

```
insert into TEMP_ATT_MATCHES_&JOBID& (
match_id,
match_score,
request_id
)
values (
&?match_id&,
&?match_score&,
&?CUST_ID&
)
```

The Match ID and Semantic Key2 values will be inserted into the matching table.

The **Attribute Match** widget, `EXPONENTIAL_SKEY2` is then connected to the `CUST_ID` and Decision Item Definition input nodes as is necessary.

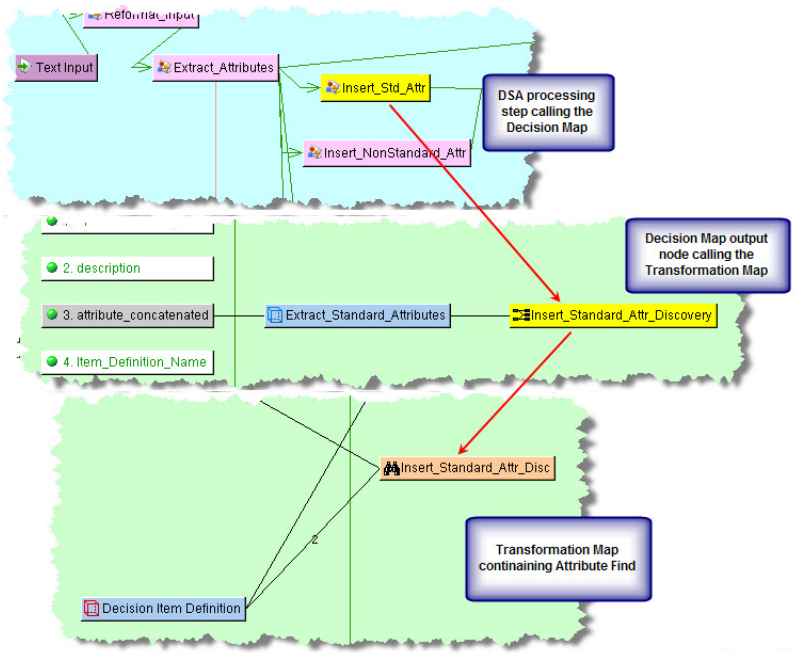


Completing the matching application process is identical to the Semantic Key1 process as described in "Using Semantic Key1 for Matching" on page 6-10.

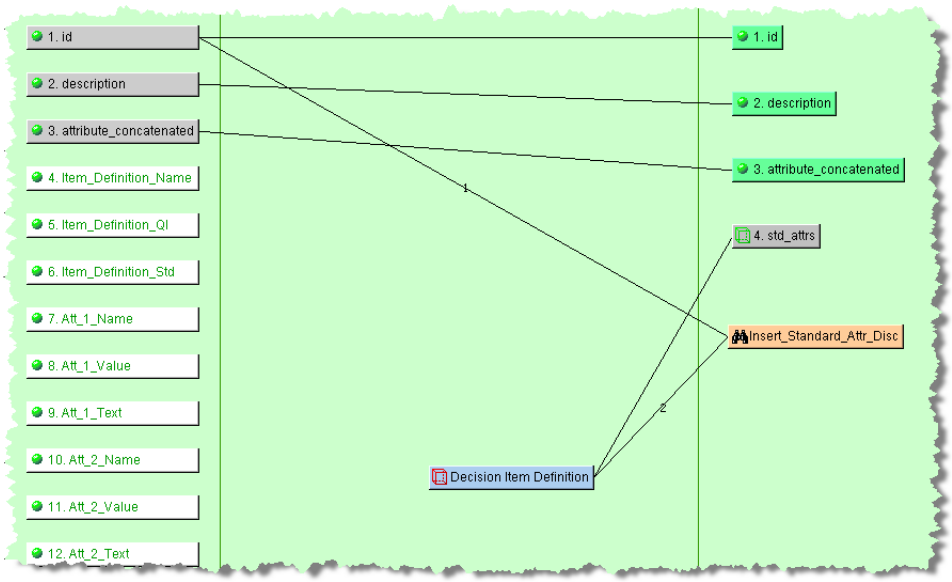
Finding Attributes

Finding unparsed, unknown, and unattributed text to convert to attributes, which enriches your data, is a powerful tool.

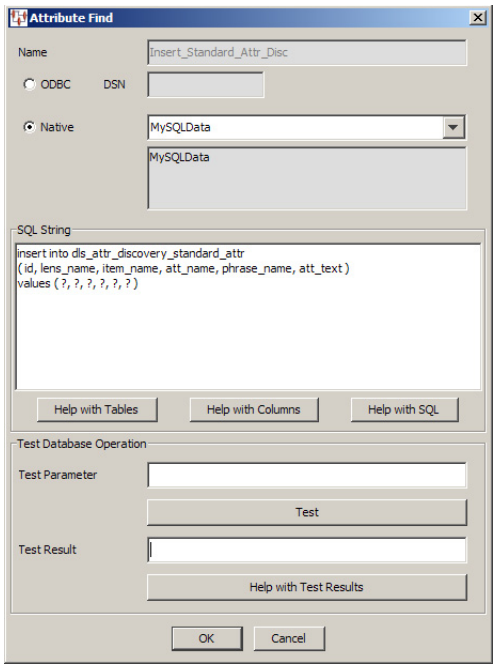
Similar to the matching functionality, to use the **Attribute Find** function, you must have two data inputs, one of which must be a Lens Transformation, as in this example. Further, it must be the output of a Decision Map as in the following example:



The following example illustrates the use of the Attribute Find functionality as it takes input data, finds standard attributes that are processed by an Item Definition Decision, and then provides the output data in an Item Definition node for use in the next step of a DSA:



The **Attribute Find** widget is used in the output column and the **Attribute Find** dialog box is used in the same manner as described in "Database Query Data Input" on page 4-7.



The requirements for the fields in the `insert` clause are as follows:

- There must be a minimum of two nodes connected to the **Attribute Find** widget: an ID input node and an Item Definition transformation node.
- The ID field is required and is obtained from the first connected node.

  In addition to this field, you can include additional fields that get their data from the corresponding connected nodes. There *must* be exactly the same number of connected non-Item Definition nodes as there are non-Item Definition values in the `insert` clause and these nodes *must* be in the same order as the fields in the `insert` clause.

  For example, if you want to pass in the Organization ID and the Department ID, you would have four nodes connected to the **Attribute Find** widget. The non-Item Definition nodes *must* be in the same order as the fields in the `insert` clause.

  A sample `insert` clause is as follows:

```
insert into TABLE_&JOBID&
     ( id, organization_id, department_id, lens_name, item_name, att_name,
       phrase_name, att_text )
     values ( ?, ?, ?, ?, ?, ?, ?, ? )
```

- The `lens_name`, `item_name`, `att_name`, `phrase_name`, `att_text` fields are all obtained from the connected Item Definition node and have the following requirements:
  - They must appear in the following order:

    `id, lens_name, item_name, att_name, phrase_name, att_text`
  - The spelling must match the corresponding fields in the database table.
  - They must be the last five fields of the `insert` clause.
  - There can be no intervening fields

The example SQL statements for the **Attribute Find** widget, `Insert_Standard_Attr_Dsc`, insert the following standard attributes into a table: the ID, lens name, Item Definition name, attribute name, associated phrase name, and the attribute text.

> **Note:** You must ensure that the number of values matches the number of question marks in the `values ( ?, ?, ?, ?, ?, ? )` clause to avoid errors.

The table created, named `dls_attr_discovery_standard_attr` is then passed to the next DSA processing step via the `std_attr` Item Definition output node.

Matching Ngrams

An **Ngram** can be a single word (Unigram), two words (Bigram), or three words (Trigram). An Ngram matching application is similar in concept to an attribute matching application and consists of a set of two DSAs that you use to find appropriate data records that match a set of Ngrams from an input record. An Ngram matching process uses an Ngram Cache, which is similar to an attribute cache (as described in the previous section). The **Ngram Create** and **Ngram Match** widgets use id and description as input then automatically generates Unigrams, Bigrams, and Trigrams from the description along with a set of key values to insert them into a table (Ngram Cache).

