Informatica

Informatica®

10.5

Developer Workflow Guide

Informatica Developer Workflow Guide
10.5
March 2021

Publication Date: 2021-06-14

Table of Contents

Preface

Use the *Informatica® Developer Workflow Guide* to learn how to create, run, and administer workflows. Understand workflow concepts so you can run mappings and other tasks in a single operation. Learn how to recover a workflow if it is interrupted or if an error occurs.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Workflows

This chapter includes the following topics:

Workflows Overview

A workflow is a graphical representation of a set of events, tasks, and decisions that define a business process. Use the Developer tool to create a workflow and to save the workflow to the Model repository. You deploy the workflow as application to the Data Integration Service.

You create a workflow in the Developer tool. You add objects to the workflow and connect the objects with sequence flows. The Data Integration Service uses the instructions that you configure in the workflow to run the objects.

A workflow object is an event, task, or gateway. An event starts or ends the workflow. A task is an activity that runs a single unit of work in the workflow, such as running a mapping, sending an email, or running a shell command. A gateway defines a decision to split or merge the data paths in the workflow.

A sequence flow connects one workflow object to another. Sequence flows specify the order in which the Data Integration Service runs the objects. You can define a condition on a sequence flow to determine whether the Data Integration Service runs the object that the sequence flow specifies.

You can define and use workflow variables and parameters to make workflows more flexible. A workflow variable represents a value that records run-time information and that can change during a workflow run. A workflow parameter represents a constant value that you define before you run a workflow. You use workflow variables and parameters in conditional sequence flows and object fields. You also use workflow variables and parameters to pass data between a task and the workflow.

You can configure a workflow for recovery so that you can complete an interrupted workflow instance. A running workflow instance can be interrupted when an error occurs, when you cancel the workflow instance, or when a service process shuts down unexpectedly. You cannot recover an aborted workflow.

After you deploy a workflow, you run an instance of the workflow from the deployed application using the infacmd wfs command line program. You monitor the workflow instance run in the Monitoring tool.

Developing a Workflow

To develop a workflow, you select the objects to run in the workflow and you connect the objects with sequence flows. You can use gateways to split or merge the sequence flows based on conditions that you define.

1. Create a workflow.
2. Add objects to the workflow and configure the object properties.
3. Connect objects with sequence flows to specify the order in which the Data Integration Service runs the objects.
4. Define variables for the workflow to capture run-time information. Use the workflow variables in conditional sequence flows and object fields.
5. Define parameters for the workflow so that you can change parameter values each time you run a workflow. Use the workflow parameters in conditional sequence flows and object fields.
6. Optionally, configure the workflow for recovery.
7. Validate the workflow to identify any error that the workflow might contain. Fix any error that you find.
8. Add the workflow to an application and deploy the application to the Data Integration Service.

After you deploy a workflow, you run an instance of the workflow from the deployed application using the infacmd wfs command line program. You monitor the workflow instance run in the Monitoring tool.

Workflow Objects

A workflow object is an event, task, or gateway. You add objects as you develop a worklow in the editor. Workflow objects are non-reusable. The Developer tool stores the objects within the workflow only.

Events

Events define the start and the end of the activity that the workflow specifies for the data. Each workflow has a Start event and an End event. Optionally, a workflow can include one or more Terminate events.

The Developer tool gives each event a default name of Start_Event, End_Event, or Terminate_Event. You can rename and add a description to an event in the event properties.

The following table describes all events that you can add to a workflow:

Event	Description
Start	Defines the beginning of the workflow. The Start event represents the start the sequence of possible actions that the workflow specifies for the data. A workflow contains a single Start event.
End	Defines the end of the workflow. The End event represents the completion of the sequence of possible actions that the workflow specifies for the data. A workflow contains a single End event.
Terminate	Defines a point before the End event at which the workflow can end. A workflow terminates if you connect the task to a Terminate event and the task output satisfies a condition on the sequence flow. The Terminate event aborts the workflow before any further task in the workflow can run. A workflow can have one or more Terminate events.

Note: A Human task also contains a Start event and an End event. The events create a sequence flow for one or more steps in the Human task. A Human task cannot contain a Terminate event.

Tasks

A task in an activity that runs a single unit of work in the workflow, such as running a mapping, sending an email, or running a shell command. A task represents something that is performed during the workflow. The editor displays tasks as squares.

The following table describes all tasks that you can add to a workflow:

Task	Description
Assignment	Assigns a value to a user-defined workflow variable.
Command	Runs a single shell command or starts an external executable program.
Human	Defines actions that one or more users perform on the workflow data. Create a Human task when you want Analyst tool users to analyze the output of a mapping that runs in a Mapping task.
Mapping	Runs a mapping.
Notification	Sends an email notification to specified recipients.
Voting	Not active. Analyst tool users run approval workflows to approve or reject the assets in a business glossary. Analyst tool users do not interact with the workflow features of the Developer tool to configure or run approval workflows.

A workflow can contain multiple tasks of the same task type.

The Developer tool gives each task a default name of `<task type>_Task`, for example `Command_Task`. When you add another task of the same type to the same workflow, the Developer tool appends an integer to the default name, for example `Command_Task1`. You can rename and add a description to a task in the task general properties.

Gateways

A gateway splits a sequence flow into multiple branches or merges multiple branches into a sequence flow. Gateways operate in pairs. One gateway defines the branches that the workflow data can follow. Another gateway restores the branches to a single sequence flow. You can add a condition to the sequence flow that starts each branch. The Data Integration Service uses the condition values to identify the branches that the workflow data follows.

Exclusive Gateways and Inclusive Gateways

You can add Exclusive gateways and Inclusive gateways to a workflow. You add gateways to a workflow in pairs. Add Exclusive gateways when the workflow data must follow a single branch between the gateways. Add Inclusive gateways when the workflow data might follow multiple branches between the gateways. When you add Inclusive gateways, the Data Integration Service runs the objects on each branch in parallel.

The Developer tool gives a gateway the default name of `Exclusive_Gateway` or `Inclusive_Gateway`. When you add another gateway to the same workflow, the Developer tool appends an integer to the default name, for example `Exclusive_Gateway1`. You can rename and add a description to a gateway in the gateway properties

Related Topics:

- "Exclusive Gateways" on page 118
- "Inclusive Gateways" on page 119

Sequence Flows

A sequence flow connects workflow objects to specify the order that the Data Integration Service runs the objects. The editor displays sequence flows as arrows. You can create conditional sequence flows to determine whether the Data Integration Service runs the next object.

You cannot use sequence flows to create loops. Each sequence flow can run one time.

The number of incoming and outgoing sequence flows that an object can have depends on the object type:

Events

A Start event must have a single outgoing sequence flow. An End event must have a single incoming sequence flow.

Tasks

Tasks must have a single incoming sequence flow and a single outgoing sequence flow.

Gateways

Gateways must have either multiple incoming sequence flows or multiple outgoing sequence flows, but not both. Use multiple outgoing sequence flows from a gateway to split a workflow. Use multiple incoming sequence flows to a gateway to merge multiple branches into a single flow.

When you connect objects, the Developer tool gives the sequence flow a default name. The Developer tool names sequence flows using the following format:

```
<originating object name>_to_<ending object name>
```

If you create a conditional sequence flow, you might want to rename the sequence flow to indicate the conditional expression. For example, if a conditional sequence flow from a Mapping task to a Command task includes a condition that checks if the Mapping task ran successfully, you might want to rename the sequence flow to MappingSucceeded. You can rename and add a description to a sequence flow in the sequence flow general properties.

Conditional Sequence Flows

Create a conditional sequence flow to determine whether the Data Integration Service runs the next object in the workflow.

A conditional sequence flow includes an expression that the Data Integration Service evaluates to true or false. The expression must return either a boolean or an integer value. If an expression returns an integer value, any non-zero value is the equivalent of true. A value of zero (0) is the equivalent of false.

If the expression evaluates to true, the Data Integration Service runs the next object. If the expression evaluates to false, the Data Integration Service does not run the next object. If you do not specify a condition in a sequence flow, the Data Integration Service runs the next object by default.

When an expression in a conditional sequence flow evaluates to false, the Data Integration Service does not run the next object or any of the subsequent objects in that branch. When you monitor the workflow, the Monitoring tool does not list objects that do not run in the workflow. When a workflow includes objects that do not run, the workflow can still complete successfully.

You cannot create a conditional sequence flow from the Start event to the next object in the workflow or from the last object in the workflow to the End event.

Failed Tasks and Conditional Sequence Flows

By default, the Data Integration Service continues to run subsequent objects in a workflow after a task fails. To stop running subsequent workflow objects after a task fails, use a conditional sequence flow that checks if the previous task succeeds.

You can use a conditional sequence flow to check if a Mapping, Command, Notification, or Human task succeeds. These tasks return an Is Successful general output. The Is Successful output contains true if the task ran successfully, or it contains false if the task failed. Create a boolean workflow variable that captures the Is Successful output returned by a task. Then, create an expression in the outgoing conditional sequence flow that checks if the variable value is true.

For example, you create a boolean workflow variable that captures the Is Successful output returned by a Mapping task. You create the following expression in the conditional sequence flow that connects the Mapping task to the next task in the workflow:

```
$var:MappingTaskSuccessful = true
```

If the Mapping task fails, the expression evaluates to false and the Data Integration Service stops running any subsequent workflow objects.

Related Topics:

- "Task Recovery Strategies" on page 127
- "Task Output" on page 27

Terminate Events and Conditional Sequence Flows

You can use a conditional sequence flow to connect a task to a Terminate event. If the output from the task satisfies the condition on the sequence flow, the workflow reaches the Terminate event and the workflow ends. A workflow that ends at a Terminate event enters an aborted state.

Parameters and Variables in Conditional Sequence Flows

You can include workflow parameters and variables in an expression for a conditional sequence flow.

You can select a workflow parameter or variable in the **Condition** tab, or you can type the parameter or variable name in the conditional expression using the required syntax.

For example, you create a workflow variable that captures the number of rows written to the target by a mapping run by a Mapping task. You create the following expression in the conditional sequence flow that connects the Mapping task to a Command task:

```
$var:TargetRowsMapping > 500
```

The Data Integration Service runs the Command task if the mapping wrote more than 500 rows to the target.

Related Topics:

- "Parameter Names in Expressions and Strings" on page 40
- "Variable Names in Expressions and Strings" on page 33

Creating a Workflow

When you create a workflow, the Developer tool adds a Start event and an End event to the workflow.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File** > **New** > **Workflow**.

   The Developer tool gives the workflow a default name.
3. Optionally, edit the default workflow name.
4. Click **Finish**.

   A workflow with a Start event and an End event appears in the editor.

Adding Objects to a Workflow

Add the tasks and gateways that you want to run in the workflow. A workflow must contain one Start event and one End event. When you create a workflow, the Developer tool adds the Start event and End event to the workflow.

1. Open the workflow in the editor.
2. Select an object from the **Workflow Object** palette and drag it to the editor. If you selected a Mapping task, click **Browse** to select the mapping and then click **Finish**.

   Or to add a Mapping task, select a mapping from the **Object Explorer** view and drag it to the editor.

   The object appears in the editor. Select the object to configure the object properties.

Connecting Objects

Connect objects with sequence flows to determine the order that the Data Integration Service runs the objects in the workflow.

To connect two objects, select the first object in the editor and drag it to the second object. To connect multiple objects, use the **Connect Workflow Objects** dialog box.

1. Right-click in the editor and select **Connect Workflow Objects**.

   The **Connect Workflow Objects** dialog box appears.
2. Select the object that you want to connect from, select the object that you want to connect to, and click **Apply**.
3. Continue connecting additional objects, and then click **OK**.

   The sequence flows appear between the objects.

Creating a Conditional Sequence Flow

A conditional sequence flow includes an expression that evaluates to true or false. Create a conditional sequence flow to determine whether the Data Integration Service runs the next object in the workflow.

1. Select a sequence flow in the editor.
2. In the **Properties** view, click the **Condition** tab.
3. Enter the conditional expression.

   The **Functions** tab lists transformation language functions. The **Inputs** tab lists workflow parameters and variables. Double-click a function, parameter, or variable name to include it in the expression.
   Type operators and literal values into the expression as needed.
4. Validate the condition using the **Validate** button.

   Errors appear in a dialog box.
5. If an error appears, fix the error and validate the condition again.

Workflow Validation

When you develop a workflow, you must configure it so that the Data Integration Service can read and process the entire workflow. The Developer tool marks a workflow as not valid when it detects errors that will prevent the Data Integration Service from running the workflow.

When you validate a workflow, the Developer tool validates sequence flows, expressions, and workflow objects.

Workflow Object Validation

The Developer tool performs workflow object validation each time you validate a workflow.

The Developer tool validates the following workflow objects:

Events

The workflow contains one Start event that is the first object in the workflow. The workflow contains one End event that is the last object in the workflow. The workflow has a path from the Start event to the End event.

Tasks

Each task has a unique name within the workflow. If applicable, task input is assigned to workflow parameters and variables with compatible types. If applicable, task output is assigned to workflow variables with compatible datatypes. Task configuration properties are assigned to valid values.

Each Assignment task assigns a valid value to a single workflow variable. The value assigned to the workflow variable has a compatible datatype. If the task uses workflow parameters or variables in the assignment expression, the Developer tool verifies that the parameters and variables exist.

Each Command task includes a command that does not contain a carriage return character or line feed character. If the command uses workflow parameters or variables, the Developer tool verifies that the parameters and variables exist.

Each Mapping task includes a valid mapping that exists in the repository.

Each Notification task includes at least one recipient. If the task uses workflow parameters or variables, the Developer tool verifies that the parameters and variables exist.

Gateways

Each gateway has a unique name within the workflow.

Sequence Flow Validation

The Developer tool performs sequence flow validation each time you validate a workflow.

The Developer tool makes the following sequence flow validations:

- The workflow must not include looped sequence flows. Each sequence flow can run one time.
- The Start event has a single outgoing sequence flow.
- The sequence flow that leads from the Start event does not include a condition.
- The End event has a single incoming sequence flow.
- Each task has a single incoming sequence flow and a single outgoing sequence flow.
- Each gateway has either multiple incoming sequence flows or multiple outgoing sequence flows, but not both. A gateway that splits the workflow has at least two outgoing sequence flows, with one of the sequence flows set as the default. A gateway that merges the workflow does not have a default outgoing sequence flow.
- An expression in a conditional sequence flow must return a boolean or integer value. The expression cannot contain a carriage return character or a line feed character.

Expression Validation

You can validate an expression in a conditional sequence flow or in an Assignment task while you are creating the expression. If you did not correct the errors, error messages appear in the **Validation Log** view when you validate the workflow.

Validating a Workflow

Validate a workflow to ensure that the Data Integration Service can read and process the entire workflow.

1. Open the workflow in the editor.
2. Click **Edit** > **Validate**.

   Errors appear in the **Validation Log** view.
3. If an error appears, fix the error and validate the workflow again.

Importing Workflows from PowerCenter

When you import a workflow from PowerCenter, the Start and Session task, which includes both reusable and non-reusable task, imports into the Model repository. In the Model repository, each workflow contains an End event after you import from PowerCenter.

Use the Developer tool or infacmd to import mappings from PowerCenter or export mappings to PowerCenter to reuse the metadata.

To import data from PowerCenter into the Model Repository, complete the following tasks:

1. Export PowerCenter objects to a file using the PowerCenter Client or with the following command:
   ```
   pmrep ExportObject
   ```
2. Convert the export file to a Model repository file using the following command:
   ```
   infacmd ipc importFromPC
   ```
3. Import the objects using the Developer tool or with the following command:
   ```
   infacmd tools importObjects
   ```

Branches in the PowerCenter workflows gets routed through Inclusive gateways in the imported workflows in the Model repository. An Inclusive gateway requires at least one of its outgoing sequence flows to be set as default. The first outgoing sequence flow from the gateway becomes the default and the imported workflow remains valid.

Rules and Guidelines for Importing Workflows from PowerCenter

If you import a workflow, consider the following rules and guidelines:

The import process links PowerCenter session or task instances without outgoing links to the End event in the Model repository.

If a PowerCenter workflow contains a session or task instance that does not have outgoing links, the imported workflow adds a link to the gateway to connect the session or task instance to the End event in the Model repository.