For example, you can assess a set of descriptions to see if there are matches against an existing set of descriptions or a classification. Another example is to conduct a data assessment to determine the number of variants that exist for a given gold form of a word or phrase (Ngram). The word PACK might be found as a variant in the data as PAKC or PAC. The data assessment has the ability to find fuzzy matches that represent variants or errors in the data set.

To meet this need, you can construct an Ngram matching application that will process (match) the incoming requests against an existing set of records in the Ngram Cache.

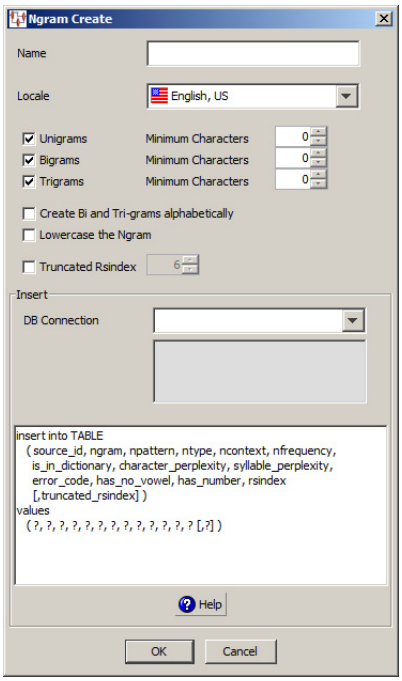
The Ngram computation process creates and uses a set of keyword values. The process takes the input record and creates the set of Ngrams dynamically. If you chose to use Unigram, Bigram, and Trigram as part of your data assessment, each type of Ngram (`NTYPE`) is processed by comparing the selected Ngrams types of the input record to those found in the Ngram Cache by type. As a result, if you only select Unigrams then the process only compares Unigrams (`NTYPE = 1`).

The following sections illustrate a high-level example of an Ngram matching application.

Step 1: Creating an Ngram Cache that Includes a Set of Unique Ngrams with Frequencies

- Passes the original descriptions through the **Ngram Create** widget to produce a set of Ngrams per record.
- Outputs a set of Ngrams and key values.

Create your Ngram cache DSA using the **Ngram Create** widget.



Use the **Ngram Create** dialog box as follows.

Name Field

Enter a name for your Ngram Create widget. This name is displayed in the **Map Component Tree** pane and in the **Graphical Map Builder** pane.

Locale List

Select the language that this Ngram matching widget will use.

Type of Ngram Check Boxes

Select the type of Ngrams you want to create and a minimum number of characters that will be created if desired.

Unigram

Creates a unigram and optionally a minimum number of characters if the value is greater than zero.

Bigram

Creates a bigram and optionally a minimum number of characters if the value is greater than zero.

Trigram

Creates a trigram and optionally a minimum number of characters if the value is greater than zero.

Create Bi and Tri-grams alphabetically Check Box

Select this check box to alphabetize the bigrams and trigrams created. Selecting this option allows matches to succeed when words are not in same order though are in fact the same phrase. For example, “Blood Test” will not match against “Test Blood” without this option selected.

Lowercase the Ngram Check Box

Select this check box to lowercase all Ngrams created. This ensures that matches are consistently found because certain databases or database selections enforce case. The result is that "Pack" does not match "pack".

Truncated Rsindex

By default, the `rsindex` value is created with a Soundex algorithm to set a sounds like value that can be used for data matching.

Select this check box to truncate the `rsindex` value to a specific number of characters. A lower number yields more matching results though the default, six, or more may yield closer matching results depending on your data.

You must change the SQL statements to reflect the additional output field and one more question mark for the 14th value. The following insert statement is an example with the additional field, `rs6index`.

```
insert into ngram_cache
   ( source_id, ngram, npattern, ntype, ncontext, nfrequency,
     is_in_dictionary, character_perplexity, syllable_perplexity,
     error_code, has_no_vowel, has_number, rsindex, rs6index )
values
   ( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? )
```

Insert Section

Use this section as follows:

DB Connection

You must select the type of database connection that you want to use. The list of database connections is populated based on those that you are configured in the Oracle DataLens Server. If the type of database connection is not listed, you must configure it in the Oracle DataLens Server so that it is available for selection when creating Transformation Maps.

SQL Field

Use the field to construct your database query with standard SQL query statements and syntax. For example, you could use the following SQL statements:

```
insert into DLS_SEARCH_NGRAMS&DATASETNAME&
( source_id, ngram, npattern, ntype, ncontext, nfrequency,
is_in_dictionary, character_perplexity, syllable_perplexity,
error_code, has_no_vowel, has_number, rsindex )
values
( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? )
```

> **Note:** All 13 variables must be inserted into the table, which must include 13 question marks in the `values` clause.

In this example, the query is inserting the following information into the table:

Variable	Description	Output Value
`source_id`	A unique id for the input line.	1302346
`ngram`	Unigram, Bigram, or Trigram value.	REV

Variable	Description	Output Value
pattern	'A' for alpha or '9' for number and blanks between words.	AAA
ntype	1 - Unigram 2 - Bigram 3 - Trigram	1
ncontext	0 - Ngram found at beginning 1 - Ngram found at end 2 - Ngram found in both	1
nfrequency	The number of times that the Ngram appears in the line.	1
is_in_dictionary	0 - Ngram not in dictionary 1 - Ngram in dictionary The dictionary is locale dependent; for example, en_US.	0
character_perplexity	Internal field; used for diagnostics.	N/A
syllable_perplexity	Internal field; used for diagnostics.	N/A
error_code	Internal field; used for diagnostics.	N/A
no_vowel	0 - Ngram has no vowels 1 - Ngram has vowels	0
has_number	0 - Ngram has no numbers 1 - Ngram has numbers	0
rsindex	Soundex pattern with first character being the first letter of the Ngram.	R902

This insert clause must be compatible with your database. At run-time, the question mark character is replaced with the transformation input data. This allows you to create a database transformation that varies with the content of the record being processed. Additionally, when database transforms are used to aggregate data fields from several different data sources, common access key information can be used across all data sources.

Note: Duplicate Ngrams are not saved. For example, if the Unigram "Large" appears three times in a description, the Unigram is only saved once, but the nfrequency value will be 3 for the three times that the Ngram appears in the line.

Step 2: Process a Set of Records Against the Ngram Index to Determine Matches

- Passes the new, incoming source data through the **Ngram Match** Widget
- Creates a set of Ngrams from the input source
- Checks for matches, using either an exact Ngram match and/or a Fuzzy Ngram match between the new data and previously created Ngram Cache. An Ngram is considered a match either when the Ngrams match exactly or if Fuzzy logic is used to create a fuzzy match.

- Presents the matches in the format required for the downstream application.

Create a DSA to process your input data to look for Ngram matches. The **Ngram Match** widget requires at least two inputs. These are an ID (also known as the request ID) and the text (description) for the Ngram Match, which are provided in this order. You may provide additional inputs. If you do so, refer to the input values using the name of the input node.

For example, if you have an input node called 'source', refer to it in your SQL statements as `&?source&`. If you want to refer to the standard inputs, a node called 'description', is referred to as `&?description&`.

The second parameter, the text, is analyzed for Ngrams, and matches made against the database using the Query SQL. Based on your Ngrams selections, the Application Studio matches unigrams, bigrams, and trigrams.

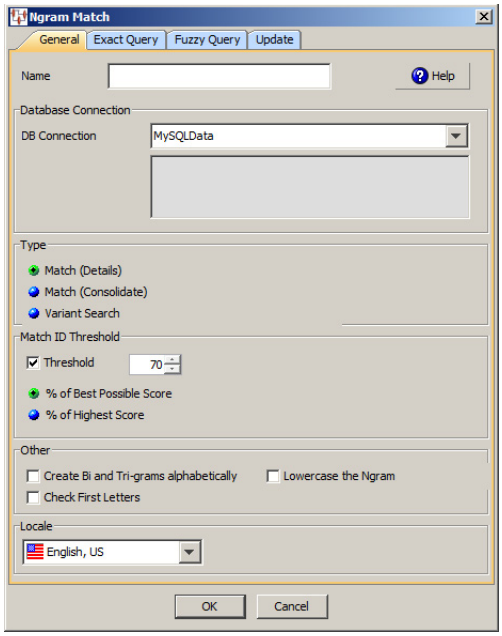
Then the matches are used with the Update SQL statements to insert the matched data into the update table.

The following table describes the keywords that are used and returned by the Ngram computation:

Keywords	Description
NGRAM	Ngram created from the input text.
DB_NGRAM	Ngram from the Ngram database cache that matched to the input Ngram. This could be a unigram, bigram or trigram.
NTYPE	1 - Unigram 2 - Bigram 3 - Trigram
MATCH_ID	ID value associated with the matching record.
MATCH_SCORE	Score of the matching record based on the Ngram weights selected in the **Exact Query** and/or the **Fuzzy Query** tabs of the Ngram Match widget.
RSINDEX	Soundex pattern with first character being the first letter of the Ngram.
MAX_SCORE	Max score of an individual record in the set of matching records; used with the **Match (Details)** type.
PCT_MAX_SCORE	Percent of an individual matching record score to the Max Score of the best scoring record within the set of matching records for a given input record; used with the **Match (Details)** type.
BEST_POSSIBLE	Best possible score for input record; used with the **Match (Consolidate)** type.
PCT_BEST_POSSIBLE	Percent of a matching record score to the Best Possible Score of Input Record; used with the **Match (Consolidate)** type.
COUNT	Count of Ngram variants; used with the **Variant Search** type.

The following sections describe how to use the **Ngram Match** dialog box.

General Tab



Name Enter a name for the widget.

DB Connection You must select the type of database connection that you want to use. The list of database connections is populated based on those that you are configured in the Oracle DataLens Server. If the type of database connection is not listed, you must configure it in the Oracle DataLens Server so that it is available for selection when creating Transformation Maps.

Type Section Select the type of Ngram matching process from the following:

Match (Details)
For a given input (Request ID), the output is the set of both exact and fuzzy Ngrams matches and the score for a matching record (Match ID) where *NGRAM* is the input and `DB_NGRAM` is the output match. The `Percent_Best_Possible` keyword is not populated.

Match (Consolidate)
For a given input (Request ID), the output is a single consolidated score for a matching Ngrams associated with the output record (Match ID) and NGRAM. The `NGRAM` and `DB_NGRAM` keywords are not output because it does not have any meaning in the context of a consolidated result. The `Percent_Best_Possible` keyword is populated.

Variant Search
The Query from the Fuzzy Query processes is expected to produce a count for a given variant from the Ngram Cache. The Ngram process inserts the variant count into the update table using the special keyword `&?COUNT&`. This greatly reduces the number of inserts into the update table.

Match ID Threshold Section Use this section to select Match ID record threshold to limit the number of records that are returned that are matched against the input record (Request ID).

Threshold
Allows you to specify a threshold percent that represents a cumulative score that the Ngram result must equal or exceed. This selection applies to any selection in the **Type** section.

% of Best Possible Score
Scores the input record then compares the consolidated score of all the matching records to determine if that matching record equals or exceeds the threshold value. For example, if the input record scores 100 and a matching record scores a 90 (90%) with a threshold of 85% then the matching record exceeds the threshold value and it appears in the results.

% of Highest Score
Scores all matching records and determines the highest scoring record. Only those matching records that equal or exceed the threshold value are included in the results. For example, if the highest scoring matching record scores 100 and the threshold is 85% then all matching records that equal or exceed the threshold value appear in the results.

Other Section Use this section to select the following additional options:

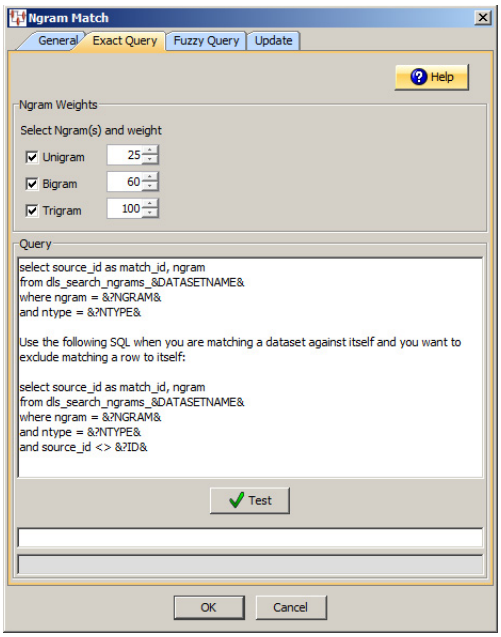
Create Bi and Tri-grams Alphabetically
Creates a bigrams and trigrams in alphabetical order.

Lowercase the Ngram
Creates Ngrams in lowercase.

Check First Letters
Rejects Ngrams when the first letter does not match the incoming word. It does not apply to Threshold 1.

Locale Section Select the language locale you want to use for Ngram matching.

Exact Query Tab
Click the **Exact Query** tab.



Ngram Weights Section Use this section to select the type of Ngram(s) and how you want them to be numerically weighted. The weights are used in the scoring process when one of Ngram of that type matches:

Unigram
Creates a unigram with the selected weight.

Bigram
Creates a bigram with the selected weight.

Trigram
Creates a trigram with the selected weight.

Query Section Use this section to set up your database query.

SQL Field
Use the field to construct your database query with standard SQL query statements and syntax.

The default SQL Query is similar to the following:

```
select source_id as match_id, ngram
from dls_search_ngrams_&DATASETNAME&
where ngram = &?NGRAM&
and ntype = &?NTYPE&
```

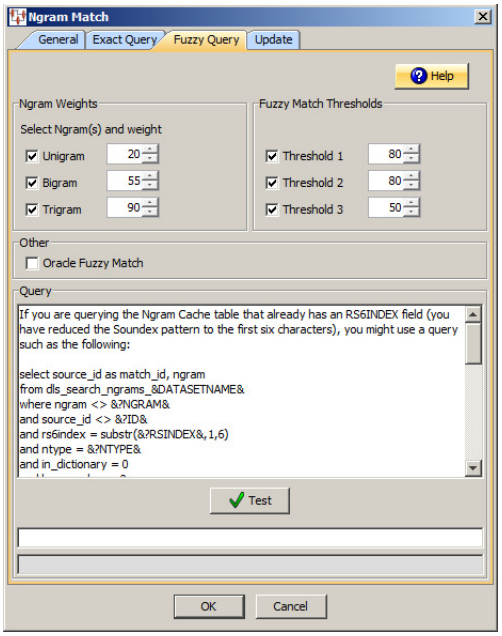
Use the following SQL when you are matching a data set against itself and you want to exclude matching a row to itself:

```
select source_id as match_id, ngram
from dls_search_ngrams_&DATASETNAME&
where ngram = &?NGRAM&
and ntype = &?NTYPE&
and source_id <> &?ID&
```

The `select` clause only returns the Match ID and the Ngram. It is irrelevant what the fields are named; the Match ID *must* be the first field in the `select` clause and Ngram the second field (anything else is ignored).

Test Button
Use this button to test your database connection.

Fuzzy Query Tab
Click the **Fuzzy Query** tab.



Ngram Weights Section Use this section to select the type of Ngram(s) and how you want them to be numerically weighted. The weights are used in the scoring process when one Ngram of that type matches:

Unigram
Creates a unigram with the selected weight.