The imported workflow removes an unsupported PowerCenter task. If the unsupported tasks are part of a branch or merge, the import process removes unsupported tasks and replaces the branches or merges with gateway tasks.

If a PowerCenter mapping contains an task in the workflow that is not supported for import into the Model repository, then the unsupported tasks are removed from the imported workflow in the Model repository. However, if the unsupported tasks were part of a branch or merge, the gateways for these tasks are created and are present as an unconnected task in the final imported workflow.

The import process splits session tasks with additional session tasks for each mapping corresponding to each pipeline based on target load order.

If a workflow contains a session task in PowerCenter, the task gets split when the session points to a mapping with multiple pipelines for each target load order after import in the Model repository. The imported workflow contains additional session tasks for each mapping corresponding to each pipeline based on the target load order.

For example, you have a PowerCenter workflow containing a Start task linked to a session. The session points a mapping having two pipelines. The first pipeline has a source, Source1 pointing to the transformation, Tx1 that leads to a target, Target1. The second pipeline has source, Source2 that points to transformation, Tx2 that leads to the target, Target2.

After you import the workflow in the Model repository, two sessions are created and linked according to the target load order as follows: Start -> Session_Mapping_pipeline_1 -> Session_Mapping_pipeline_2 -> End, where Session_Mapping_pipeline_1 points to the mapping created from the first pipeline and Session_Mapping_pipeline_2 points to the mapping created from the second pipeline.

Importing a Command Task from PowerCenter

You can import a Command task from PowerCenter into the Model repository.

If you import a Command task, consider the following information:

You can add multiple commands within a PowerCenter Command task. The Developer tool has a single field for adding all the commands for the Command task.

In PowerCenter, you can specify multiple commands within a Command task. In the Developer tool, a single field contains all the commands.

Check if you have selected or deselected the Fail Task if Any Command Fails session property before importing the Command task.

Before you import a Command task from PowerCenter into the Model repository, check whether you have selected the **Fail Task if Any Command Fails** session property in PowerCenter or not. The import process might append either a double ampersand (&&) or a single semicolon (;) at the end of each command except the last command based on the session property.

If you have disabled the **Fail Task if Any Command Fails** session property for the Command task, the import process appends each command except the last one with a semicolon in the Developer tool. For example, `<first_command>; <second_command>`.

If you have enabled the **Fail Task if Any Command Fails** session property for the Command task, the import process appends each command with a double ampersand (&&) in the Developer tool. For example, `<first_command>&& <second_command>`.

Conversion Behavior with the Command Task Import

Consider conversion behavior for importing the Command task when you use a semicolon or a backslash within a command.

The following scenarios show the conversion behavior with the Command task:

- There are two commands in PowerCenter. The first command contains a semicolon at the end, such as `<c1;c2;>` and the second command is `<c3>`.

  The import process converts the command with the following syntax:

  `“<C1;C2>”; “<C3>”`

  OR

  `“<C1;C2>”&& “<C3>”`

- There are two commands in PowerCenter. The first command contains a backslash and a semicolon at the end, such as `<C1;C2\;>` and the second command is `<C3>`.

  The import process converts the command with the following syntax:

  `“<C1;C2\;>”; “<C3>”`

  OR

  `“<C1;C2\;>”&& “<C3>”`

- You have two commands in PowerCenter. The first command contains a backslash and two semicolons at the end as `“<C1;C2\;;>”` and the second command is `“<C3>”`.

  The import process converts the command with the following syntax:

  `“<C1;C2\;>”; “<C3>”` OR `“<C1;C2\;>”&& “<C3>”`

  OR

  `“<C1;C2\;>”&& “<C3>”`

Workflow Advanced Properties

The workflow advanced properties include properties that define how workflow instances run.

Tracing Level

Determines the amount of detail that appears in the workflow log. You can select a value for the tracing level. Or, you can assign the tracing level to a parameter so that you can define the value of the property in a workflow parameter. The tracing level has a string datatype.

Default is INFO.

The following table describes the workflow tracing levels:

Tracing Level	Description
ERROR	Logs error messages that caused the workflow instance to fail. The workflow log displays this level as SEVERE.
WARNING	In addition to the error level messages, logs warning messages that indicate failures occurred, but the failures did not cause the workflow instance to fail. The workflow log displays this level as WARNING.

Tracing Level	Description
INFO	In addition to the warning level messages, logs additional initialization information and details about the workflow instance run. Logs task processing details including the input data passed to the task, the work item completed by the task, and the output data produced by the task. Also logs the parameter file name and expression evaluation results for conditional sequence flows. The workflow log displays this level as INFO.
TRACE	In addition to the info level messages, logs additional details about workflow or task initialization. The workflow log displays this level as FINE.
DEBUG	In addition to the trace level messages, logs additional details about task input and task output and about the workflow state. The workflow log displays this level as FINEST.

Enable Recovery

Indicates that the workflow is enabled for recovery. When you enable a workflow for recovery, you can recover a workflow instance if a task with a restart recovery strategy encounters a recoverable error, if you cancel the workflow instance, or if the Data Integration Service process shuts down unexpectedly. When you enable a workflow for recovery, you must define a recovery strategy for each task in the workflow.

Default is disabled.

Automatically Recover Workflows

Indicates that the Data Integration Service process automatically recovers workflow instances that were interrupted by an unexpected service process shutdown. The workflow recovery starts after the Data Integration Service process restarts. You can select this option if the workflow is enabled for recovery.

Default is disabled.

Workflow Deployment

When you develop a workflow in the Developer tool, you create a workflow definition. To run an instance of the workflow, you add the workflow definition to an application. Then, you deploy the application to the Data Integration Service.

Deploy workflows to allow users to run workflows using the infacmd wfs startWorkflow command. When you deploy a workflow, the Data Integration Service creates a separate set of run-time metadata in the Model repository for the workflow. If you make changes to a workflow definition in the Developer tool after you deploy it, you must redeploy the application that contains the workflow definition or deploy an application patch that updates the workflow for the changes to take effect.

Use the Developer tool to deploy workflows. You deploy workflows using the same procedure that you use to deploy other Model repository objects.

Deploy and Run a Workflow

When you deploy a workflow to the Data Integration Service, you can run a single instance of the workflow immediately after you deploy it. When you deploy and run a workflow, you cannot specify a parameter file. If the workflow uses parameters, the Data Integration Service uses the default parameter values.

To run a workflow immediately after you deploy it, click **Run Object** in the **Deploy Completed** dialog box. If the deployed application contains multiple workflows, select the workflows to run. The Data Integration Service concurrently runs an instance of each selected workflow. If the deployed application contains other object types, you cannot select those objects to run.

Monitor the workflow instance run in the **Monitoring** tab of the Administrator tool. To run additional instances of the workflow, use the infacmd wfs startWorkflow command.

If you receive an error message when you deploy and run a workflow, view the workflow and Data Integration Service logs for more information.

Running Workflows

After you deploy a workflow, you run an instance of the workflow from the deployed application using the infacmd wfs startWorkflow command. You can specify a parameter file for the workflow run.

You can concurrently run multiple instances of the same workflow from the deployed application. When you run a workflow instance, the application sends the request to the Data Integration Service. The Data Integration Service runs the objects in the workflow according to the sequence flows connecting the objects.

For example, the following command runs an instance of the workflow MyWorkflow in the deployed application MyApplication using the parameter values defined in the parameter file MyParameterFile:

```
infacmd wfs startWorkflow -dn MyDomain -sn MyDataIntSvs -un MyUser -pd MyPassword -a
MyApplication -wf MyWorkflow -pf MyParameterFile.xml
```

Monitoring Workflows

You monitor a workflow instance run in the Monitoring tool. The Monitoring tool is a direct link to the **Monitoring** tab of the Administrator tool.

The Monitoring tool shows the status of running workflow and workflow object instances. You can abort or cancel a running workflow instance in the Monitoring tool. You can also use the Monitoring tool to view logs for workflow instances and to view workflow reports.

Deleting a Workflow

You might decide to delete a workflow that you no longer use. When you delete a workflow, you delete all objects in the workflow.

When you delete a workflow in the Developer tool, you delete the workflow definition in the Model repository. If the workflow definition has been deployed to a Data Integration Service, you can continue to run instances of the workflow from the deployed workflow definition.

To delete a workflow, select the workflow in the **Object Explorer** view and then click **Edit** > **Delete**.

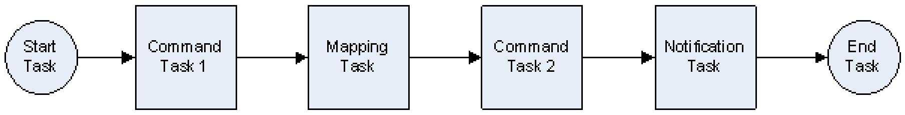
Workflow Examples

The following examples show how you might want to develop workflows.

Example: Running Commands Before and After Running a Mapping

You can develop a workflow that runs commands to perform steps before and after a mapping runs. For example, you might configure a Command task before a Mapping task to drop indexes on the mapping target before the mapping runs. You might configure a Command task after the Mapping task to recreate the indexes when the mapping completes.

The following figure shows a workflow that runs a command, runs a mapping, runs another command, and sends an email notifying users of the status of the workflow:



Parameter files provide you with the flexibility to change the parameter values each time you run a workflow. You can use the following parameters in this workflow:

- Workflow parameter that represents the command that the first Command task runs.
- Mapping parameter that represents the connection to the source for the mapping.
- Mapping parameter that represents the connection to the target for the mapping.
- Workflow parameter that represents the command that the second Command task runs.
- Workflow parameter that represents the email address that the Notification task sends an email to.

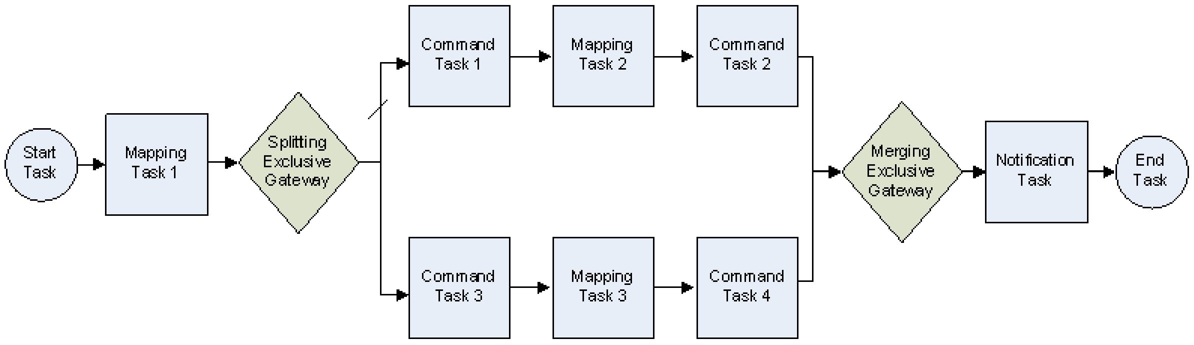
Define the values of the parameters in a parameter file. Specify the parameter file when you run the workflow. You can run the workflow with different parameter files to run different commands, to connect the mapping to a different source or target, or to send an email to different users.

Example: Creating Multiple Sequence Flows

You can develop a workflow that splits a sequence flow into multiple sequence flows and that uses conditions to determine the path that the workflow data follows. Use gateways to create the sequence flows.

For example, you might develop a workflow that follows one sequence flow if a mapping runs successfully and follows another sequence flow if the mapping fails.

The following image shows a workflow that uses an Exclusive gateway to create the sequence flows:



The workflow includes the following components:

- Start event and End event.
- Mapping task. The task runs a mapping and assigns the *Is Successful* output to a boolean workflow variable.
- Exclusive gateway that specifies two outgoing sequence flows. One of the sequence flows includes a condition that evaluates the value of the workflow variable. If the condition evaluates to true, the Data Integration Service runs the next task on the sequence flow. If the condition evaluates to false, the Data Integration Service runs the next task on the other sequence flow.

  In this example, each sequence flow includes a Command task, a Mapping task, and another Command task.
- Exclusive gateway that merges the sequence flows into a single flow.
- Notification task that sends an email to notify users of the status of the workflow.

CHAPTER 2

Workflow Variables

This chapter includes the following topics:

Workflow Variables Overview

A workflow variable represents a value that can change during a workflow run. Use workflow variables to reference values and record run-time information.

You can use system or user-defined workflow variables. A system workflow variable returns system run-time information such as the workflow instance ID, the user who started the workflow, or the workflow start time.

A user-defined workflow variable captures task output or captures criteria that you specify. After you create a user-defined workflow variable, you configure the workflow to assign a run-time value to the variable.

Assign workflow variables to task input and task output to pass data between a task and the workflow.

Use workflow variables for the following reasons:

Determine whether to run the next object based on run-time data.

Use workflow variables in expressions in conditional sequence flows when you want the Data Integration Service to evaluate the variable value and then determine which object to run next. For example, create a user-defined boolean variable that captures the Is Successful task output. Use the variable in the expression of a conditional sequence flow on a gateway to evaluate whether the previous task ran successfully. If it did, run task A. If not, run task B.

Use run-time data for the value of a task field.

Use workflow variables in task fields when you want the Data Integration Service to use the variable value for the field at run time. For example, use the UserName system variable in the list of recipients for a Notification task to send an email to the user that started the workflow.

Task Input

Task input is the data that passes into a task from workflow parameters and variables. The task uses the input data to complete a unit of work.

When you configure a task, you specify which workflow parameters and variables the task requires. The Data Integration Service copies the workflow parameter and variable values to the task when the task starts.

Some tasks include an **Input** tab where you specify the workflow parameters and variables that the task requires. For other tasks, you specify the workflow parameters and variables needed by the task in other tabs.

Related Topics:

- “Assigning Workflow Parameters to Task Input” on page 39
- “Assigning Variables to Task Input” on page 32

Task Output

When you configure a Mapping, Command, Notification, or Human task, you can define the task output. Task output is the data that passes from a task into workflow variables.

When you configure a task, you specify the task output values that you want to assign to workflow variables. The Data Integration Service copies the task output values to workflow variables when the task completes. The Data Integration Service can access these values from the workflow variables when it evaluates expressions in conditional sequence flows and when it runs additional objects in the workflow.

For example, each task includes an Is Successful output value that indicates whether the task ran successfully. The workflow cannot directly access this task output data. To use the data in the remainder of the workflow, you create a boolean workflow variable named TaskSuccessful and assign the Is Successful output to the variable. Then use the TaskSuccessful workflow variable in an expression for a conditional sequence flow. The Data Integration Service runs the next object in the workflow if the previous task ran successfully.

Tasks produce general outputs and task specific outputs. If a task fails, the Data Integration Service copies the general task output values to workflow variables. The service does not copy the task specific output values to workflow variables. If a task aborts, the Data Integration Service does not copy any task output values to workflow variables.

The following table describes the general outputs produced by each task:

Output Data	Datatype	Description
Start Time	Date	Date and time that the task started running.
End Time	Date	Date and time that the task finished running.
Is Successful	Boolean	Indicates whether the task ran successfully.

Note: The Assignment task does not produce general or task specific outputs.

Related Topics:

- "Mapping Task Output" on page 88
- "Assign a Value with Task Output" on page 31
- "Command Task Output" on page 62
- "Notification Task Output" on page 115

System Workflow Variables

System workflow variables return system run-time information.

You cannot create system workflow variables. The Developer tool provides a pre-defined list of system workflow variables that you can use in a workflow.

Use a system workflow variable in a conditional sequence flow or in a task field so that the Data Integration Service uses the variable value at run time. For example, use the UserName system variable in the list of recipients for the Notification task to send an email to the user that runs the workflow.

The following table describes the system workflow variables:

System Variable	Datatype	Description
InstanceID	String	Unique ID of the workflow instance.
StartTime	Date	Date and time that the workflow instance starts running.
UserName	String	Name of the user that runs the workflow instance.

User-Defined Workflow Variables

Create user-defined workflow variables to capture task output or to make a workflow decision based on criteria that you specify. You can create a user-defined workflow variable of datatype boolean, date, integer, or string.

To use user-defined workflow variables, complete the following steps:

1. Create a workflow variable with an initial value.

   The Data Integration Service uses the initial value of the variable when the workflow starts.
2. Configure the workflow to assign a run-time value to the variable.