Bigram
Creates a bigram with the selected weight.

Trigram
Creates a trigram with the selected weight.

Fuzzy Match Thresholds Section Use this section to select the level of fuzzy match thresholds, 1 through 3, and how you want them to be weighted:

If you are using fuzzy matching, you may want to filter the results from the database based on various different threshold values. The value computed for the Ngram (between 0 and 100) must be greater than or equal to every active threshold value. These settings are data dependent and must be tuned for maximum performance.

Note: Fuzzy matching is not used with Trigrams because it is not effective.

Note: Several of these threshold values are only computable if the selected Locale is English-based. Non-English locales default to a value of 100 and thus pass through the filter.

Other Section Use this section to select this additional option:

Oracle Fuzzy Match
Allows you to use fuzzy matching when connecting to an Oracle database. It automatically quotes each word in the Ngram with the appropriate backslash characters so that it is in an acceptable format for the Oracle database. The SQL statements that you provide in the **Query** section must configure the Oracle fuzzy matching functionality.

Note: The use of this option cannot be used in conjunction with the fuzzy matching threshold functionality of the Application Studio. The two fuzzy matching functions are mutually exclusive.

Note: To use Oracle Fuzzy, you must turn on the Oracle Text feature and create Fuzzy Indexes for each locale you want to search on. Consult with your Database Administrator for details.

Query Section Enter your query as follows:

SQL Field
Use the field to construct your database query with standard SQL query statements and syntax. For example, you could use the following SQL statements:

```
select source_id as match_id, ngram
from dls_search_ngrams_&DATASETNAME&
where ngram <> &?NGRAM&
and source_id <> &?ID&
and rs6index = substr(&?RSINDEX&,1,6)
and ntype = &?NTYPE&
and in_dictionary = 0
and has_number = 0
```

Ensure that the Match ID is the first variable in the `select` clause and the Ngram is the second variable; anything else is ignored. This `select` clause must be compatible with your database.

The following SQL statements are an example of a Variant Search match type:

```
select ngram, count(*) frequency from dls_search_ngrams_&DATASETNAME&
where ngram <> &?NGRAM&
and source_id <> &?ID&
and rs6index = substr(&?RSINDEX&,1,6)
and ntype = &?NTYPE&
and in_dictionary = 0
and has_number = 0
group by ngram
```

The following is an example of the preceding query that incorporates Oracle fuzzy matching:

```
select match_id from TABLE where CONTAINS(ngram, 'fuzzy(('||&?NGRAM&||'), 65, 20,
weight)', 1) > 0 and &?NTYPE& = ntype
```

It is important to recognize the following:

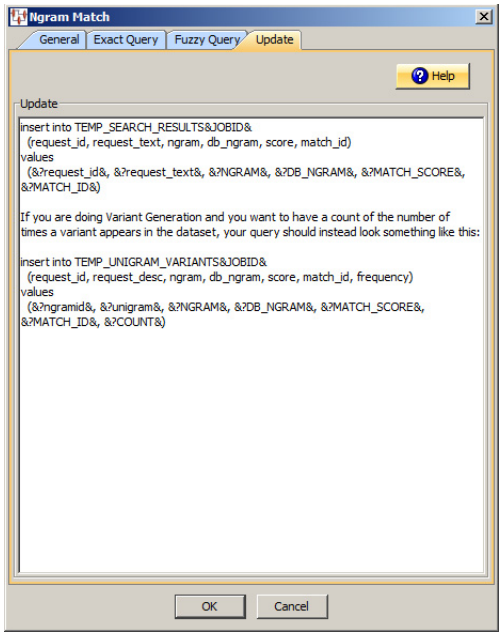
- The second parameter to `CONTAINS` is a string, and the `||` concatenates are being used to concatenate the first part of the fuzzy function with the Ngram and the last part of the fuzzy function. The `and` surrounding the Ngram are needed so that the fuzzy function is able to handle Bigrams and Trigrams. For example, if there is a phrase rather than a single word.
- Each word in the Ngram phrase *must* be escaped since it is possible that the word is an Oracle keyword. If you use Oracle fuzzy matching, the Application Studio escapes each word appropriately.

Test Button

Use this button to test your database connection.

Update Tab

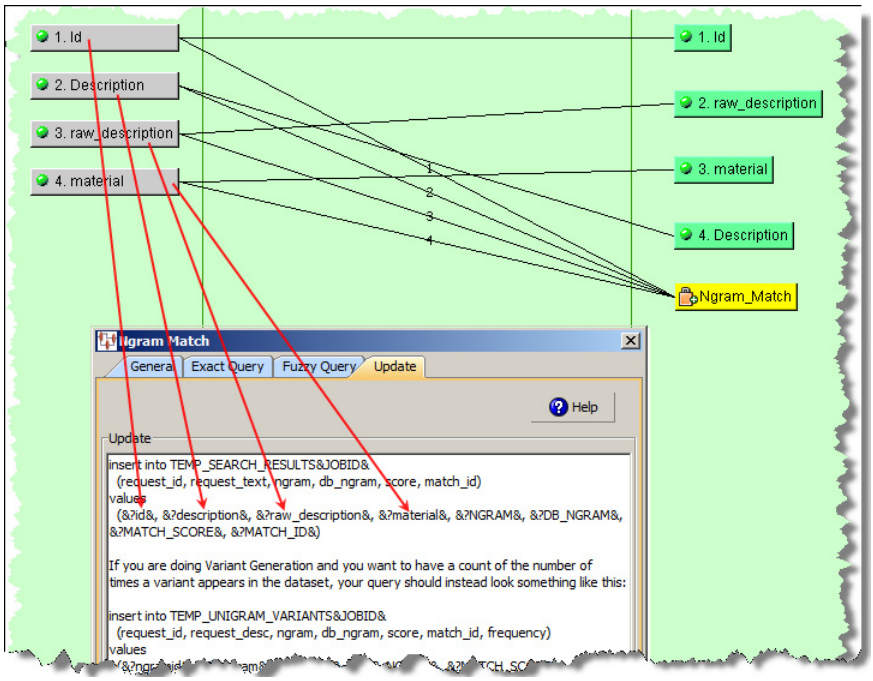
Click the **Update** tab. This tab allows you to capture the results of the Ngram Match process by inserting those results into a database table.



Update Field

Use this field to construct your database query with standard SQL query statements and syntax.

You can insert data from either an input node or the results from an Ngram computation. For data from an input node, you specify the value with the name of the node surrounded by `&?` and `&`. For example if your input node is called `id`, you refer to it as `&?id&` in the `values` clause. The input node data must appear before the computed keyword values as in the following example:



For example, you could use the following SQL statements:

```
insert into TEMP_SEARCH_RESULTS&JOBID&
(request_id, request_desc, raw_desc, ngram, db_ngram, score, match_id)
values
(&?id&, &?description&, &?raw_description&, &?material&, &?NGRAM&, &?DB_NGRAM&,
&?MATCH_SCORE&, &?MATCH_ID&)
```

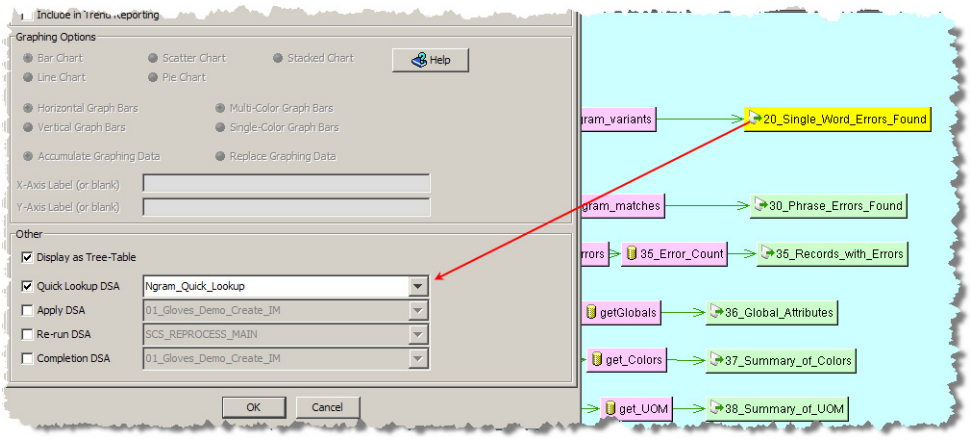
The following SQL statements are an example of a Variant Search match type:

```
insert into TEMP_UNIGRAM_VARIANTS&JOBID&
(request_id, request_desc, ngram, db_ngram, score, match_id, frequency)
values
(&?ngramid&, &?unigram&, &?NGRAM&, &?DB_NGRAM&, &?MATCH_SCORE&, &?MATCH_ID&,
&?COUNT&)
```

You can insert whatever data you want into the update table as long as the data is data from either an Input node, or results from the Ngram computation. For data from an Input node, you specify the value with the name of the node surrounded by `&?` and `&`. For example, if your input node is called `request_id`, you refer to it as `&?request_id&` in the `value` clause.

> **Note:** The database field names do not have to match the input node names.

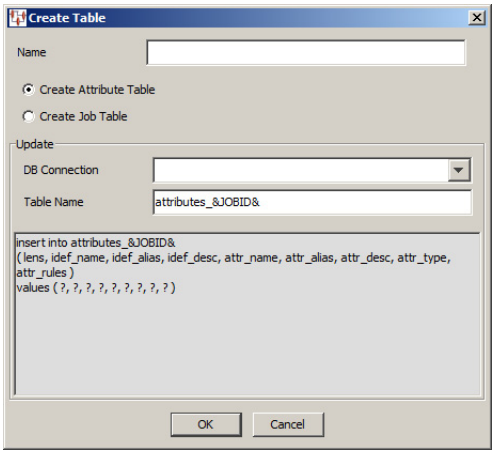
If your Ngram processing DSA is of the variant search matching type, you must configure it as a Quick Lookup DSA for the associated Output step of the main DSA as in the following example:



For more information, see "Using the Quick Lookup DSA Option" on page 6-39.

Using the Create Table Widget

The Oracle DataLens Server repository contains statistical information regarding each job run for each DSA and attribute information related to the DSA. It may be advantageous to retrieve this type of information for reporting purposes. The **Create Table** widget can be used to retrieve this information and create a table using a new table you create with this widget or an existing table. This widget is only active in Transformation Maps that use database inputs.



Enter a name for the **Create Table** widget. Select the type of data you want to retrieve job statistics or attribute information.

Use the **Update** section to select the type of database connection and enter your query as follows:

DB Connection

You must select the type of database connection that you want to use. The list of database connections is populated based on those that you are configured in the Oracle DataLens Server. If the type of database connection is not listed, you must configure it in the Oracle DataLens Server so that it is available for selection when creating Transformation Maps.

Table Name

Enter the name of the table into which you want to insert data. The `insert` clause in the SQL field is automatically updated with this table name to avoid errors.

SQL Field

Use the field to construct your database query with standard SQL query statements and syntax to create a table or insert the information into an existing table.

For example, you could use the following SQL statements to insert the information retrieved into the existing table, `attributes_&JOBID&`:

```
insert into attributes_&JOBID&
( lens, idef_name, idef_alias, idef_desc, attr_name, attr_alias, attr_desc, attr_
type, attr_rules )
values ( ?, ?, ?, ?, ?, ?, ?, ?, ? )
```

This `insert` clause must be compatible with your database. Optionally, you can use a question mark (`?`) in the `insert` clause. At run-time, the question mark character is replaced with the transformation input data. This allows you to create a database transformation that varies with the content of the record being processed. Additionally, when database transforms are used to aggregate data fields from several different data sources, common access key information can be used across all data sources.

Using Secondary DSAs

DSA output steps can be associated with specific secondary DSAs to further process output data within the Governance Studio. The secondary DSA allows you to interact with the data in an environment that operates like an application. There are three types of dedicated secondary DSAs that all receive text output from a Governance Studio output step as follows:

Apply DSAs

Results have been reviewed and approved by a user to continue downstream processing. The results could be inserted into a database table or delivered using optional methods like email, file, etc.

Re-Run DSAs

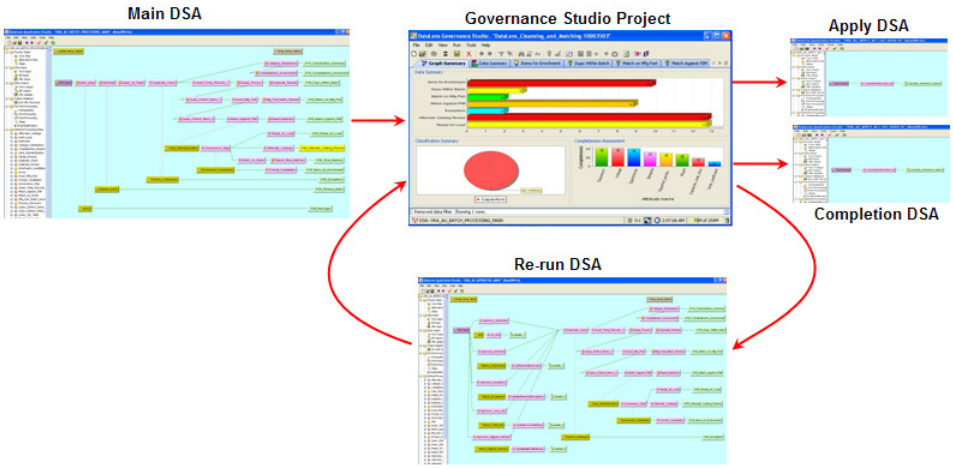
Results require continued processing within the application. The results are returned to the calling Governance Studio project and merged into existing steps. Since the results of the Re-Run DSA are expected back in the calling Governance Studio project, the output steps from the calling application *must* coincide (name and structure) with the Re-Run DSA.

Quick Lookup DSAs

These DSAs are specialized and used in the Ngram process to distinguish a variant in the context of the original description. They are considered to be Re-Run DSAs. For more information, see "Matching Ngrams" on page 6-22.

Completion DSAs

Completes the current Governance Studio job, saves the state of the data, and then makes the Governance Studio project read only.

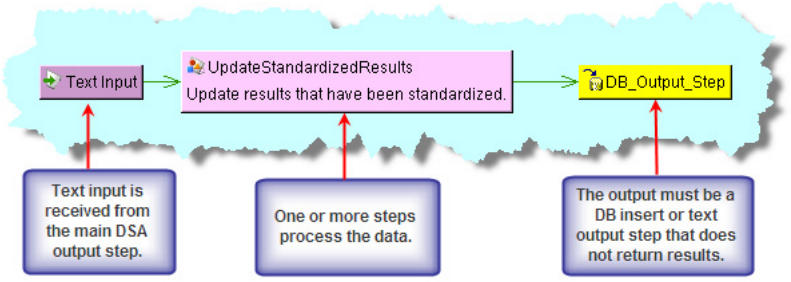


You can use one or all of the secondary DSA processing types and change output steps at any time.

The following sections describe how to implement the use of secondary DSAs from a main DSA.