   As a workflow progresses, the Data Integration Service can calculate and change the initial variable value according to how you configure the workflow. You can assign a value to a user-defined variable using an Assignment task. Or, you can assign a value to a user-defined variable using task output.
3. Use the variable in a conditional sequence flow or in a task field so that the Data Integration Service uses the variable value at run time.

Creating a User-Defined Variable

Create a user-defined workflow variable to record run-time information.

1. Open the workflow in the editor.
2. Create a user-defined workflow variable in the workflow properties or in a task properties.
    - In the workflow **Properties** view, click the **Variables** tab. In the **User** view, click **Add**.
    - In a task **Properties** view, select the **Input** tab or **Output** tab. Select **New Variable** in the Value or Variable column.

    The **Add Variable** dialog box displays.
3. Enter a name and optionally a description for the variable.
4. Select the variable datatype.
5. Enter an initial value for the variable.
6. Click **OK.**

Assign a Value with an Assignment Task

An Assignment task assigns a value to a user-defined workflow variable.

When you create a user-defined workflow variable, you enter an initial value. The Data Integration Service uses the initial value of the variable when the workflow starts. You can add an Assignment task to the workflow to assign another value to the variable. The Data Integration Service uses the assigned value for the variable during the remainder of the workflow.

For example, you create a counter variable and set the initial value to 0. In the Assignment task, you increment the variable by setting the variable to its current value plus 1.

When you add an Assignment task to a workflow, you select the user-defined variable whose value you want to change. Then, you write an expression to assign a value to the selected variable.

The following table lists the values that you can assign to a user-defined variable:

Value	Example
Literal value	For example, to assign the value 500 to a user-defined variable, enter the following value in the expression: `500`
Workflow parameter	For example, to assign the value of a workflow parameter to a user-defined variable, enter the following value in the expression: `$par:MyParameter`

Value	Example
Workflow system or user-defined variable	For example, to assign the value of a workflow system or user-defined variable to another user-defined variable, enter the following value in the expression: `$var:MyVariable`

Any valid expression using the transformation language functions and operators	The expression must return a boolean, date, integer, or string value. Use a conversion function to convert a return value with another datatype to one of the supported datatypes.
	For example, to assign the value of an expression to a user-defined variable, enter the following value in the expression:
	`LENGTH('test')`
	If you use the equality operator (=) in the expression, the Data Integration Service checks whether both sides of the expression are equal and returns true or false. For example, the following expression assigns either true or false to the selected user-defined variable:
	`$var.MyVariable = 7 + 5`

You cannot assign values to system workflow variables.

Parameters and Variables in Assignment Expressions

You can include workflow parameters and variables in the expression value that you assign to a user-defined workflow variable.

You can select a workflow parameter or variable from the **Inputs** tab in the **Assignment Expression Editor**, or you can type the parameter or variable name in the expression using the required syntax.

For example, you create a user-defined workflow variable named Counter and set the initial value to 0. Use the Assignment task to increment the value of the variable by 1. Enter the following expression in the Assignment task:

```
$var:Counter + 1
```

The Data Integration Service does not resolve workflow variable or parameter values that are included in a string literal in an assignment expression. For example, you use an Assignment task to assign the following value to a variable:

```
‘The mapping completed successfully: ${var:MappingIsSuccessful}’
```

The Data Integration Service does not evaluate the string literal, and so does not resolve the MappingIsSuccessful variable value. The Data Integration Service displays the variable name in the string.

Configuring an Assignment Task

Before you can use an Assignment task to assign a value to a user-defined workflow variable, you must create the workflow variable with an initial value.

1. Add an Assignment task to the workflow.
2. Select the Assignment task in the editor.
3. In the **Properties** view, click the **Assignment** tab.
4. In the **User-defined Variable** column, select a user-defined workflow variable.
5. Click the arrow in the Expression column.

   The **Assignment Expression Editor** appears.
6. Enter the value or expression to assign to the variable.

The **Functions** tab lists transformation language functions. The **Inputs** tab lists workflow parameters and variables. Double-click a function, parameter, or variable name to include it in the expression.
Type operators and literal values into the expression as needed.

7. Validate the expression using the **Validate** button.

   Errors appear in a dialog box.
8. Fix errors and validate the expression again.
9. Click **OK**.

Assign a Value with Task Output

Assign task output to a user-defined workflow variable when you want to pass output data produced by the task to the remainder of the workflow.

When you create a user-defined workflow variable, you enter an initial value. The Data Integration Service uses the initial value of the variable when the workflow starts. You use a task **Output** tab to assign another value to the variable. After the task completes, the Data Integration Service uses the task output value for the variable during the remainder of the workflow.

For example, you create a workflow variable named CommandStdOutput and set the initial value to "test." In the Command task **Output** tab, you assign the CommandStdOutput workflow variable to the standard output returned by the command. When the workflow starts, the Data Integration Service sets the workflow variable value to "test." If you use the echo command in the Command task to print the value of the CommandStdOutput variable, the Data Integration Service prints the initial value of "test." After the Command task completes, the Data Integration Service sets the workflow variable value to the standard output returned by the command.

You cannot assign task output to system workflow variables.

Related Topics:

- “Mapping Task Output” on page 88
- “Task Output” on page 27
- “Command Task Output” on page 62
- “Notification Task Output” on page 115

Assigning Task Output

You can assign task output values to user-defined workflow variables.

1. Open the workflow in the editor.
2. Select a task that produces output data in the editor.
3. In the **Properties** view, click the **Output** tab.

   The tab lists all output data that the task produces.
4. Enter a string to search for an output.

   You can use wildcard characters in the string. The string is not case sensitive.
5. Click the **Variable** column for an output.
6. Select a variable name or click **New Variable** to create and assign a new variable to the output.
7. To clear an output assignment, select an output and click **Clear**. Or, click **Clear All** to clear all output assignments.

Where to Use Workflow Variables

Use a workflow variable in an expression in a conditional sequence flow when you want the Data Integration Service to evaluate the variable value and then determine which object to run next. Use a workflow variable in a task field when you want the Data Integration Service to use the variable value for the field.

Depending on the expression or task field, you can select or type the workflow variable name.

The following table lists the objects and fields where you can use workflow variables:

Object	Tab or Dialog Box	Fields	Select or Type
Sequence flow	Condition tab	Condition	both
Assignment task	Assignment Expression Editor dialog box	Expression	both
Command task	Command tab	Command	both
Command task	Input tab	Advanced configuration properties assigned to task input	select
Human task	Input tab	Number of items processed	select
Mapping task	Input tab	User-defined mapping parameters Advanced configuration properties assigned to task input	select
Notification task	Notification tab	Dynamic email content	type
Notification task	Email Properties dialog box	Dynamic recipients Dynamic email addresses	select
Notification task	Email Properties dialog box	Dynamic email content	both

Assigning Variables to Task Input

Mapping and Command tasks include an **Input** tab where you specify the workflow variables that the task requires.

In a Mapping and Command task **Input** tab, you can assign task configuration properties to task input to define the value of the property in a workflow variable. The **Advanced** tab for a task lists the task configuration properties.

In a Mapping task **Input** tab, you can also assign user-defined mapping parameters to workflow variables to use workflow run-time data for the user-defined mapping parameter value.

1. Select a Mapping or Command task in the editor.
2. In the **Properties** view, click the **Advanced** tab to assign an advanced configuration property to task input.

   In the **Value** column for a property, select **Assigned to task input**.
3. Click the **Input** tab.
4. Enter a string to search for an input.

   You can use wildcard characters in the string. The string is not case sensitive.

5. Click the **Value** column for a configuration property or mapping parameter.
6. Assign the property or parameter to an existing workflow variable, to a new workflow variable, or to a literal value.
    - Select a workflow variable name.
    - Click **New Variable**. In the **Add Variable** dialog box, enter the name, type, and initial value for a workflow variable. The Developer tool creates the workflow variable and assigns the variable to the property.
    - Click **New Value**. In the **Add Value** dialog box, enter the literal value and datatype to assign to the property.
7. To clear an input assignment, select an input and click **Clear**. Or, click **Clear All** to clear all input assignments.

Related Topics:

- "Mapping Task Input" on page 84
- "Task Input" on page 27
- "Command Task Input" on page 61

Variable Names in Expressions and Strings

When you use a workflow variable name in an expression or a string field, you can select the name from the **Inputs** tab or you can type the name using the required syntax.

The following table shows the required syntax for workflow variable names in expression and string fields:

Field	Syntax	Example
Expression in a conditional sequence flow or in an Assignment task	- `$var:<variable_name>` for user-defined variables - `$var:sys.<variable_name>` for system variables	For example, you create a workflow variable named CommandExitCode and assign the exit code output value for a Command task to the variable. You create the following expression in the conditional sequence flow that connects the Command task to a Mapping task: `$var:CommandExitCode = 0` The Data Integration Service evaluates the condition and runs the Mapping task if the previous Command task returned 0 in the exit code which indicates that the command succeeded.
String field for a Command or Notification task	- `${var:<variable_name>}` for user-defined variables - `${var:sys.<variable_name>}` for system variables	When you enter a variable name in a string field for a Command or Notification task, you must include brackets around the variable name. For example, you create a workflow variable named MappingErrorRows and assign the number of error rows output value for a Mapping task to the variable. You enter the following text in the body of a Notification task: `Mapping failed to write ${var:MappingErrorRows} rows to the target.`

If you do not include "`var:`" in the variable name, the Data Integration Service uses the name as a parameter. For example, if you enter `$CommandExitCode` or `${CommandExitCode}`, the Data Integration Service uses `$par:CommandExitCode` or `${par:CommandExitCode}`.

Escape Characters in Strings

When you use a workflow variable name in a string field, you can use an escape character so that the Data Integration Service displays the workflow variable name in the string instead of resolving the variable value.

Use the backslash (\) as an escape character before the ${...} syntax for workflow variable names.

For example, you have a workflow string variable named myVariable with a value of "test". You enter the following text in the body field for a Notification task:

```
Variable \${var:myVariable} has a value of ${var:myVariable}
```

When you run the workflow, the Data Integration Service displays the following string in the body field of the email:

```
Variable ${var:myVariable} has a value of test
```

Escape Characters in Directory Paths

If you use a workflow variable name within a directory path, you can use the escape character before the backslashes in the directory path.

The following table provides examples using the escape character with a variable name in a directory path:

Syntax in String Field	Output Value	Description
C:\\${var:myVariable}	C:${var:myVariable}	The Data Integration Service displays the variable name as a string.
C:\\\\${var:myVariable}	C:\test	The Data Integration Service reads the backslash as a regular character and resolves the variable to its value.
C:\temp\\\\${var:myVariable}	C:\temp\test	The Data Integration Service reads the backslash as a regular character and resolves the variable to its value. No escape character is required for the first backslash.
C:\\\\\\${var:myVariable}	C:\\${var:myVariable}	The Data Integration Service reads the backslash as a regular character and displays the variable name as a string.
C:\\\\\\\\${var:myVariable}	C:\\\\test	The Data Integration Service reads two backslashes as regular characters and resolves the variable to its value.

Escape Characters in Command Tasks

Use a Command task to write the output of a command to a file. If the command output includes a variable name, use a backslash as an escape character to add the $ character before the variable name.

When you configure the Command task to run on a non-Windows operating system, use three backslashes. The first backslash is an escape character for the second backslash. The third backslash is an escape character for the $ character.

For example, you define the following command in the Command task:

```
echo \\\${var:myVariable} = ${var: myVariable} > file.txt
```

If the variable has a value of 10 when the workflow runs, the Command task writes the following string to file.txt:

```
${var:Var} = 10
```

Nested Variables

The Data Integration Service resolves one level of variable values. The Data Integration Service does not resolve variable values that are nested within another workflow parameter or variable.

For example, you create the following workflow variables with these datatypes and initial values:

- Variable1 with an integer datatype and an initial value of 4
- Variable2 with an integer datatype and an initial value of 3
- Variable3 with a string datatype and an initial value of `${var:Variable1} + ${var:Variable2}`

When you use Variable3 in an expression or task field, the Data Integration Service does not resolve the nested variables Variable1 and Variable2 to the value of 7. Instead, the Data Integration Service uses the following string value for Variable3:

```
${var:Variable1} + ${var:Variable2}
```

Workflow Variable Datatype Conversion

A workflow variable can have a datatype of boolean, date, integer, or string. You can assign a variable of one datatype to a workflow variable, parameter, literal value, task input, or task output of a different datatype if the Data Integration Service can convert the datatypes.

The following table describes the workflow variable datatype conversion that the Data Integration Service performs:

Variable Datatype	String	Integer	Boolean	Date
String	Yes	Yes	Yes	No
Integer	Yes	Yes	Yes	No
Boolean	Yes	Yes	Yes	No
Date	Yes	No	No	Yes

To convert a string to an integer, the string must contain a number.

To convert a string to a boolean, the string must contain either "true" or "false."

When the Data Integration Service converts an integer to a boolean, the service converts a value of zero (0) to false. The service converts any non-zero value to true.

When the Data Integration Service converts a boolean to an integer, the service converts a value of false to zero. The service converts a value of true to one (1).

When you run the workflow, the Data Integration Service converts the data to the valid datatype. For example, the StartTime system workflow variable has a date datatype. You can use this variable in the body string field of a Notification task. When you run the workflow, the Data Integration Service converts the date stored in the system workflow variable to a string.

Changing the Format of Date Variables

The Data Integration Service uses the format `DAY MON DD HH24:MI:SS YYYY` for workflow variables with a date datatype. You can use an Assignment task to change the default format of a date variable.

Use an Assignment task to convert the date value to a string value with a specified date format. Then, assign the converted value to a string workflow variable.

1. Create a user-defined workflow variable with a string datatype.
2. Add an Assignment task to the workflow after the task that assigns a run-time value to the date workflow variable.
3. Connect the Assignment task to other objects in the workflow.
4. Select the Assignment task in the editor.
5. In the **Properties** view, click the **Assignment** tab.
6. In the **User-defined Variable** column, select the string workflow variable.
7. Click the arrow in the **Expression** column.

   The **Assignment Expression Editor** appears.
8. Enter the following expression to convert the value of the date workflow variable to a string value with the specified date format:

   ```
   TO_CHAR(date_variable [,format])
   ```

   For example, enter the following expression:

   ```
   TO_CHAR($var:MyDateVariable, 'MM/DD/YYYY HH24:MI:SS')
   ```
9. Click **Validate** to validate the expression.

   Errors appear in a dialog box.
10. Fix errors and validate the expression again.
11. Click **OK**.
12. Use the string workflow variable in an expression or task field.

CHAPTER 3

Workflow Parameters

This chapter includes the following topics:

Workflow Parameters Overview

A workflow parameter is a constant value that you define before the workflow runs. Use workflow parameters to set values for tasks in the workflow or to set some user-defined mapping parameters. You can also use workflow parameters to set values for connection parameters or to set values for string parameters, such as configuration properties, command strings, or email addresses.

When you create parameters in a workflow, you can run a workflow with different parameter values. Parameters can reduce the overhead of creating multiple workflows when you need to change certain attributes of a workflow. All workflow parameters are user-defined parameters.

You can assign workflow parameters to task input and pass data from the workflow to the task. For example, you define a working directory parameter and assign the parameter to the Command tasks in the workflow.

You can also assign workflow parameters to user-defined mapping parameters in a Mapping task. For example, you define a workflow parameter that identifies the email address to send a notification email in a Notification task. Reference the workflow parameter in the recipients field for the Notification task.

You can override the value of a workflow parameter by including the parameter in a parameter set or a parameter file. A parameter set is a repository object that contains parameter values. You can deploy a workflow with a parameter set. A parameter file is an XML file that contains parameter values. A parameter file resides on the file system instead of in the repository. When you run a workflow, you can specify a specific parameter set or a parameter file for the workflow run.

Task Input

Task input is the data that passes into a task from workflow parameters and variables. The task uses the input data to complete a unit of work.

When you configure a task, you specify which workflow parameters and variables the task requires. The Data Integration Service copies the workflow parameter and variable values to the task when the task starts.

Mapping tasks, Command tasks, and Human tasks include an **Input** view to configure the workflow parameters and variables that the task requires. You can configure parameters for Mapping task configuration properties on the **Input** view.