Using the Apply DSA Option

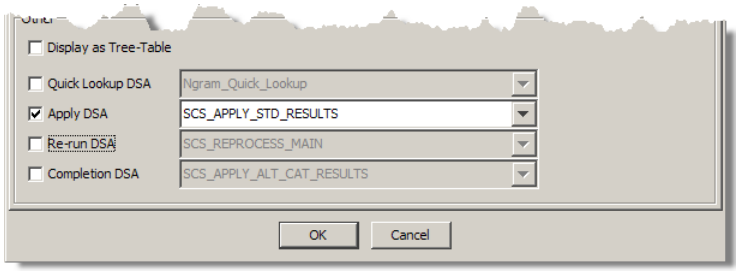
1. Create the DSA that you want to apply as a secondary DSA.
2. Open the main DSA.
3. Edit the output step to add secondary DSA processing. For more information about editing output nodes, see "Editing a Text Output Step" on page 2-8.
4. Ensure that the input columns of the Apply DSA correspond to the output columns of the main DSA as in the following example:



Note: If your output step is text, ensure that the **Do NOT return results to caller** check box is selected so that results are not returned to the DSA erroneously.

5. Select the **DataLens Governance Studio** tab.

6. Select the **Apply DSA** check box to activate the list of deployed DSAs that have an equal or greater number of columns as the selected output step.



7. Select the DSA that you want applied when the **Apply DSA** function is used in the Governance Studio project.

> **Note:** If the DSA you want to use is not listed, you must exit this dialog, check the DSA in, and then return to this section. Only DSAs that are checked-in to the Oracle DataLens Server will appear in this list.

8. Click **OK** to complete the configuration of the Apply secondary DSA.
9. Check-In the DSA so that the changes are available in the Governance Studio project.

Using the Re-Run DSA Option

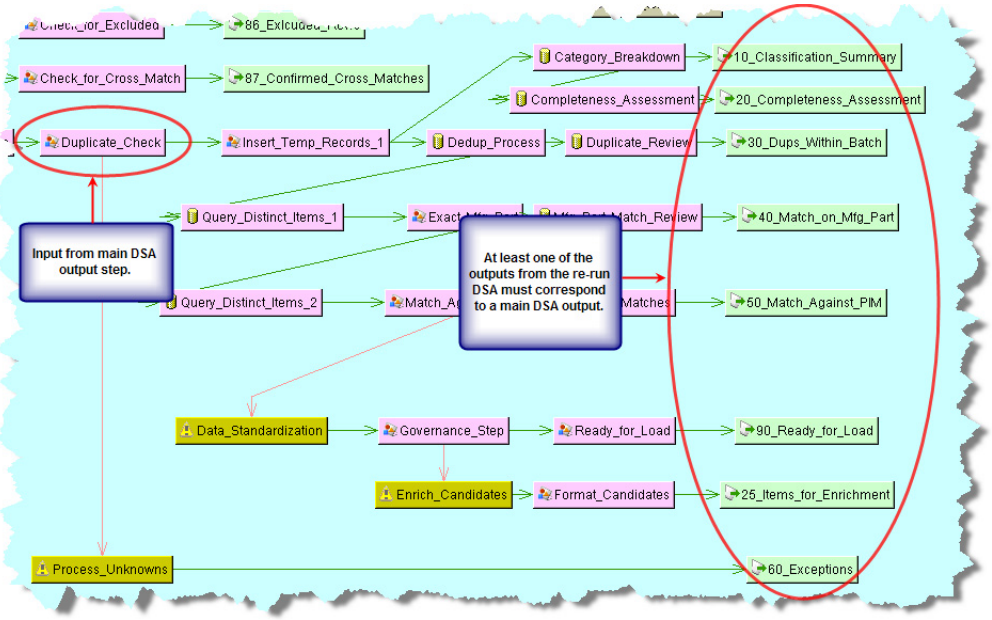
An important feature of using Re-run DSAs is that the results are automatically returned to the main (or calling) DSA. The outputs from the Re-run DSA are distributed amongst all the outputs of the main DSA whose names match. The results of the Re-run are *merged* with the existing results of the calling DSA.

Additionally, the rows selected for processing by the Re-Run DSA are automatically deleted from the main DSA output step.

> **Note:** All data selected for re-run is included in a re-run operation so all of the source data output nodes from the calling DSA is sent to the secondary DSA for processing. The secondary DSA must have at least the same number of input nodes to receive the data. On a matching output tab, both the upper and lower task pane columns are output.

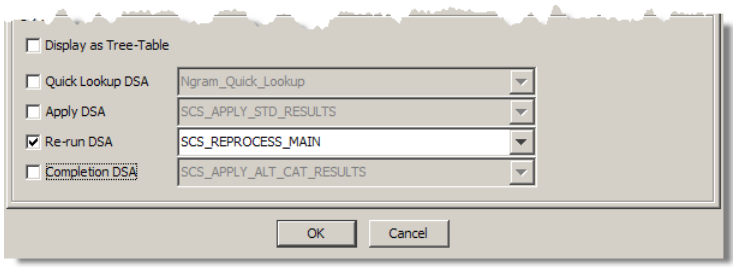
Note: If you are using a long running temporary table, the row(s) must be deleted from that table in the Re-run DSA to avoid double counting.

1. Create the DSA that you want re-run as a secondary DSA.
2. Open the main DSA.
3. Edit the output step to add secondary DSA processing. For more information about editing output nodes, see "Editing a Text Output Step" on page 2-8.
4. Ensure that one or more output columns of the Re-run DSA correspond to the output columns of the main DSA as in the following example:



5. Select the **DataLens Governance Studio** tab.

6. Select the **Re-run DSA** check box to activate the list of deployed DSAs that have an equal or greater number of columns as the selected output step.



7. Select the DSA that you want applied when the **Re-run DSA** function is used in the Governance Studio project.

> **Note:** If the DSA you want to use is not listed, you must exit this dialog, check the DSA in, and then return to this section. Only DSAs that are checked-in to the Oracle DataLens Server will appear in this list.

8. Click **OK** to complete the configuration of the Re-run secondary DSA.
9. Check-In the DSA so that the changes are available in the Governance Studio project.

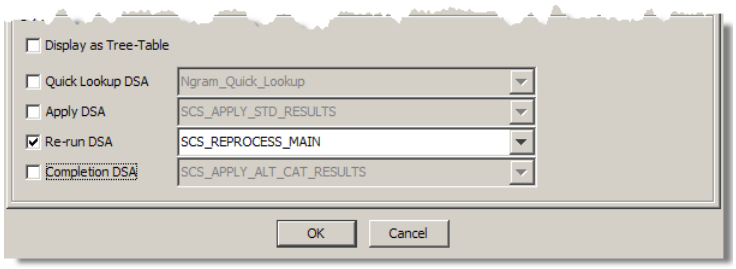
Using the Quick Lookup DSA Option

A Quick Lookup DSA is used in the output step of an Ngram matching DSA to show the variants that exist in the data for a given Ngram. The results from a Quick Lookup DSA are displayed in the Governance Studio on the Output tab for the Ngram output step.

1. Create an Ngram matching DSA that you want use as a quick lookup DSA.

   For more information, see "Matching Ngrams" on page 6-22.
2. Open the main DSA.
3. Edit the output step to add Quick Lookup DSA processing. For more information about editing output nodes, see "Editing a Text Output Step" on page 2-8.
4. Select the **DataLens Governance Studio** tab.

5. Select the **Quick Lookup DSA** check box to activate the list of deployed DSAs that have an equal or greater number of columns as the selected output step.



6. Select the DSA that you want applied when the **Quick Lookup DSA** function is used in the Governance Studio project.

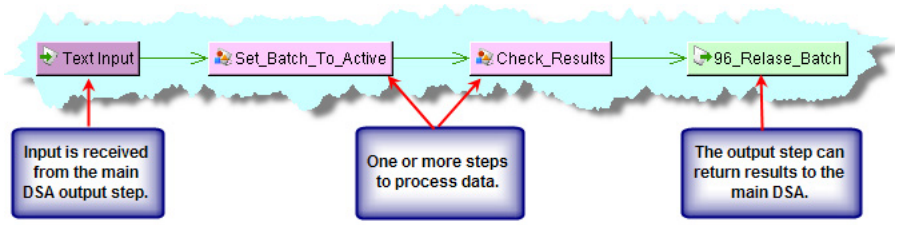
Note: If the DSA you want to use is not listed, you must exit this dialog, check the DSA in, and then return to this section. Only DSAs that are checked-in to the Oracle DataLens Server will appear in this list.

---

7. Click **OK** to complete the configuration of the Quick Lookup secondary DSA.
8. Check-In the DSA so that the changes are available in the Governance Studio project.

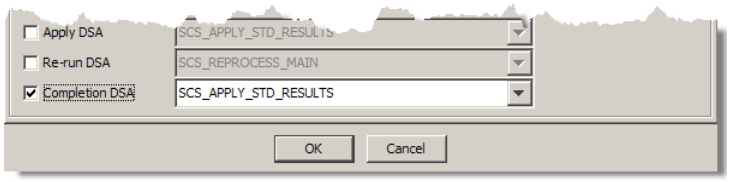
Using the Completion DSA Option

1. Create the DSA that you want re-run as a secondary DSA.
2. Open the main DSA.
3. Edit the output step to add secondary DSA processing. For more information about editing output nodes, see "Editing a Text Output Step" on page 2-8.
4. Ensure that the input columns of the completion DSA coincide with the output columns of the main DSA as in the following example:



5. Select the **DataLens Governance Studio** tab.

6. Select the **Completion DSA** check box to activate the list of deployed DSAs that have an equal or greater number of columns as the selected output step.



7. Select the DSA that you want applied when the **Completion DSA** function is used in the Governance Studio project.
8. Click **OK** to complete the configuration of the Completion secondary DSA.
9. Check-In the DSA so that the changes are available in the Governance Studio project.

Using Stored Database Procedures and Functions

Stored procedures and functions are blocks of code, stored and executed on a database, which extend the SQL language with procedural functionality and can be shared across the database. A DSA using stored database procedures and functions can benefit from the added functionality and increased performance. In addition, the use of stored procedures and functions can aggregate common functionality into a single code block that can be shared across DSAs ensuring standard database practices.

DSAs support stored database procedures and functions in the, Pre-Processing and Post-Processing steps. For information about how to use these steps, see "Pre-Post Processing Folder" on page 2-17.

Additionally, this functionality is supported in the DB Transform and DB Update transformations in Transformation Maps. For information about how to use these transformations, see "Defining DB Transforms" on page 4-36.

For more information about the support provided for database subprograms, see the vendor documentation supplied with you database software.

Note: The JDBC array interface *does not* support calling database stored procedures o functions in a batch; all stored procedures and functions are executed serially.

The support for this functionality, by database vendor, is as follows:

Database Vendor	Stored Functions Supported	Stored Procedures Supported
Oracle 9i and later	Y	Y
SQL Server 2005	N	Y
MySQL (Version >= 5)	Y	Y

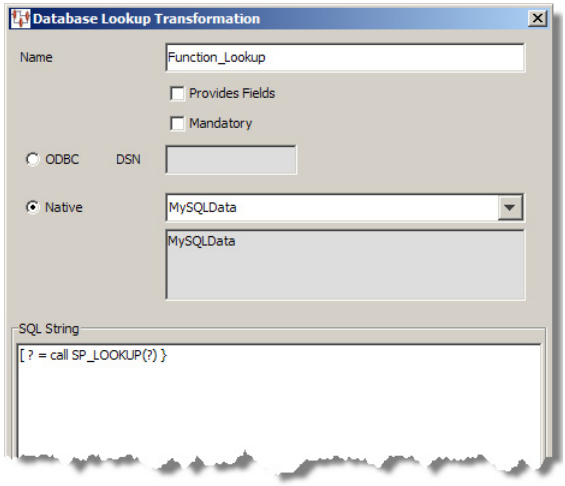
Using Stored Functions

Stored functions are only supported by the DB Transformation node. The function return type must be a string or automatically converted to a string by your database

vendor. When you create or edit either of the supported transformations, you must use the following stored function calling syntax:

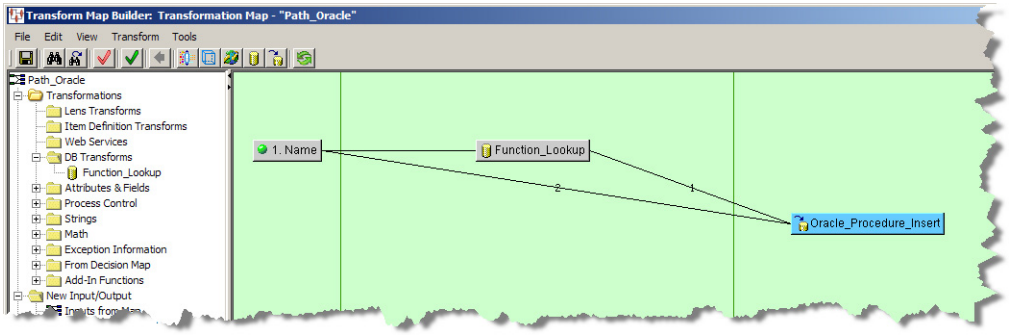
```
{ ? = call function_name(parameters) }
```

This function call is entered into the SQL String field of the Database Lookup Transformation or Database Update Definition dialog box as in the following example.



The **Provide Fields** check box cannot be used with stored functions. Selecting the **Mandatory** check box ensures that an exception is raised if a null value is returned.

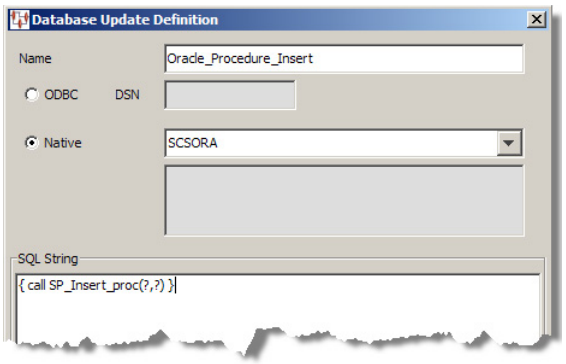
You use the returned value by connecting the DB node to the next node or widget in the map as in the following example Transformation Map.



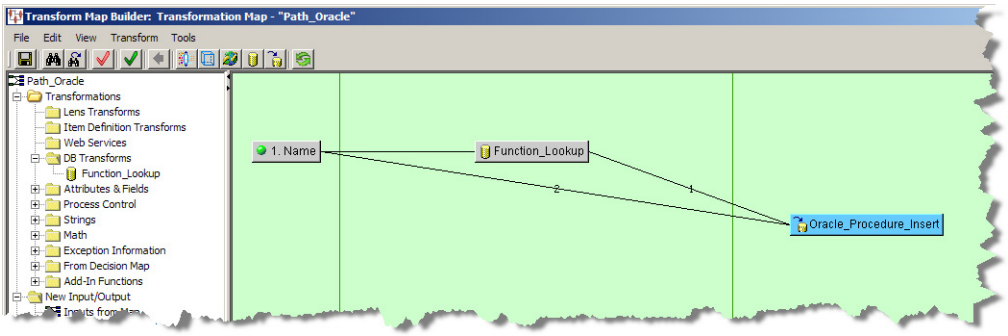
Using Stored Procedures

Stored procedures are only supported by the DB Update Transformation Map node and the Pre-Processing and Post-Processing DSA steps. When you create or edit any of the supported steps or transformations, you must use the following stored procedure calling syntax:

```
{ call procedure_name(parameters) }
```



You use the returned value by connecting the DSA step to the next step or the DB node to the next node or widget in the map as in the following example Transformation Map.



Note: The Oracle Product Data Quality version 5.5 and greater release does not support stored procedure out parameters and ignores any values returned from the procedure.

Transaction Handling

For performance and concurrence reasons, a stored subprogram *must not* issue a commit or rollback unless your database vendor supports and the stored subprogram is coded to use true nested transactions. For example, the Oracle autonomous transaction. A true nested transaction operates outside of the parent transaction; commit or rollback operations of the nested transaction do not affect the parent transaction.