You can reference workflow parameters for other tasks in different task views. For example, configure a workflow parameter that contains an email address in the **Notification** view of Notification task.

You can use workflow parameters in expressions in the conditional sequence flows from an outgoing gateway. The Data Integration Service evaluates the parameter values and identifies the object or objects to run next in the workflow.

Process to Run Workflows with Parameters

A workflow parameter represents a constant value that you define before the workflow runs. You can override the parameter value when you include the parameter in a parameter set or a parameter file.

To run workflows with different parameter values, perform the following steps:

1. Create a workflow parameter and assign it a default value.
2. Assign the parameter in task input or assign the parameter to a mapping parameter.
3. Create one or more parameter sets that include the workflow and mapping parameters in the workflow. Change the parameter values as required.
4. Deploy the workflow and the parameter sets to a Data Integration Service.
5. Run the workflow from the command line and specify which parameter set to use for the workflow run.

Note: You can create a parameter file and then run the workflow from the command line with the parameter file. You cannot run a workflow with a parameter file and a parameter set at the same time.

Where to Use Workflow Parameters

Use a workflow parameter in an expression in a conditional sequence flow when you want the Data Integration Service to evaluate the parameter value and then determine which object to run next. Use a workflow parameter in an object field when you want the Data Integration Service to use the parameter value for the field.

Depending on the expression or task field, you can select or type the workflow parameter name.

The following table lists the objects and fields where you can use workflow parameters:

Object	Tab or Dialog Box	Fields	Select or Type
Workflow	Advanced tab	Tracing level	select
Sequence flow	Condition tab	Condition	both
Assignment task	Assignment Expression Editor dialog box	Expression	both
Command task	Command tab	Command	both
Command task	Input tab	Advanced configuration properties assigned to task input	select
Human task	Input tab	Number of items processed	select
Mapping task	Input tab	User-defined mapping parameters Advanced configuration properties assigned to task input	select
Notification task	Notification tab	Dynamic email content	type
Notification task	Email Properties dialog box	Dynamic recipients Dynamic email addresses	select
Notification task	Email Properties dialog box	Dynamic email content	both

Assigning Workflow Parameters to Task Input

Mapping tasks, Human tasks, and Command tasks include an **Input** tab where you specify the workflow parameters that the task requires.

On the **Input** tab of a Mapping or Command task, you can assign task configuration properties to task input to define the value of the property in a workflow parameter. The **Advanced** tab for a task lists the task configuration properties.

On the **Input** tab of a Mapping task, you can also assign some user-defined mapping parameters to workflow parameters. You can assign a distinct value to a user-defined mapping parameter that appears multiple times in the workflow.

Note: Workflow parameter types must be either a connection type or a string type.

1. Select a Mapping, Command, or Human task in the editor.
2. Click the **Properties** view.
3. For a Mapping task or a Command task, click the **Advanced** tab to assign an advanced configuration property to the task input.

   In the **Value** column for a property, select **Assigned to task input**.
4. Click the **Input** tab.
5. At the top of the **Input** tab, search for the property that you want to update.

   You can use wildcard characters in the search string. The string is not case sensitive.
6. Click the **Value** column for the property.

7. In the **Value** column, choose to assign the property to an existing workflow parameter, to a new workflow parameter, or to a literal value.
    - Select an existing workflow parameter name.
    - Create a workflow parameter. Click **New Parameter**. In the **Add Parameter** dialog box, enter the name, type, and default value for a workflow parameter. The Developer tool creates the workflow parameter and assigns the parameter to the property.
    - Click **New Value**. In the **Add Value** dialog box, enter the literal value and data type to assign to the property.
8. To clear an input assignment, select an input and click **Clear**. Or, click **Clear All** to clear all input assignments.

Related Topics:

- "Mapping Task Input" on page 84
- "Task Input" on page 27
- "Command Task Input" on page 61

Parameter Names in Expressions and Strings

When you use a workflow parameter name in an expression or a string field, you can select the name from the **Inputs** tab or you can type the name using the required syntax.

The following table shows the required syntax for workflow parameter names in expression and string fields:

Field	Syntax	Example
Expression in a conditional sequence flow or in an Assignment task	`$par:<parameter_name>`	For example, you create the following expression in a conditional sequence flow: `$par:Connection=SourceConnection` The Data Integration Service evaluates the condition and runs the connected task if the parameter value is SourceConnection.
String field for a Command or Notification task	`${par:<parameter_name>}`	When you enter a parameter name in a string field for a Command or Notification task, you must include brackets around the parameter name. For example, the following command in a Command task uses a workflow parameter named SourceDirectory to define the source directory from which the command copies a file: `copy ${par:SourceDirectory} H:\marketing\`

If you do not include "`par:`" in the parameter name, the Data Integration Service uses the name as a parameter. For example, if you enter `$SourceDirectory` or `${SourceDirectory}`, the Data Integration Service uses `$par:SourceDirectory` or `${par:SourceDirectory}`.

Escape Character in Strings

When you use a workflow parameter name in a string field, you can use an escape character so that the Data Integration Service displays the workflow parameter name in the string instead of resolving the parameter value.

Use the backslash (\) as an escape character before the ${...} syntax for workflow parameter names.

For example, you have a workflow string parameter named myParameter with a value of "test". You enter the following text in the body field for a Notification task:

```
Parameter \${par:myParameter} has a value of ${par:myParameter}
```

When you run the workflow, the Data Integration Service displays the following string in the body field of the email:

```
Parameter ${par:myParameter} has a value of test
```

If you use a workflow parameter name within a directory path, you can use the escape character before the backslashes in the directory path.

The following table provides examples using the escape character with a parameter name in a directory path:

Syntax in String Field	Output Value	Description
C:\${par:myParameter}	C:${par:myParameter}	The Data Integration Service displays the parameter name as a string.
C:\\${par:myParameter}	C:\test	The Data Integration Service reads the backslash as a regular character and resolves the parameter to its value.
C:\temp\\${par:myParameter}	C:\temp\test	The Data Integration Service reads the backslash as a regular character and resolves the parameter to its value. No escape character is required for the first backslash.
C:\\\${par:myParameter}	C:\${par:myParameter}	The Data Integration Service reads the backslash as a regular character and displays the parameter name as a string.
C:\\\\${par:myParameter}	C:\\test	The Data Integration Service reads two backslashes as regular characters and resolves the variable to its value.

Nested Parameters

The Data Integration Service resolves one level of parameter values. The Data Integration Service does not resolve parameter values that are nested within another workflow parameter or variable.

For example, you assign the following workflow parameters these values in a parameter file:

- Parameter1 has a value of 3
- Parameter2 has a value of 4
- Parameter3 has a value of `${par:Parameter1} + ${par:Parameter2}`

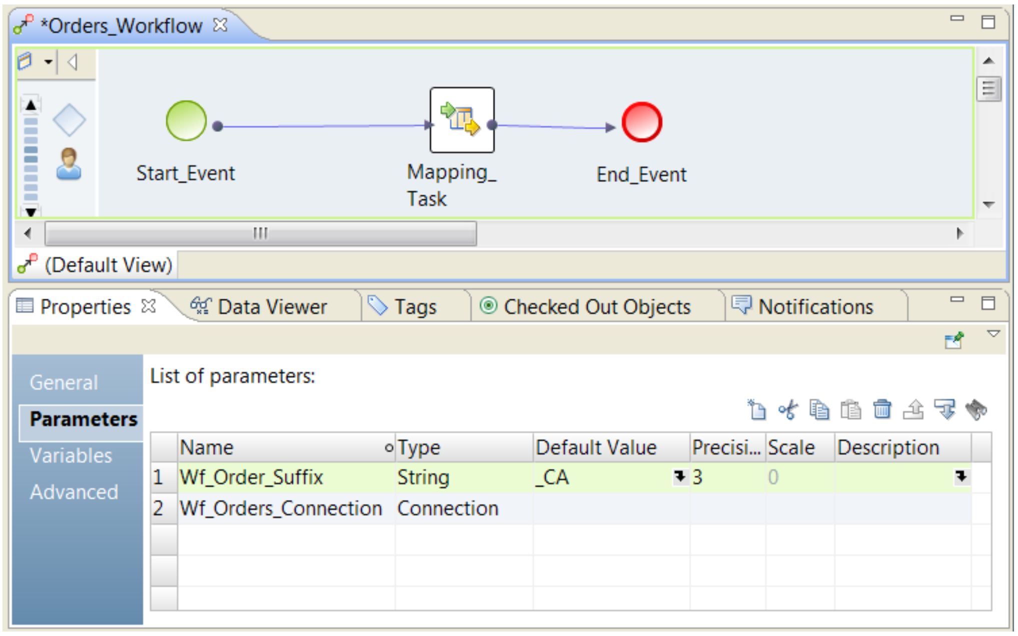
When you use Parameter3 in an expression or task field, the Data Integration Service does not resolve the nested parameters Parameter1 and Parameter2 to the value of 7. Instead, the Data Integration Service uses the following string value for Parameter3:

```
${par:Parameter1} + ${par:Parameter2}
```

Creating Workflow Parameters for User-Defined Mapping Parameters

You can create a workflow parameter to override a user-defined mapping parameter.

1. Open the workflow in the editor.
2. In the workflow **Properties** view, click the **Parameters** tab.



List of parameters:

	Name	Type	Default Value	Precisi...	Scale	Description
1	Wf_Order_Suffix	String	_CA	3	0	
2	Wf_Orders_Connection	Connection				

3. To add a parameter, click **New**.

   The Developer tool creates a parameter with default properties. Change each field in the parameter properties as required.
4. Enter a name for the parameter.
5. Select either a connection or a string parameter type.
6. Enter a default value for the parameter.

   For connection parameters, select a connection. For string parameters, enter a string value.
7. Optionally, enter the scale and the description of the parameter.

Default Parameter Values

When you create a workflow parameter, you must enter a default value.

When you run a workflow with a parameter set or a parameter file, the Data Integration Service resolves all parameters to the values set in the parameter set or parameter file.

The Data Integration Service resolves parameters to the default values in the following circumstances:

- You run a workflow without a parameter set or parameter file.
- You do not define a parameter value in the parameter set or parameter file.

Workflow Parameter data type Conversion

A workflow parameter can have a type of connection or string. You can assign a string workflow parameter to a workflow variable or to task input of an integer or boolean type if the Data Integration Service can convert the data types.

The following table describes the workflow parameter data type conversion that the Data Integration Service performs:

Parameter data type	Integer	Boolean	Date
String	Yes	Yes	No

To convert a string to an integer, the string must contain a number.

To convert a string to a boolean, the string must contain either "true" or "false."

For example, a Mapping task has a High Precision property with a boolean data type. You need to assign a workflow parameter to the property. You can define the workflow parameter as a string with a default value of "true" or "false". When you run the workflow, the Data Integration Service converts the parameter value to the boolean value.

You cannot assign a connection parameter to a parameter that is not a connection parameter.

Parameter Sets

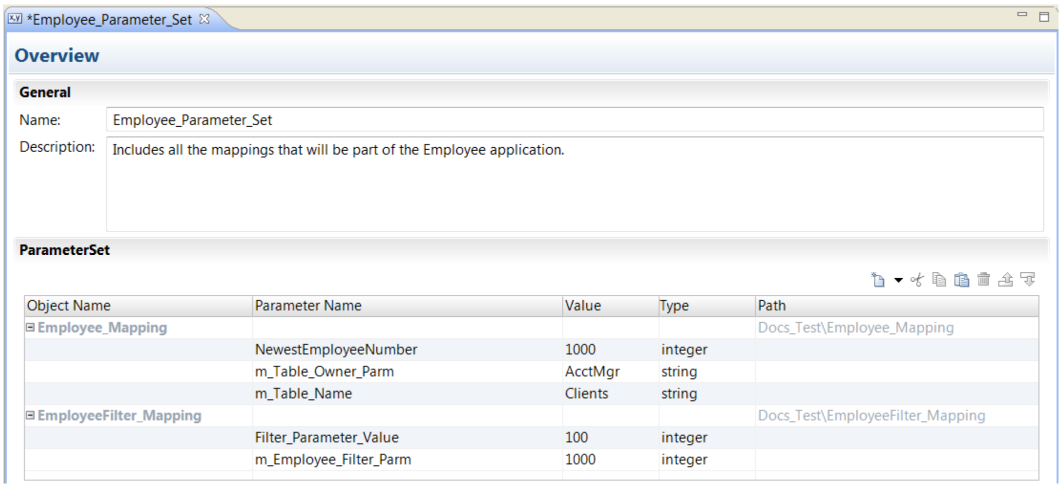
A parameter set is an object in the Model repository that contains a set of parameters and parameter values to run mappings and workflows.

When you create a parameter set, you choose a mapping or workflow to use the parameters. After you choose a mapping or workflow, you can manually enter parameters in the parameter set or you can select parameters that are already in the repository for the mapping or the workflow.

You can use parameter sets for different situations. For example, you might use a specific parameter set when you run a workflow in a test environment.

You use a parameter set with a mapping, Mapping task, or workflow. You can add one or more parameter sets to an application when you deploy the application. You can add a parameter set to multiple applications and deploy them. To use a parameter set with a workflow or mapping, you must add the parameter set to the application when you deploy the workflow or mapping.

The following image shows a parameter set that contains parameters for two mappings:


*Employee_Parameter_Set

Overview

General

Name:	Employee_Parameter_Set
Description:	Includes all the mappings that will be part of the Employee application.

ParameterSet

Object Name	Parameter Name	Value	Type	Path
⊟ Employee_Mapping				Docs_Test\Employee_Mapping
	NewestEmployeeNumber	1000	integer	
	m_Table_Owner_Parm	AcctMgr	string	
	m_Table_Name	Clients	string	
⊟ EmployeeFilter_Mapping				Docs_Test\EmployeeFilter_Mapping
	Filter_Parameter_Value	100	integer	
	m_Employee_Filter_Parm	1000	integer	

The parameter set contains the following information:

Object Name

The name of the mapping, mapplet, or workflow that contains the parameter definition.

Parameter Name

The name of the parameter in the mapping, mapplet, or workflow.

Value

The value of the parameter to use at runtime. The value of the parameter in the parameter set overrides the parameter value in the mapping or workflow.

Type

The type of the parameter. Example parameter types include strings, numeric types, connections, port lists, sort lists, and date\time parameters.

Note: The parameter type that you specify in the parameter set must match the parameter type in the mapping, Mapping task, or workflow. If the parameter types do not match, the mapping, Mapping task, or workflow uses the default value for the parameter.

When you use a parameter set to configure mapping parameter values, the link between the parameter set and the mapping depends on the project where the mapping is defined. If the project name is changed, you must re-establish the link.

To re-establish the link, edit the parameter set and re-select the mapping that uses the parameter set.

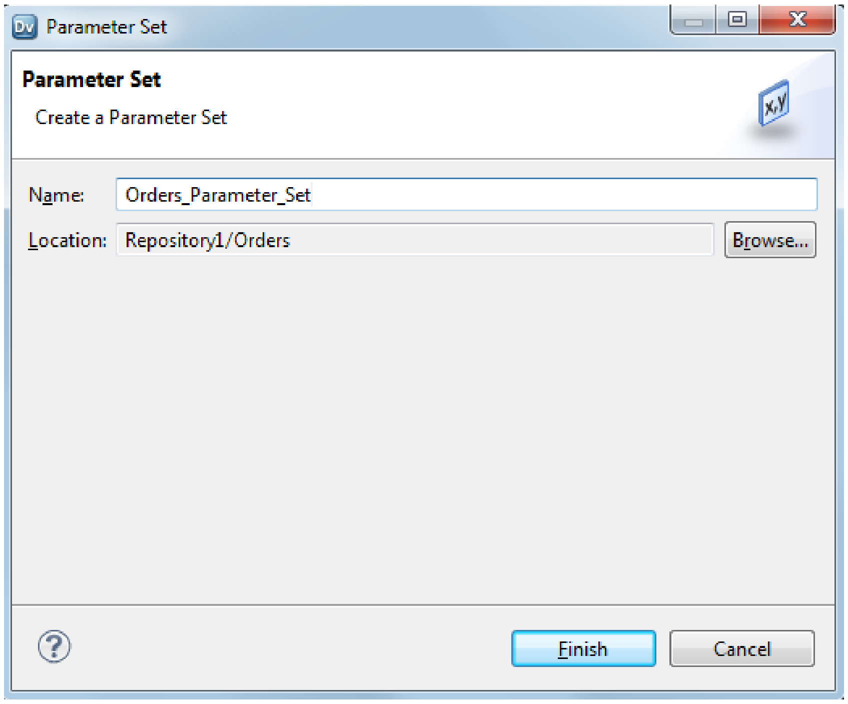
For example, you have a project named P1 that includes a mapping m_1 that uses a parameter set ps_1. You change the project name to P2. Subsequently, you must edit the parameter set to re-select mapping m_1.

Creating a Parameter Set

Create a parameter set that you can use to change the runtime context for mappings and workflows.

When you create the parameter set, choose a mapping or workflow to contain the parameters. After you choose a mapping or workflow, you can manually enter parameters in the parameter set or you can select parameters.

1. In the Object Explorer view, right-click a project and click **New** > **Parameter Set**.
2. Enter a name for the parameter set and click **Finish**.



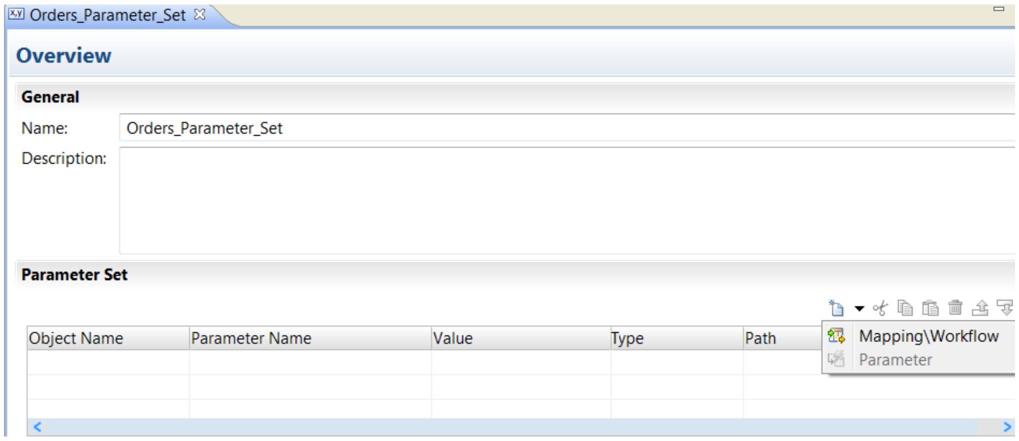
Parameter Set

Create a Parameter Set

Name: Orders_Parameter_Set

Location: Repository1/Orders

3. Drag the **Properties** panel down and view the grid to add the parameters to the parameter set.
4. Click **New** > **Mapping/Workflow**.



Name: Orders_Parameter_Set

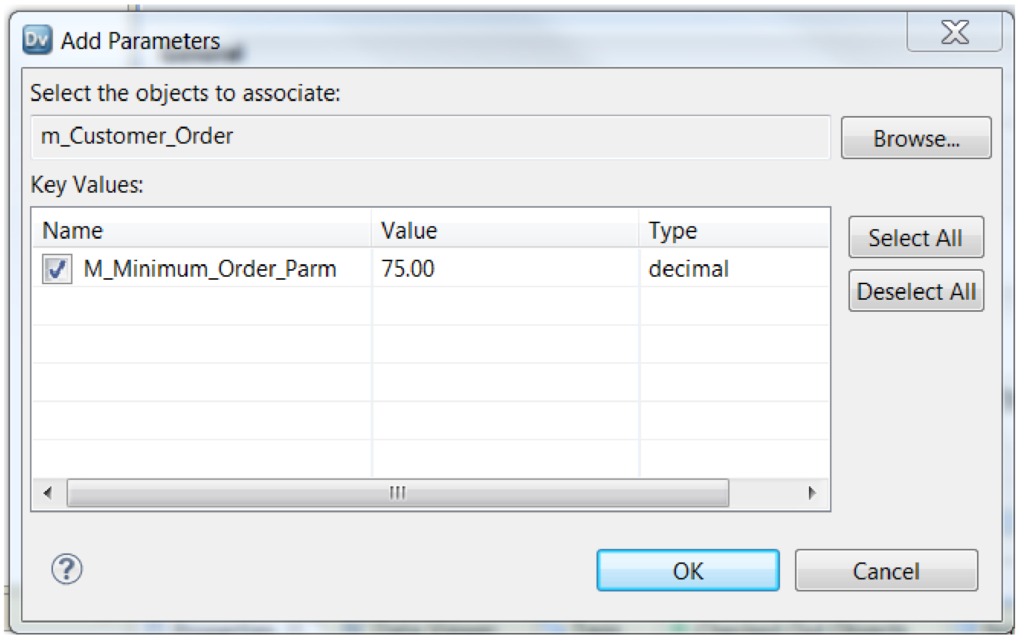
Description:

Object Name	Parameter Name	Value	Type	Path

5. In the **Add Parameters** dialog box click **Browse** to find the mapping or workflow that contains the parameters you need to include in the set.

   A list of mappings and workflows appears.
6. Select a mapping or a workflow and click **OK**.

   A list of parameters from the mapping or workflow appears.



Select the objects to associate:

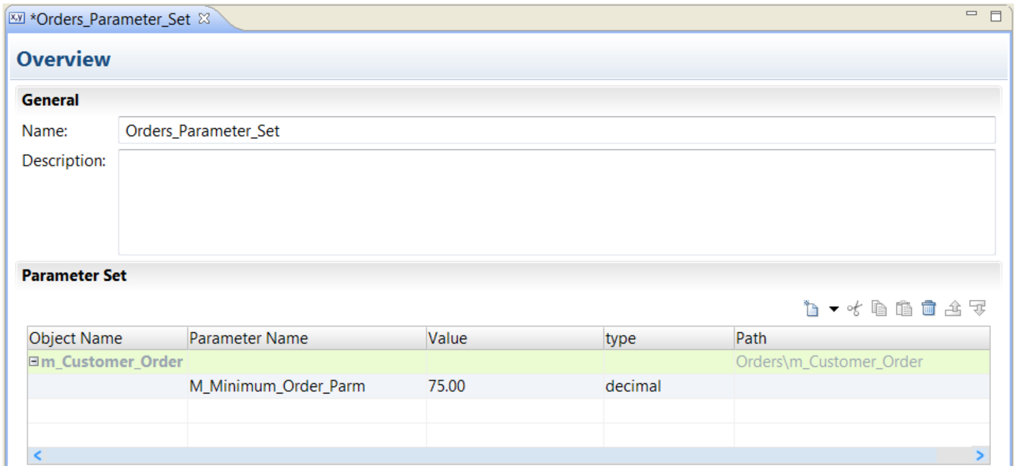
m_Customer_Order

Key Values:

Name	Value	Type
✓ M_Minimum_Order_Parm	75.00	decimal

7. Select the parameters to include in the parameter set and then click **OK**.

   The mapping or the workflow name and the path appears in the parameter set. Each parameter that you selected appears beneath the object.



Name: Orders_Parameter_Set

Description:

Object Name	Parameter Name	Value	type	Path
⊟m_Customer_Order				Orders\m_Customer_Order
	M_Minimum_Order_Parm	75.00	decimal	

8. To add a parameter that is not yet in a workflow or mapping, right-click a mapping or object name and select **Parameter** insert.

   The Developer tool creates a parameter beneath the workflow or mapping. Change the parameter name, the value, and the type.

   Note: You must add the parameter to the mapping or workflow before you use the parameter set.

Running a Workflow with a Parameter Set

Use the `startWorkflow` command to run a workflow with a parameter set. The -ps argument specifies the parameter set name.

When you deploy a workflow with a parameter set, you can use a different parameter set at run time by including the parameter set argument in the command.

For example, the following command runs the workflow myWorkflow with the parameter set "MyParameterSet:"

```
infcmd wfs startWorkflow -dn MyDomain -sn MyDataIntSvs -un MyUser -pd MyPassword -a
MyApplication -wf MyWorkflow -ps MyParameterSet
```

Parameter Files

A parameter file is an XML file that lists user-defined parameters and their assigned values. Parameter files provide the flexibility to change parameter values each time you run a workflow.

The parameter values define properties for a workflow or for a mapping included in a Mapping task that the workflow runs. The Data Integration Service applies these values when you run a workflow from the command line and specify a parameter file.

You can define mapping parameters and workflow parameters in a parameter file. If you want to specify reusable object parameters, expose the reusable object parameters as mapping parameters. Specify the mapping parameter values in the parameter file.

You cannot define system parameter values in a parameter file.

You can define parameters for multiple workflows in a single parameter file. You can also create multiple parameter files and then use a different file each time you run a workflow. The Data Integration Service reads the parameter file at the start of the workflow run to resolve the parameters.

You can export a parameter file from the Developer tool. Export the file from the mapping or the workflow **Parameters** tab. The Developer tool generates a parameter file that contains the mapping or workflow parameters and the default parameter values. You can specify the parameter file name and choose where to save the file.

Note: Parameter files for mappings and workflows use the same structure. You can define parameters for deployed mappings and for deployed workflows in a single parameter file.

You can also list the parameters and default values used in a workflow from the command line. You can use the command line output as a parameter file template.

Run the workflow from the command line to apply a parameter file.

Sample Parameter File for a Workflow

You can create a parameter file for a workflow by running the ListWorkflowParams infacmd. The workflow parameter file does not contain mapping parameters. You can manually add mapping parameters to the file.

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.informatica.com/Parameterization/1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema"><!--Specify deployed application
specific parameters here.--><!--^M
    <application name="a2">
  <workflow name="w2"/>
</application>--><project name="Orders">
      <folder name="integer_op">
         <folder name="Workflows">
            <workflow name="wf_Orders">
               <parameter name="wfStringParam">verboseData</parameter>
               <parameter name="wfConnectionParam">OracleDB</parameter>
            </workflow>
         </folder>
      </folder>
```

```
   </project>
</root>
```

Sample Parameter File

The following example shows a sample parameter file that you can create for a mapping.

```
<?xml version "1.0" encodin="UTF-8?>
<root xmlns="http://www.informatica.com/Parameterization/1.0"
xmlns:xsl="http://www.w3.org/2001/XMLSchema">

<project name="sample_project"

    <mapping name="m_myMapping"

      <parameter name="srcConnect">RelationalConnect</parameter>

      <parameter name="srcDirectory">C:\Srcfiles</parameter>

      <parameter name="wrkDirectory">C:\TempFiles</parameter>

    </mapping>

 </project>

</root>
```

Rules and Guidelines for Parameter Files

Certain rules and guidelines apply when you create parameter files.

Use the following rules when you create a parameter file:

- You can reference mapping and workflow parameters in a parameter file. You cannot reference reusable transformation, mapplet, or data object parameters. To reference reusable object parameters, expose the reusable object parameters as mapping parameters. Specify the mapping parameter values in the parameter file.
- The application element contains mapping or workflow parameters that apply only to the specified application when the application runs. If you run a mapping with a parameter file from the Developer tool and do not deploy the mapping as an application, do not specify an application element. Specify the mapping within a project element.
- Parameter values cannot be empty. For example, the Data Integration Service fails the workflow run if the parameter file contains the following entry:

  ```
  <parameter name="Param1"> </parameter>
  ```
- Within an element, artifact names are not case-sensitive. Therefore, the Data Integration Service interprets <parameter name="SrcDir"> and <parameter name="Srcdir"> as the same application.
- A parameter that identifies a reference table must use a forward-slash (/) to separate folder names in a repository folder path.

Export a Parameter File

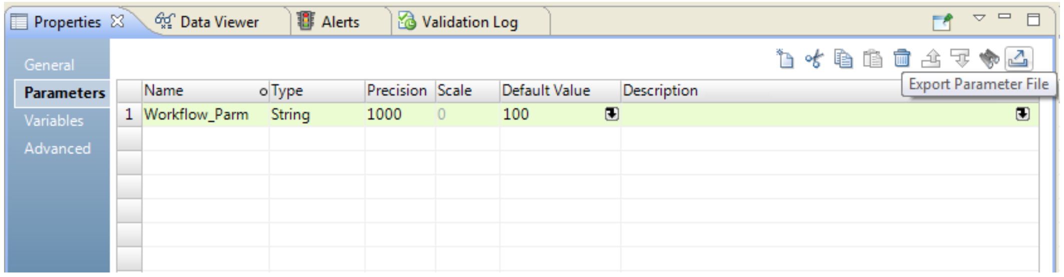
You can export a mapping parameter file or a workflow parameter file from the Developer tool. Define the parameters in the Developer tool and then export them to a file. The Developer tool creates a parameter file in .XML format.

You can export a parameter file that contains mapping parameters or workflow parameters. You can export parameters from the mapping **Parameters** tab or from the workflow **Parameters** tab. The Developer tool exports all the parameters from the **Parameters** tab.

To export a parameter file, perform the following steps:

1. Define the parameters and the parameter defaults for a mapping or a workflow.
2. On the **Parameters** tab of the mapping or workflow **Properties**, click the **Export Parameter File** option.
3. Enter a name for the parameter file and browse for a location to put the file.
4. Click **Save**.

The following image shows the **Export Parameter File** option on the Parameters tab for a workflow:



	Name	Type	Precision	Scale	Default Value	Description
1	Workflow_Parm	String	1000	0	100	

When you export a parameter file, the Developer tool creates a parameter file with either mapping parameters or workflow parameters in it. The Developer tool does not export mapping and workflow parameters to the same file.

For example, when you export the workflow parameter, Workflow_Parm, the Developer tool creates the following parameter file:

```
<?xml version="1.0" encoding="UTF-8"?>
-<root version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema" xmlns="http://
www.informatica.com/Parameterization/1.0">
   -<project name="Orders">
      -<workflow name="Customer_Workflow">
         <parameter name="Workflow_Parm">100</parameter>
      </workflow>
   </project>
</root>
```

Creating a Parameter File from infacmd ms ListMappingParams

The infacmd wfs ListWorkflowParams command lists the parameters for a workflow in a deployed application and the default value for each parameter. Use the output of this command to create a parameter file.

1. Run the infacmd wfs ListWorkflowParams command to list the parameters for a workflow and the default value for each parameter.

   The -o argument sends command output to an XML file.

For example, the following command lists the parameters in workflow MyWorkflow in file "MyOutputFile.xml":

```
infacmd wfs ListWorkflowParams -dn MyDomain -sn MyDataIntSvs -un MyUser -pd
MyPassword -a MyApplication -wf MyWorkflow -o MyOutputFile.xml
```

The Data Integration Service lists all mapping parameters with their default values.

2. If you did not specify the -o argument, you can copy the command output to an XML file and save the file.
3. Edit the XML file and replace the parameter default values with the values you want to use when you run the workflow.
4. Save the XML file.

Running a Workflow with a Parameter File

To run a mapping with a parameter file from the command line, you must deploy the mapping as an application. Run the mapping and specify the parameter file. Use the infacmd wfs StartWorkflow command. The -pf argument specifies the parameter file name.

For example, the following command runs the workflow MyWorkflow using the parameter file "MyParamFile.xml":