A

Installing the Client Software

Oracle Product Data Quality uses a concept called Java Web Start to initially install and maintain the current version of the software on your client desktop. The process requires you to access the Oracle DataLens Server to initiate the connection and download the software.

The Oracle Product Data Quality client applications downloaded and installed using Java Web Start by browsing to the installation page for your Oracle DataLens Server as follows:

1. Ensure that you have the Java SE Runtime Environment (JRE) 6 Update 21 installed. You can download the JRE and obtain the installation instructions by browsing to:

   `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

2. Start Microsoft Internet Explorer.

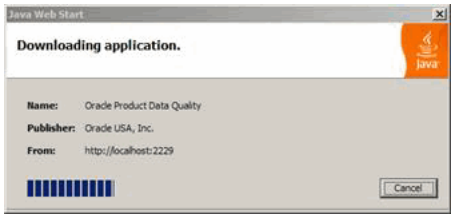
3. Initiate a connection and download the client software by browsing to:

   `http://`*`server`*`:2229/datalens/datalens.jnlp`

   Where *server* is the hostname of the Oracle DataLens Server.

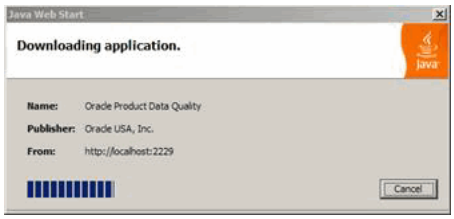
   Note: If you have setup a different port number for your application server other than 2229, you must use that port number in the following URL when browsing to the Oracle DataLens Server to download the client applications.

   The application download and verification begins.



   Note: If you receive a **File Download** message indicating that the `.jnlp` file is not associated with a program, you do not have the supported JRE installed. Click **Cancel** and return to Step

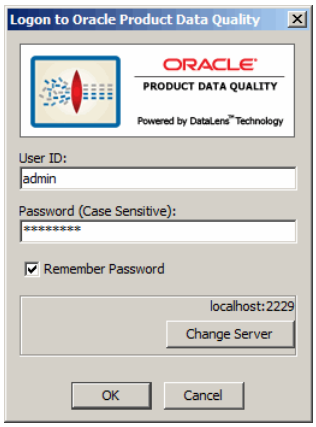
After the verification completes, the installation begins. Oracle Product Data Quality files are digitally signed by a trusted source so the following security warning is displayed:



Tip: To avoid this security dialog in the future, select the **Always trust content from this publisher** check box.

4. Click **Run** to continue and complete the installation.

   The Oracle Product Data Quality log on dialog is displayed.



B

Regular Expressions

Regular Expressions use character pattern matching to find and capture the information you need. Regular Expressions are used most frequently in the Knowledge Studio when creating Terminology rules.

To use Regular Expressions, you must learn the syntax. Regular Expressions use special characters, wildcards, to match a range of other characters. A Regular Expression found in a Terminology rule is surrounded by forward slashes.

Special Characters in Regular Expressions

The following table lists of many of the special characters used in a regular expression and some example expressions:

Wildcard or Meta-Characters	Description and Examples
.	The dot character matches any single character. For example, the terminology rule regular expression, "/a.b/", matches all text where there is an "a" followed by any single character, followed by a "b", as in, "a5b".
*	The asterisk matches the preceding pattern or character zero or more times. For example, "/fo*/" matches the following text fragments: "f", "fo", "foo", "fooo" Combining the period and asterisk, "/a.*b/" will match "a5b", "a55b", "a123b", and so on.
+	The plus sign matches the preceding pattern or character one or more times. For example, /ca+r/ matches the following text fragments: "car", "caar" and "caaar", but will not match "cr".
?	The question mark character matches the preceding pattern or character zero or once. For example, "/ca?r/" matches both "car" and "cr"; it will not match "caar".
{n}	The curly brackets are used to match exactly n instances of the proceeding character or pattern. For example, "/x{2}/" matches "xx". **Note:** The curly brackets are used in the application to differentiate white space bounded text or characters from text or characters that are embedded among other characters with no identifiable white space.

Wildcard or Meta-Characters	Description and Examples
{n,m}	This form of the curly brackets is used to match the preceding character or pattern from n to m times, with n greater than m. If m is not present then the pattern is matched n or more times. For example, "/x{2,3}/" matches "xx" and "xxx".
[…]	The square brackets match any one of characters inside the brackets. A range of characters in the alphabet can be matched using the hyphen. For example, "/[xyz]/ " will match any of "x", "y", or "z". Also, "/[xyz]+/" will match "x", "xx", "y", "yy", and so on. Within square brackets, a range of characters can be defined using the dash (-). For example, "[a-z]" matches any lowercase letter, and "[A-Z]" matches any uppercase letter. When using the dash to define a range of characters, the first character must precede the second character in alphabetic or numeric order. For example, "[0-9]" is valid, but "[9-0]" is not valid.
(…)	The parentheses are used to group characters. For example, "(cars?)|bus" will match "car", "cars", or "bus". **Note:** The parentheses are equivalent to "(?:…)"
x|y	The pipe (|) character matches either "x" or "y", where "x" or "y" are blocks of characters. For example, "car|bus" will match either "car" or "bus".
\	Backslash has two meanings: Matches against characters that normally have special meaning such as star (*) and dot (.), see preceding descriptions. In this case a "*" matches the star character. Similarly "" matches the dot character. Used to define a meta-character. The character "w" will normally match "w". A "\w" will match a sequence of alphanumeric characters not interrupted by white space, see the following description.
\w	Matches any alphanumeric character or the underscore. This is identical to "[A-Za-z0-9_]".
\W	Matches any character that is not alphanumeric and not underscore.
\d	Matches all digits. Identical to "[0-9]". For example, "/\d+/" will match one or more digits. For example, positive integers.
\D	Matches all non-digits including white space.
\s	Matches any white space character including a tab or a space.
\S	Matches any character other than white space characters.
(?i)	The "(?i)" meta-characters indicate that the following pattern should ignore the case of letters when performing the match. For example, the pattern "(?i)car" will match "Car", "car", "cAR", and so on. And "(?i)cars?" will match "Car", "Cars", "CarS", and so on. **Note:** The syntax differences between this match rule and the following three are where the pattern is inside the parentheses.
(?!pattern1)pattern 2	The "(?!…)… meta-characters say that if the first pattern is not present, pattern1, then accept the second pattern, pattern2. For example, /(?!x)car/ matches "car"; it will not match "xcar". **Note:** Both pattern1 and pattern2 are required.

Useful Regular Expressions in Terminology Rules

Year

```
[year] case=insensitive
        /[12][0-9][0-9][0-9]/
```

Zip Code

```
[zip] case=insensitive
        /\d\d\d\d\d/    # 5 digit zip code
        /\d\d\d\d\d-\d\d\d\d/   # 9 digit zip code
```

First Name

```
[first_name] case=insensitive
        ([A-Za-z]+)     # a name surrounded by white space
```

Street Name

```
[street_name] case=insensitive
      [street_name]
      ((?!(?i)rd |dr |ave |ln |ct |st )[A-Za-z]+)     # street name not starting with dr, ave, ln, ...
      (\d[Rr][Dd])            # 3rd, ...
      (\d\d[Rr][Dd])          # 23rd, ...
      (\d\d\d[Rr][Dd])        # 103rd, ...
      (\d[tT][hH])            # 5th, ...
      (\d\d[tT][hH])          # 25th, ...
      (\d\d\d[tT][hH])        # 105th, ...
      (\d[sS][tT])            # 1st, ...
```

For more information on regular expressions, consult *Perl for Dummies*, by Paul Hoffman, or *Mastering Regular Expressions*, by Jeffrey Friedl.