```
infacmd wfs StartWorkflow -dn MyDomain -sn MyDataIntSvs -un MyUser -pd MyPassword -a
MyApplication -wf MyWorkflow -pf MyParamFile.xml
```

The Data Integration Service fails the workflow when you run it with a parameter file and the parameter file is not valid. The Data Integration Service fails the workflow if it cannot find the parameter file or it cannot access the parameter file.

For more information about using parameter sets with infacmd, see the *Informatica Command Reference*.

CHAPTER 4

Cluster Tasks

Cluster Task Overview

Cluster tasks are workflow tasks that can create and delete compute clusters on a cloud platform.

Use a Create Cluster task to create, configure and start a compute cluster. Use a Delete Cluster task to delete the same cluster after mapping tasks and any other tasks in the workflow are complete.

A cluster workflow is a workflow that includes a Create Cluster task and one or more Mapping tasks. When you run a cluster workflow, the Create Cluster task issues commands to create a cluster of the size and type that you specify. Mapping tasks and other tasks in the workflow can run on the same cluster or on another cluster that you specify. You can include a Delete Cluster task to terminate the cluster when all tasks are complete. If you do not include a Delete Cluster task, the cluster continues to run.

For more information about cluster workflows, see the *Data Engineering Integration User Guide*.

Create Cluster Task

The Create Cluster task contains all the settings that the distribution requires to create a cluster with a master node and worker nodes. It also contains a reference to a Hadoop or Databricks connection and the cloud provisioning configuration.

When you create a cluster workflow, you drag a Create Cluster task into the workflow editor, then configure task properties.

A cluster workflow has only one Create Cluster task.

Configure the advanced properties based on the cloud platform type.

Create Cluster Task General Properties

The following table describes the General properties that you configure for the Create Cluster task:

Property	Description
Name	Name of the task.
Description	Optional. Description of the task.
Connection Name	Name of the cloud provisioning configuration to use with the workflow.
Connection Type	Choose one of the following Hadoop distributions: - Amazon EMR - HDInsight - Azure Databricks - AWS Databricks Default is Amazon EMR

Create Cluster Task Output

Enter output properties for the Create Cluster task.

Verify that the Cluster Identifier property is set to the default value AutoDeployCluster.

Advanced Properties for Amazon EMR

Set the advanced properties for an Amazon EMR cluster.

General Options

The following table describes general options that you can set for an EMR cluster:

Property	Description
Cluster Name	Name of the cluster to create.
Release Version	EMR version to run on the cluster. Enter the AWS version tag string to designate the version. For example: `emr-5.8.0` Default is Latest version supported.
Connection Name	Name of the Hadoop connection that you configured for use with the cluster workflow.
S3 Log URI	Optional. S3 location of logs for cluster creation. Format: `s3://<bucket name>/<folder name>` If you do not supply a location, no cluster logs will be stored.

Master Instance Group Options

The following table describes master instance group options that you can set for an EMR cluster:

Property	Description
Master Instance Type	Master node EC2 instance type. You can specify any available EC2 instance type. Default is m4.4xlarge.
Master Instance Maximum Spot Price	Maximum spot price for the master node. Setting this property changes the purchasing option of the master instance group to Spot instead of On-demand.

Core Instance Group Options

The following table describes core instance group options that you can set for an EMR cluster:

Property	Description
Core Instance Type	Core node EC2 instance type. You can specify any available EC2 instance type. Default is m4.4xlarge.
Core Instance Count	Number of core EC2 instances to create in the cluster. Default is 2.
Core Instance Maximum Spot Price	Maximum spot price for core nodes. Setting this property changes the purchasing option of the core instance group to Spot instead of On-demand.
Core Auto-Scaling Policy	Optional. Auto-scaling policy for core instances. Type the policy JSON statement here, or provide a path to a file that contains a JSON statement. Format: `file:\\<path_to_policy_config_file>`

Task Instance Group Options

The following table describes task instance group options that you can set for an EMR cluster:

Property	Description
Task Instance Type	Task node EC2 instance type. You can specify any available EC2 instance type. Default is m4.4xlarge.
Task Instance Count	Number of task EC2 instances to create in the cluster. Default is 2.

Property	Description
Task Instance Maximum Spot Price	Maximum spot price for task nodes. Setting this property changes the purchasing option of the task instance group to Spot instead of On-demand.
Task Auto-Scaling Policy	Optional. Auto-scaling policy for task instances. Type the policy JSON statement here, or provide a path to a file that contains a JSON statement. Format: `file:\\<path_to_policy_config_file>`

Additional Options

The following table describes additional options that you can set for an EMR cluster:

Property	Description
Root device EBS volume size (GB)	Number of GB of the EBS root device volume. Enter a value between 10 and 100. Default is 10.
Tags	Optional. Tags to propagate to cluster EC2 instances. Tags assist in identifying EC2 instances. Format: `TagName1=TagValue1,TagName2=TagValue2`
Bootstrap Actions	Optional. Actions to perform after EC2 instances are running, and before applications are installed. Type the JSON statement here, or provide a path to a file that contains a JSON statement. Format: `file:\\<path_to_policy_config_file>`
Custom AMI ID	Optional. ID of a custom Amazon Linux Amazon Machine Image (AMI). Copy the value from the AWS console.
Security Configuration	Optional. The name of a security configuration for authentication and encryption on the cluster. Amazon EMR supports server-side encryption (SSE) and client-side encryption (CSE) configurations. You can use the following at-rest security configurations: - SSE with Amazon S3-managed keys (SSE-S3) - SSE with AWS KMS-managed keys (SSE-KMS) - CSE with AWS KMS-managed keys (CSE-KMS) - Custom CSE configurations* You can use the following in-transit security configurations: - PEM - Custom in-transit configurations* You can also use custom AMIs for local disk security. ** To use custom security configurations, manually copy the .jar file to the Data Integration Service machine.*
Applications	Optional. Applications to add to the default applications that AWS installs. AWS installs certain applications when it creates an EMR cluster. In addition, you can specify additional applications. Select additional applications from the drop-down list. This field is equivalent to the Software Configuration list in the AWS EMR cluster creation wizard.

Property	Description
Software Settings	Optional. Custom configurations to apply to the applications installed on the cluster. This field is equivalent to the Edit Software Settings field in the AWS cluster creation wizard. You can use this as a method to modify the software configuration on the cluster. Type the configuration JSON statement here, or provide a path to a file that contains a JSON statement. Format: `file:\\<path_to_custom_config_file>`
Steps	Optional. Commands to run after cluster creation. For example, you can use this to run Linux commands or HDFS or Hive Hadoop commands. This field is equivalent to the Add Steps field in the AWS cluster creation wizard. Type the command statement here, or or provide a path to a file that contains a JSON statement. Format: `file:\\<path_to_command_file>`

Advanced Properties for Azure Databricks

Set general and advanced options for advanced properties of the Azure Databricks Create Cluster task.

General Options

The following table describes the general options that you can set for a Databricks cluster:

Property	Description
Cluster Name	Name of the cluster to create.
Databricks Runtime Version	The Databricks version to run on the cluster. Default is the latest supported version. To manually enter a version, select *(Assign to Task Input)*. The Developer tool creates a Create Cluster Task Input called *Databricks Runtime Version*. Set the value of the input to the Databricks version tag string. For example, set the value to `5.5.x-scala2.11` to use Databricks version 5.5.
Python Version	The Python version to run if you include a Python transformation. Default is the latest supported version.
Warm Pool	Check this option to enable creation of the Databricks cluster using a warm pool. Default is unchecked (false).
Pool ID	Value of the DatabricksInstancePoolId property that identifies the warm pool resource. To retrieve the value of this property, go to the warm pool in the Databricks workspace and browse to **Configuration** > **Tags**. Required when the Warm Pool option is checked.
Driver Type	The type of node that you want to use for the driver node. Default is the worker type ID.
Worker Type	The type of node that you want to use for the worker node.
Workers	The number of worker nodes to create for the cluster. If you configure the cluster to scale automatically, this property is ignored. Default is 1.

Property	Description
Autoscale	Automatically scales the number of worker nodes based on workload.
Min Workers	The minimum number of worker nodes to use when the cluster is configured to scale automatically. Default is 0.
Max Workers	The maximum number of worker nodes to use when the cluster is configured to scale automatically. Default is 1.

Advanced Options

Configure advanced options such as environment variables and automatic termination.

The following table describes the advanced options that you can set for an Azure Databricks cluster:

Property	Description
Auto Termination	Enables automatic termination of the cluster.
Auto Termination Time	Terminates the cluster after it is inactive for the specified number of minutes. Enter a value between 10 and 10,000. If you do not configure this, or if you set to 0, the cluster will not automatically terminate.
Cluster Log Conf	The location to deliver logs for long-term storage. If configured, the Databricks Spark engine will deliver the logs every five minutes. Provide the path to DBFS.
Init Scripts	The location where you store init scripts. You can enter multiple destinations. The scripts are run sequentially in the order that you configure them. If you need to install additional Python libraries, specify the init script file location in this property. Use the following format: `dbfs:/<path to init script>,dbfs:/<path to init script>`
Cluster Tags	Labels that you can assign to resources for tracking purposes. Enter key-value pairs in the following format: <key1>=<value1>,<key2>=<value2>. You can also provide a path to a local file that contains the key-value pairs. Use the following format: `file:\\<file path>`
Spark Configurations	Performance configurations for the Databricks Spark engine. Enter key-value pairs in the following format: key1='value1' key2='value2'. You can also provide a path to a file that contains the key-value pairs.
Environment Variables	Environment variables that you can configure for the Databricks Spark engine. Enter key-value pairs in the following format: key1='value1' key2='value2'

Advanced Properties for AWS Databricks

Set the advanced properties for an AWS Databricks cluster.

General Options

The following table describes the general options that you can set for a Databricks cluster:

Property	Description
Cluster Name	Name of the cluster to create.
Databricks Runtime Version	The Databricks version to run on the cluster. Default is the latest supported version. To manually enter a version, select *(Assign to Task Input)*. The Developer tool creates a Create Cluster Task Input called *Databricks Runtime Version*. Set the value of the input to the Databricks version tag string. For example, set the value to `5.5.x-scala2.11` to use Databricks version 5.5.
Python Version	The Python version to run if you include a Python transformation. Default is the latest supported version.
Warm Pool	Check this option to enable creation of the Databricks cluster using a warm pool. Default is unchecked (false).
Pool ID	Value of the DatabricksInstancePoolId property that identifies the warm pool resource. To retrieve the value of this property, go to the warm pool in the Databricks workspace and browse to **Configuration** > **Tags**. Required when the Warm Pool option is checked.
Driver Type	The type of node that you want to use for the driver node. Default is the worker type ID.
Worker Type	The type of node that you want to use for the worker node.
Workers	The number of worker nodes to create for the cluster. If you configure the cluster to scale automatically, this property is ignored. Default is 1.
Autoscale	Automatically scales the number of worker nodes based on workload.
Min Workers	The minimum number of worker nodes to use when the cluster is configured to scale automatically. Default is 0.
Max Workers	The maximum number of worker nodes to use when the cluster is configured to scale automatically. Default is 1.

Advanced Options

Configure advanced options such as automatic termination and on-demand instances.

The following table describes advanced options that you can set for an AWS Databricks cluster:

Property	Description
Enable autoscaling local storage	Enables Databricks to monitor available disk space on worker nodes and automatically add additional EBS volumes.
EBS Volume Type	The type of volume that Databricks can add to cluster nodes. Set this property when you enable autoscaling.
Number of Volumes	The number of volumes to provision for each instance. Enter a value between 0 and 10. Set this property when you enable autoscaling and configure EBS volume type.
Size in GB	The size in gigabytes of each EBS volume. Set this property when you enable autoscaling and configure EBS volume type.
Auto Termination	Enables automatic termination of the cluster.
Auto Terminate Time	Terminates the cluster after it is inactive for the specified number of minutes. Enter a value between 10 and 10,000. If you do not configure this, or if you set to 0, the cluster will not automatically terminate.
On-Demand/Spot Composition	The number of on-demand nodes. Enter a value between 0 and the number of worker nodes set in General Options. Any remaining worker nodes will be spot instances. On-demand nodes are always available to use. Spot instances might terminate running jobs if they become unavailable. The driver node is always an on-demand node. Set this property when you enable **Spot fall back to On-Demand**. Default is 1.
Spot fall back to On-Demand	Enables on-demand instances to be used as a fallback. If you are using spot instances and the market price for spot instances surges above your spot bid price, AWS terminates the spot instances. When you enable this property, on-demand instances are used in place of the spot instances when they terminate.
Availability Zone	The AWS cluster availability zone. Default is us-east-1e.
Spot Bid Price	The maximum percent of the on-demand instance price that you bid on spot instances. Spot instances are priced as a percentage of the on-demand price and are not always available. If the market price for spot instances surges above the bid price set here and you do not enable **Spot fall back to On-Demand**, AWS terminates the spot instance. Default is 100%.
IAM Role ARN	The instance profile ARN (Amazon Resource Name) that corresponds to the AWS IAM (Identity and Access Management) role. Copy the value from the AWS console in the following format: `arn:aws:iam::<account-id>:instance-profile/<role-name>` IAM roles allow you to access data from Databricks clusters. Add new IAM roles in the Administrator tool.

Property	Description
Spark Configurations	Performance configurations for the Databricks Spark engine. Enter key-value pairs in the following format: key1='value1' key2='value2'. You can also provide a path to a file that contains the key-value pairs.
Environment Variables	Environment variables that you can configure for the Databricks Spark engine. Enter key-value pairs in the following format: key1='value1' key2='value2'
Cluster Tags	Labels that you can assign to resources for tracking purposes. Enter key-value pairs in the following format: <key1>=<value1>,<key2>=<value2>. You can also provide a path to a local file that contains the key-value pairs. Use the following format: `file:\\<file path>`
SSH Public Key	The SSH public key to log into the driver and worker instances if you enable SSH. Copy the value from the Databricks console.
Cluster Log Conf	The location to deliver logs for long-term storage. If configured, the Databricks Spark engine will deliver the logs every five minutes. Provide the path to DBFS.
Init Scripts	The location where you store init scripts. You can enter multiple destinations. The scripts are run sequentially in the order that you configure them. If you need to install additional Python libraries, specify the init script file location in this property. Use the following format: `dbfs:/<path to init script>,dbfs:/<path to init script>`

Delete Cluster Task

The Delete Cluster task terminates and deletes a Hadoop cluster on a cloud platform.

If you do not add a Delete Cluster task in a cluster workflow, the cluster remains running when the workflow ends.

A cluster that you create and then terminate when jobs are complete is called an ephemeral cluster.

Set the following properties for the Delete Cluster task:

- In the General properties, optionally rename the Delete Cluster task.
- In the Advanced properties, set the Task Recovery Strategy property to Restart task.

CHAPTER 5

Command Task

This chapter includes the following topics:

Command Task Overview

A Command task runs a single shell command or starts an external executable program during the workflow.

You might specify a shell command to delete reject files, copy a file, or archive target files. You can use workflow parameters and variables in the command.

When you run a workflow, the workflow passes input data to a Command task in parameters and variables. The Command task uses the input data to run the command. When the Command task finishes running, the task passes output data back to the workflow in variables.

When you configure a Command task, you specify the command to run, the input data needed by the task, and the output data that the task passes to the workflow. You also set the advanced properties that the task uses when it runs the command.

Command Syntax

The command syntax depends on whether the Data Integration Service runs on UNIX or Windows.

When the Data Integration Service runs on UNIX, you can use any valid UNIX command or shell script. The service runs the following command during the workflow where `<command>` is the command that you enter in the Command task:

```
/bin/sh -c "<command>"
```

When the Data Integration Service runs on Windows, you can use any valid DOS or batch file. The service runs the following command during the workflow where `<command>` is the command that you enter in the Command task:

```
cmd.exe /c "<command>"
```

For example, you might use a shell command to copy a file from one directory to another. For Windows, enter the following shell command to copy the SALES_ ADJ file from the source directory, L, to the target, H:

```
copy L:\sales\sales_adj H:\marketing\
```

For UNIX, enter the following command to perform a similar operation:

```
cp sales/sales_adj marketing/
```

Use the following rules and guidelines when you enter a command:

- The command cannot contain a carriage return character or line feed character.
- To run an external executable program from the command, enter the fully qualified path to the program. For example, to run a custom application named myCustomApp.exe, use the following command:

  ```
  c:\myCustomApp.exe
  ```

  When you run an external program from the Command task, the task remains in a running state until the program closes.
- Each Command task runs in the same environment as the Data Integration Service. To change the environment settings, create a script or batch file that changes the settings and includes the command to run. Then use the following command to run the file:

  ```
  c:\mybatfile.bat
  ```

Parameters and Variables in a Command

You can include workflow parameters and variables in a command.

You can select a workflow parameter or variable from **Inputs** tab in the **Command** tab, or you can type the parameter or variable name in the command using the required syntax.

For example, the following command uses a workflow parameter named SourceDirectory to define the source directory from which the command copies a file:

```
copy ${par:SourceDirectory} H:\marketing\
```

Related Topics:

- "Parameter Names in Expressions and Strings" on page 40
- "Variable Names in Expressions and Strings" on page 33

Command Task Input

Command task input is the data that passes into a Command task from workflow parameters and variables.

Assign a Command task configuration property to task input to define the value of the property in a workflow parameter or variable. The **Advanced** tab lists the Command task configuration properties.

For example, you assign the working directory to the same workflow parameter for all Command tasks in the workflow. You want each Command task to run the same command executable. In the parameter file, you set the workflow parameter value to the directory where the executable is located in the development environment. When you deploy the workflow to the production environment, each Command task must use the command executable located in the production environment. You change the value of the workflow parameter in the parameter file instead of editing each Command task in the Developer tool.

Note: The Command task does not validate the working directory path. In the workflow context, a Command task can run successfully when the working directory path is not valid.

Related Topics:

- "Assigning Workflow Parameters to Task Input" on page 39
- "Assigning Variables to Task Input" on page 32

Command Task Output

Command task output is the data that passes from a Command task into workflow variables. Command task outputs include general outputs and task specific outputs.

When you configure a Command task, you specify the task output values that you want to assign to workflow variables on the **Output** tab. The Data Integration Service copies the Command task output values to workflow variables when the Command task completes.

If the task fails, the Data Integration Service copies the general task output values but not the task specific output values to workflow variables. If the task aborts, the Data Integration Service does not copy any task output values to workflow variables.

For example, a Command task produces an exit code output value that indicates whether the command ran successfully. The workflow cannot directly access this Command task output data. To use the data in the remainder of the workflow, you assign the exit code output to a workflow variable named CommandExitCode. Then use the CommandExitCode workflow variable in an expression for a conditional sequence flow. The Data Integration Service runs the next object in the workflow if the command ran successfully.

General outputs include output data produced by all tasks such as the task start time, end time, and whether the task successfully ran.

Command task outputs include data produced by the Command task when it runs the command.

The following table describes the output data produced by the Command task:

Output Data	Datatype	Description
Exit Code	Integer	Exit code returned by the command. A successful command returns 0. An unsuccessful command returns a non-zero value.
Standard Error	String	Standard error message returned by the command. By default, the first 1,024 characters of the error are returned. You can change the length of the standard error in the Command task advanced configuration properties.
Standard Output	String	Standard output returned by the command. By default, the first 1,024 characters of the output are returned. You can change the length of the standard output in the Command task advanced configuration properties.

Related Topics:

- "Task Output" on page 27
- "Assign a Value with Task Output" on page 31

Command Task Advanced Properties

The **Advanced** tab for a Command task includes properties that the task uses to run the command.

Configure the following Command task advanced properties:

Task Recovery Strategy

Determines whether the Data Integration Service reruns or skips a task that is interrupted or that encounters an error. If the workflow is not enabled for recovery, the Data Integration Service ignores the task recovery strategy.

By default, the task has a restart recovery strategy.

Working Directory

Directory from which to run the command. You can also specify the working directory in the command that you enter for the task. If you specify the directory in both locations, the directory specified in the command overrides the directory specified in this property.

Enter a value for the Working Directory. Or, assign the Working Directory property to task input. Then in the Command task **Input** tab, you can assign the property to a workflow parameter or variable. The property has a string datatype.

Default is the home directory of the Data Integration Service. The Command task does not validate the working directory path.

Standard Output Length

Number of characters to return in the standard output for the command. Maximum is 32,768 characters.

Default is 1024 characters.

Standard Error Length

Number of characters to return in the standard error for the command. Maximum is 4096 characters.

Default is 1024 characters.

Troubleshooting Command Tasks

The solution to the following situation might help you troubleshoot Command tasks.

The command run by a Command task was unsuccessful, but the Data Integration Service continues running the next object in the workflow.

A Command task can succeed even if the command is unsuccessful or if you provide a working directory that is not valid. The Is Successful general output indicates whether the Command task ran successfully. The Exit Code task output indicates whether the command ran successfully. You can use a conditional sequence flow to check if both the Command task succeeded and if the command returned an exit code of 0.

For example, create a boolean workflow variable that captures the Is Successful output returned by the Command task. Create an integer workflow variable that captures the exit code returned by the command. Create the following expression in the outgoing conditional sequence flow from the Command task:

```
$var:CommandTaskSuccessful = true AND $var:CommandExitCode = 0
```

The Data Integration Service runs the next task if the previous Command task ran successfully and if the command returned an exit code of 0.

CHAPTER 6

Human Task

This chapter includes the following topics:

Human Task Overview

A Human task defines actions that one or more users perform on the workflow data. Create a Human task when you want Analyst tool users to analyze the output of a mapping that runs in a Mapping task.

A Mapping task can identify records in a data set that contain unresolved data quality issues. The records may contain errors, or they may contain duplicate information. The Human task organizes the records into one or more task instances that Analyst tool users can work on. The Analyst tool users resolve the issues in the records and update the data quality status of the records.

When you configure a Human task, you create one or more task steps. A task step represents the type of work that a user performs on the records in a task instance. You can assign all task data to a single user, or you can create multiple task instances so that different users can work on different records concurrently.

A Human task starts when a Mapping task ends. The workflow uses the *exceptionLoadCount* output variable from a Mapping task to define the data for a Human task. When all task instances are complete, the data passes to the next stage in the workflow.

Human Tasks and Exception Data Management

A Human task reads the output of a mapping that contains an Exception transformation. An Exception transformation analyzes output from other transformations to validate the data quality status of the records in a data set. A mapping developer uses the Exception transformation to identify records that need manual processing.

The Exception transformation writes records to one or more database tables based on the data quality status of each record. The transformation specifies a table as a target for records with an unverified data quality status. The objective of the user in a Human task is to verify the data quality status of the records in the table.

When the mapping developer completes the mapping that contains the Exception transformation, a workflow developer adds the mapping to a Mapping task in a workflow. When you add a Human task to the workflow, you configure the Human task to read the database table created when the Mapping task runs. The users who perform the Human task examine the records and makes any change required.

The users then update the status of the records in one of the following ways:

- If a record is valid, the user updates the table metadata so that the record is confirmed for persistent storage in the database.
- If a record is not valid, the user updates the table metadata so that the record is removed from the database at a later stage in the workflow.
- If the status of a record cannot be confirmed, the user updates the table metadata so that the record is returned to the workflow for further processing in a Mapping task.

Types of Exception Data

The Exception transformation generates database tables that contain records with an unverified data quality status. The Human task user examines each record and attempts to resolve any issue the record contains.

A record can have the following types of data quality issue:

- The record may contain errors or empty cells. The Human task user examines the records and attempts to update the record with correct and complete data.
- The record may be a duplicate of another record. The Analyst tool displays duplicate record sets in groups called **clusters**. The Human task user examines the clusters and attempts to create a single preferred version of the records in each cluster.

The user can apply the following status indicators to a record or cluster:

- The record or cluster issues are resolved, and the record can remain in the database. In the case of clusters, the preferred record remains in the table and the redundant duplicate records are dropped.
- The record or cluster issues are unresolved and the record needs further processing.

The record or cluster contains unusable data and can be dropped from the table.

Analyst Tool

The Analyst tool is a web-based application that enables users to view and update records and clusters in a Human task.

The Analyst tool uses an Inbox to notify users of the Human tasks assigned to them. A user logs in to the Analyst tool and opens a task from the **My Tasks** panel.

The Analyst tool provides options to edit record or cluster data and to update the status of a record or cluster. The task view includes metadata columns that contain the status indicators for each record or cluster.

When a user completes a task in the Analyst tool, the records in the task pass to the next step in the Human task.

Human Task Instances

When you add a Human task to a workflow, you configure one or more instances of the task. You assign users to the task instances, so that each user works on part of the data set. The Data Integration Service creates the task instances when the workflow runs.

Use the task distribution options in the Human task to create multiple instances of the task. The task distribution options determine the size and composition of the data in each task instance. For example, if the input data set for the Human task contains one thousand records, you can create five task instances with two hundred records in each instance.

You can create task instances in the following ways:

- Specify the number of task instances to create. The Data Integration Service divides the input data set into task instances of equal size.
- Specify the number of rows or clusters to include in a task. The Data Integration Service creates task instances that contain the number of rows or clusters you specify. The number of task instances depends on the number of rows or clusters in the input data set.
- Specify the data rows or clusters to assign to each user. You select an input data column, and you identify the users who can work on each task instance. The Data Integration Service assigns rows or clusters to users based on the values in the column you select.

If you add multiple steps to a Human task, the data associated with each task instance passes through every step in the Human task. You select the users who work on the task instance in each step. You can assign the same set of users to all steps in the Human task, or you can assign different users to each step.

Human Task Steps

A Human task contains one or more steps. A step is a blueprint for the task instances that the Analyst tool users must work on to complete the Human task. When a workflow runs, the step identifies the data to assign to each task instances and defines the type of work that an Analyst tool user must perform on the data.

You can add the following steps to a Human task:

- Cluster step. Configure a Cluster step when you want a user to examine duplicate record clusters and create a preferred record from the values in the duplicate records. The Analyst tool identifies the tasks that a cluster step defines as tasks to correct duplicates.
- Exception step. Configure an Exception step when you want a user to examine and fix errors in records. The Analyst tool identifies the tasks that an exception step defines as tasks to correct exceptions.
- Review step. Configure a Review step when you want a user to review the work done in an Exception or Cluster step. Review steps are not mandatory. The Analyst tool identifies the tasks that a review step defines as tasks to review duplicates or review exceptions.

You can add a Cluster step or Exception step to a Human task, but you cannot add both. The database tables in cluster steps and exception steps have different structures.

You can add steps in any order. For example, you can add a Review step before or after a Cluster or Exception step.

Human Task Roles

When you create a Human task and configure the steps and task instances, you specify the users or groups who can work on the task. You assign different roles to the users and groups.

You can assign the following roles to users and groups:

- Task performer. A user or group that you select to work on task instances in the Analyst tool.
- Business administrator. A user or group that manages the status of the task instances associated with a Human task or with a step in the task. If a task performer is unable to complete a task instance on schedule, a business administrator reassigns the task to another task performer.

You identify business administrators at the Human task level. You can optionally identify business administrators at the step level. Any business administrator you identify in a Human task is also a business administrator in each step in the Human task. When you define a business administrator in a step, the business administrator status applies within the step only.

You can identify task performers at the Human task level and the step level. Identify task performers at the Human task level when you want to assign users or groups to task instances based on the data that the task instances contain. When you identify task performers at the Human task level, the task performers work on the same data for the duration of the Human task. When you define a task performer in a step, the task performers work on tasks within the step only.

You can assign a user and a group with the same name to a Human task. For example, you can assign the group Domain/Administrator and the user Domain/Administrator to the same Human task.

Human Task Properties

When you add a Human task to a workflow, the task properties appear on the **Properties** view of the workflow.

You configure options on the following tabs:

- General tab. Set the name and text description for the Human task.
- Participants tab. Identify the business administrators who can participate in the task.
- Data source tab. Identify the database connection name and the database resource that contains the Mapping task output.
- Task distribution tab. Determine the size and composition of task instances.
- Notifications tab. Identify the users who receive email notifications when the Human task is complete, and write the email text.
- Input tab. Set the variables that identify the input data for the task.
- Output tab. Set the variables that provide the final status of the task to the workflow.
- Advanced tab. Set the task recovery strategy.

General Tab

The General tab identifies the Human task. Enter a name and an optional description for the Human task. You can also open the Human task from the General tab.

The following table describes the options on the General tab:

Property	Description:
Name	The name of the Human task.
Description	Optional. The description of the Human task.
Task	A link to the Human task repository object. Click the Task link to open the Human task in the Developer tool.

Participants Tab

Use the Participants tab options to identify one or more business administrators for the task instances that the Data Integration Service creates for the Human task. A business administrator can also work on the task instances in the Analyst tool.

The following table describes the option on the Participants tab:

Property	Description
Name	Name of the user or group that you identify as a business administrator for the Human task. To select a business administrator, click **Select**.

You can assign business administrators to the Human task, and you can assign business administrators to a step in a Human task. A business administrator on a Human task can perform task management actions on any task instance that the Data Integration Service creates for the Human task. A business administrator on a step can perform task management actions on any task instance that the Data Integration Service creates for the step.

A business administrator on a Human task can also complete all of the task instances associated with the Human task in a single operation. To complete the tasks, the business administrator runs the infacmd wfs completeTask command on the command line.

Data Source Tab

Configure the options on the Data Source tab to identify the tables that contain the task data.

The following table describes the options on the Data Source tab:

Property	Description
Connection	The database connection name for the database that contains the Mapping task output.
Resource	The name of the database table that contains the Mapping task output. The user who performs the task connects to the database and works on the data in the table. You can select a database table synonym as the resource if the table resides in an Oracle database or an IBM DB2 database.

Task Distribution Tab

Set the options on the Task Distribution tab to create instances of the Human task. Use the task distribution options when you want to assign a task to multiple users. You can create task instances of equal size, or you can create task instances based on the data in a column.

The following table describes the options on the Task Distribution tab:

Property	Description
Enable task distribution	Enables options to create multiple instances of the Human task. If you clear this option, the workflow processes a single instance of the Human task.
Divide task by number of items	Creates task instances with the number of records or clusters that you specify, or creates the number of task instances that you specify.
Divide task by data value	Creates task instances based on the values in a column that you select. You select the column from the data source that the **Data Source** tab identifies. When the workflow runs, the Data Integration Service creates a task instance for each set of records that share a common value in the column. You identify the users or groups who can work on the task instances, and you specify a value to associate with each user or group. You can optionally associate each user or group with a range of numeric values or date values. The workflow assigns the tasks instances to users or groups based on the associations that you define. Identify the users or groups who will work on the task instances in the following ways: - Enter the users or groups to a grid, and enter the column values to associate with each user or group. - Import the user or group names and the column values from a reference table in the Model repository. - Import the user or group names and the column values from a local delimited file. - Connect to an external database table that contains a list of user or group names and a list of data values. When you connect to an external database, you can update the list of users, groups, and values independently of the workflow configuration. The Data Integration Service reads the table when the workflow runs. Optionally, you can identify a user or group to receive task instances that contain records that do not meet the task distribution criteria.

Related Topics:

- "Create Task Instances" on page 78

Notifications Tab

Set the options on the Notifications tab to notify users or groups when the Human task is complete. The workflow sends an email notice to the users or groups that you identify.

Before you can configure a Human task to send emails, an administrator must enable and configure the Email Service in the Administrator tool.

The following table describes the options on the Notifications tab:

Property	**Description**
Task Events	Lists the types of status change for which a user or group can be notified. At the workflow level, you can select the Complete option.
Recipients	Lists one or more users or groups to notify when the Human task is complete.
Subject	Lists the contents of the subject line for the email notifications. You can edit the subject line.
Mime Type	Specifies the content type for the email. Select one of the following values: - Plain. Send a message in plain text. - HTML. Send HTML content. You can include lists and hyperlinks in HTML content. The HTML view provides basic a HTML structure by default. If you edit the HTML, you cannot restore the default structure.
Message	Displays the html tags that define the email content. Type the body text for the message between the <body> tags.
Preview	Displays the email content as it will appear to the recipients.

Input Tab

Human task input is the data that passes into the Human task from a workflow variable.

The Input tab shows a single option named **Number of items processed**. Use the option to set the *exceptionLoadCount* variable.

The *exceptionLoadCount* variable stores the number or records or clusters in the database table that you specify on the Data Source tab. The variable indicates the number of records or clusters that are processed in the Human task.

Note: You do not set input parameters on a Human task.

Related Topics:

- "Assigning Variables to Task Input" on page 32

Output Tab

Human task output is the data that passes from a Human task into workflow variables. Human task outputs include general outputs. General outputs include output data produced by all tasks such as the task start time, end time, and whether the task successfully ran.

When you configure a Human task, you specify the task output values that you want to assign to workflow variables on the Output tab. The Data Integration Service copies the Human task output values to workflow variables when the Human task completes.

For example, a Human task produces a start time output value that indicates when the Data Integration Service started running the task. The workflow cannot directly access the Human task output data. To use the data in the remainder of the workflow, you assign the start time output to a workflow variable that you name HumanTaskStartTime. Then you use the HumanTaskStartTime workflow variable in an expression for a conditional sequence flow. The Data Integration Service runs the next object in the workflow if the Human task started before the specified time.

RELATED TOPICS:

- "Task Output" on page 27
- "Assign a Value with Task Output" on page 31

Advanced Tab

The Advanced tab for a Human task includes the task recovery strategy.

A Human task specifies a restart recovery strategy. The restart option is read-only. The Data Integration Service always restarts an interrupted Human task when a workflow recovers.

If the workflow is not enabled for recovery, the Data Integration Service ignores the task recovery strategy.

Step Properties

When you add a step to a Human task, you identify the users who can work on the step and you set the duration of the step. Use the Properties view to configure a step.

You configure the following options for the step:

- General. Sets the name and text description for the step.
- Participants. Specifies the users who will work on the tasks that the Data Integration Service creates for the step.
- Permissions. Specifies permissions on task data and metadata. The permissions apply to the users who will work on the tasks for the step.
- Configuration. Review steps only. Identifies the Exception step or Cluster step that users will review.
- Timeout. Sets the time frame within which all users must complete task instances for the current step. You set the timeout value as a period of minutes, hours, and days. The timeout period begins when the workflow runs.

  Note: The Human task uses the timeout period to calculates a deadline for all task instances associated with the step. A user who performs a task instance sees the deadline date and time and not the timeout period.
- Notifications. Identifies the users who can be notified when a task instance associated with the step changes status.

General Options

Set the General options to identify the step in the Human task. Enter a name and an optional description for the step.

The following table describes the general options of a step:

Property	Description:
Name	The name of the step in the Human task.
Description	Optional. The description of the step.

Configuration Options

Use the Configuration options to specify how a task performer can update a value in a task that corrects or reviews exception records. Additionally, use the configuration options in a Review step to identify the step that passes task data to the current step for review.

The following table describes the configuration options of a step:

Property	Description
Step to review	Identifies the step that passes data to the current step for review. Select the step from the menu in a Review step.
Disallow empty error cells	Specifies whether the task performer must update any null or empty cell that the task identifies as an exception. The option applies to any user who works on the task. By default, the task performer can change a cell value to an empty or null value and can complete a task that identifies a null or empty cell as an exception. To specify that the task performer must update every exception with a data value, select the option to disallow empty error cells. **Note:** The option does not apply when the task does not grant edit permission to the task performer.

Participants Options

Use the Participants options to identify the users or groups who can work on the task instances that the current step defines. You can select users or groups to act as task performers and business administrators.

The following table describes the participant options of a step:

Property	Description
Task Performer	Identifies the users or groups who can work on the task instances that the Data Integration Service creates for the current step. The Data Integration Service ignores the task performers on the step when you specify the following configuration options: - You select a step that defines task instances to correct exceptions or duplicate records. - You configure the Task Distribution properties in the Human task to divide tasks by data value. The Task Distribution properties that you set identify the users and groups who can work on the task instances that the step defines.
Business Administrator	Identifies the users or groups who can perform task management operations for the task instances that the Data Integration Service creates for the current step. Business administrators can also work on task instances. You can also select business administrators when you configure the Human task. Any business administrator that you select for the Human task is a business administrator for the task instances that each step in the Human task defines.

Permissions Options

Set the Permissions options to specify the data that users can view and the types of action that users can perform in the Analyst tool. The permissions apply to all users who can view or edit a task instance that the current step defines.

You can set viewing permissions and editing permissions. The viewing permissions define the task instance data that the Analyst tool displays. The editing permissions define the actions that users can take to update record data or cluster data. The permissions that you set do not affect the users' ability to view or update status information for a record, cluster, or task instance.

Exception steps and cluster steps support different types of permissions. The permissions that you set in a review step depend on whether the preceding step is an exception data step or a cluster step.

Consider the following rules and guidelines when you set permissions on a step:

- The viewing permissions take precedence over the editing permissions. If a user cannot view a column of data in a task instance, the user cannot edit the data in the column.
- If a user cannot view a data column in a task instance, the user cannot view the column in the audit trail.
- The step that you configure might read the output from another step in the Human task. Set permissions that allow Analyst tool users to view and edit the appropriate data from the task instances that the preceding step generates.

Viewing Permissions

The following table describes the viewing permissions that you can set in each step:

Permission Value	**Description**	**Step Type**
View all data	Analyst tool users can view all of the data in a task instance. Default option.	All step types.
View selected columns	Analyst tool users can view data in the columns that you select in a task instance.	All step types.

Editing Permissions

The following table describes the editing permissions that you can set in each step:

Permission Value	**Description**	**Step Type**
Edit all data	Analyst tool users can edit all of the visible data in the task instance. Default option on steps that read exception data.	Exception step Review step for exception data
Edit selected columns	Analyst tool users can edit the data in one or more columns that you select if the columns are visible in the task instance.	All step types
Edit exception data only	Analyst tool users can edit any visible value in the task instance that the workflow identifies as a data quality issue.	Exception step Review step for exception data
Perform all actions	Analyst tool users can perform all actions on the visible data in the task instance. Default option on steps that read cluster data.	Cluster step Review step for cluster data

Permission Value	Description	Step Type
Perform cluster actions only	Analyst tool users can define a preferred record in each cluster in a task instance. Analyst tool users can create clusters and move records from one cluster to another. Analyst tool users cannot the edit data values in the cluster records.	Cluster step Review step for cluster data
Perform review actions only	Analyst tool users can perform any action to update the status of the task instance and the records or clusters in the task instance. Analyst tool users cannot edit the data values or the preferred records in the task instance.	Review step for exception data Review step for cluster data

Related Topics:

- "Rules and Guidelines for Step Permission Options" on page 81

Timeout Options

Use the Timeout options to set a time frame during which the task performers must complete the task instances that the step defines. If a task instance does not complete in the time frame that you specify, the Analyst tool lists the task as overdue. The workflow can reassign overdue tasks to users or groups that you specify in the step.

The following table describes the timeout options of a step:

Property	Description
Duration	The time period for completion of all task instances associated with the step. The time period begins when the Human task creates the task instances. Specify a time period in days, hours, and minutes.
Name	Lists the users or groups who can receive task assignments. If you expect that more than one task will fail to complete on time, consider adding multiple users or groups to the list.
Reassign Task	Indicates whether a user or group will receive a reassigned task. If you clear the **Reassign Task** option, the workflow does not reassign overdue tasks to the user or group.

Notifications Options

Set the Notifications options to notify users or groups when a task instance that a step defines changes status. The Human task sends an email notice to the users or groups that you identify.

Before you can configure a Human task to send emails, an administrator must enable and configure the Email Service in the Administrator tool.

The following table describes the notification options of a step:

Property	Description
Task Events	Lists the types of status change for which a user can be notified. You can choose one of the following values: - Create. A Human task creates a task instance associated with the step. - Complete. A user completes a task instance. - Escalated. A user or group does not complete a task instance on time. - Reassign. A business administrator reassigns a task instance to another user. **Note:** You can configure the Notifications options to send a different notification for each task event. For example, you can notify a user when a Human task creates the task instance, and you can notify another user when the task instance is complete. You can also enter a different email subject and message for each task event.
Recipients	Lists one or more users to notify for the status change that you select. You can select or clear the option to notify the task owner in addition to any recipient that you select when the task instance changes status. The option applies when a single user owns the task instance. When you select the option to notify the task owner, you can optionally leave the Recipients field empty.
Subject	Lists the contents of the subject line for the email notifications. You can edit the subject line.
Mime Type	Specifies the content type for the email. Select one of the following values: - Plain. Send a message in plain text. - HTML. Send HTML content. You can include lists and hyperlinks in HTML content. The HTML view provides basic a HTML structure by default. If you edit the HTML, you cannot restore the default structure.
Message	Displays the email content as you type.
Preview	Displays the email content as it will appear to the recipients.

Workflow Variables in Task Instance Notifications

You can use workflow variables to write information about a Human task instance to an email notification. The variables store information about the task instance when a user completes, escalates, or reassigns a task instance. To display the list of variables, click in the subject line or the notification message body and press the $+CTRL+SPACE keys.

The notification can display the following variables:

$taskEvent.eventTime

The time that the workflow engine performs the user instruction to escalate, reassign, or complete the task instance.

$taskEvent.owner

The owner of the task instance at the time that the workflow engine escalates or completes the task. Or, the owner of the task instance after the engine reassigns the task instance.

$taskEvent.status

The task instance status after the engine performs the user instruction to escalate, reassign, or complete the task instance. The status names are READY and IN_PROGRESS.

$taskEvent.taskEventType

The type of instruction that the engine performs. The variable values are escalate, reassign, and complete.

$taskEvent.taskId

The task instance identifier that the Analyst tool displays.

Human Tasks and Workflow Configuration

When you configure a workflow with a Mapping task and Human task, you can add a second Mapping task to reunite records processed in the Human task with records from the previous Mapping task.

For example, you can use the following steps to configure a workflow to manage exception records:

1. You create a workflow and you add a Start event and End event.
2. You create a mapping *Mapping_1* that examines a data set for duplicate records. The mapping contains an Exception transformation that writes good-quality records to a database table named *Good_Records* and writes exceptions to a table named *Exceptions*.
3. You add a Mapping task to the workflow, and you configure the task to run *Mapping_1*.
4. You add a Human task to the workflow, and you configure the task to assign the data in the *Exceptions* table to users.
5. You configure a mapping *Mapping_2* to read the *Exceptions* table and to write records to the *Good_Records* table if the Human task updated the record with an Approved status.
6. You add a second Mapping task to the workflow, and you add *Mapping_2* to the Mapping task.

   You configure a Notification task to send an email message to selected users. The email message states that the second Mapping task is complete.
7. You connect the events and tasks in the workflow.

When the workflow runs, the first mapping creates the Exceptions table and the second mapping writes the good-quality records from the *Exceptions* table to the *Good_Records* table.

Human Task Configuration

A Human task is composed of task instances and steps.

You create task instances to divide the data set among multiple users. You create steps to define the actions that users must complete. You identify the users who will work on task instances when the task data reaches the step.

Configuring Task Instances in a Human Task

You configure task instances at the Human task level in a workflow.

Use the Human task **Properties** view to configure the task options. If you cannot see the Properties view, select **Window** > **Show View** > **Properties.**

1. Open a workflow in the Developer tool, and add a Human task.
2. In the **General** tab, verify the Human task name and optional description.
3. In the **Participants** tab, identify the business administrators who will manage the task instances.

4. In the **Data Source** tab, specify the database connection information for the database that stores the exception records.
5. In the **Task Distribution** tab, configure the options that the workflow uses to assign data to task instances.
6. In the **Notifications** tab, identify the users to notify when the Human task are complete, and write the email notification.
7. In the **Input** tab, define the Human task input.

   Set the input to read the *exceptionLoadCount* variable from the Mapping task in the workflow.
8. Optionally set the output to write task information to one or more variables in the **Output** tab.

   The outputs indicate the start time, end time, and whether the task successfully ran.

Configuring Task Steps

You configure one or more steps in a Human task. You can add Exception steps or Cluster steps. Optionally create Review steps to verify the work in the Exception or Cluster steps.

1. Open a Human task from a workflow.
2. In the **General** tab, verify the step name and optional description.
3. In the **Participants** tab, select the users or groups who can work on tasks in the step.

   You identify the users or groups who perform the tasks, and you identify the business administrators to notify about the task assignments.
4. In the **Timeout** tab, set the time period in which the tasks must complete.
5. In the **Notifications** tab, identify the users to notify when a task instance associated with the step changes status.
6. In the **Configuration** tab, select the Exception or Cluster step to review. The Configuration tab appears in Review tasks only.

Create Task Instances

You use the task distribution options to create instances of a Human task.

You can configure the options to create task instances based on the number of items in the data set, or you can configure options to create task instances based on the values in a column you select.

If you select the option to create task instances by text, you define a user list that you must use when you configure the task steps. If you select the option to create task instances by number of items, you do not define a user list.

Note: If you change from one task configuration policy to another, you discard the previous task configuration.

Creating Task Instances of Equal Size

You can create task instances based on the number of records or clusters in the data set. You can also specify the number of task instances to create. In each cases, you create tasks of equal size.

1. Open a Human task.

2. Select the **Task Distribution** tab, and enable task distribution.
3. Choose to create task instances by number of items.
4. Set the number of tasks to create, or set the number of rows or clusters in each task.

Creating Task Instances by Data Value

You can create task instances that contain all the records in the data set that have common values in a column that you select.

1. Open a Human task.
2. Select the **Task Distribution** tab, and enable task distribution.
3. Choose to create task instances by data value.

   Optionally, specify the maximum number of records that any task instance can contain.
4. Select a column name from the **Column** menu. The menu lists the column names on the resource that you specify on the Data Source tab.

   If you add a Cluster step to the Human task, select the group key column that you used in the mapping that generated the clusters. You select the group key column to ensure that the task distribution process does not split the records in a cluster into different clusters.

   Note: The Column menu has a precision of 65. The menu does not display a column name that contains more than 65 characters.
5. Select the method that the workflow will use to assign users or groups to task instances.

   You can associate each user or group with a single value in the column, or you can associate each user or group with a range of column values.
6. Create the assignments between the users or groups and the column data values.

   You can add the users or groups and the values to associate with them to a grid in the **Task Distribution** tab. Or, you can connect to an external database table that stores the lists of users, groups, and column values.
7. To add values to the grid, select one of the following options:
   - **Add Data Value**. Enter a value, and select a user or group from the domain.
   - **Add Data Value from Reference Table**. Import data values and user or group names from a reference table. Create the reference table with a column of user or group names and a column of data values.
   - **Add Data Value from Local File**. Import data values and user or group names from a delimited file.
8. To add values from an external database table, browse to a database connection and select a database table. resource from the database.
9. Optionally, select a user or group to receive any record in the data source that does not satisfy the task distribution criteria that you specify.

Rules and Guidelines for Task Distribution By Column Values

Consider the following rules and guidelines when you specify data values and user or group names for task distribution:

- You can enter the data values and the user or group names in a grid in the **Task Distributio**n tab. Or, you can import the data values and the user or group names from a table or a file.
- Enter each combination of data values and user or group names on a single row.
- To associate a value with a user or group, enter a single value in the first column and a single user or group in the second column.

To associate a range of numbers with a user or group, enter the lower value in the first column and the upper value in the second column. Then, enter the user or group name in the third column.

To associate a range of dates with a user or group, enter the earlier date in the first column and the later date in the second column. Then, enter the user or group name in the third column.

- You can select a table that contains more than three columns. The task configuration uses the first two columns if you specify single data values or the first three columns if you specify ranges of values. The task configuration disregards additional columns.
- The lower and upper values correspond to the Start and End values in the **Task Distribution** tab. The earlier and later dates correspond to the Start and End dates.
- On each row, the Start value must be lower than or equal to the End value. The Start date must be lower than or equal to the End date.
- The workflow treats each range as inclusive. For example, a range of 50 through 99 includes the values 50 and 99. The ranges that you set cannot overlap.
- The users and groups that you specify must exist in the Informatica domain in which the workflow runs. The data values in the table or file must match the values in the data source column that you select. Include the security domain name with the user or group name. For example, enter *Native\Jefferson* to identify the user name Jefferson in the Native domain.
- If a data value contains a delimiter, use quotation marks as text qualifiers to enclose the data value.
- You can associate a column value or a range of values with more than one user or group. When the workflow runs, the Data Integration Service assigns any task that contains the value or values to the first user or group in the distribution list.
- If a record contains a value that does not match the distribution criteria, the Data Integration Service adds the record to a unique task instance. The Data Integration Service assigns the instance to a user or group that you specify on the **Task Distribution** tab.
  You can select any user or group in the domain, including any user or group that you select to receive other tasks.

Setting Permissions on a Step

Enter a short description of the task here (optional).

1. Open a Human task.
2. Select the **Permissions** tab.

   The tab displays the options for viewing permissions and editing permissions.
3. To set each permission, open the menu in the **Value** field. Use the menu options to select the permission that you need.
   - When you opt to set viewing permissions on selected columns, the **Visible Columns** dialog box opens. Use the dialog box options to select the columns to show or hide in the Analyst tool.
   - When you opt to set editing permissions on selected columns, the **Editable Columns** dialog box opens. Use the dialog box options to select the columns that users can edit in the Analyst tool.

     Note: Define editing permissions on one or more of the visible columns. Analyst tool users cannot edit data in a column that they cannot view.

Related Topics:

- "Rules and Guidelines for Step Permission Options" on page 81
- "Permissions Options" on page 73

Rules and Guidelines for Step Permission Options

By default, Analyst tool users can view all data and perform any action in the task instances that they own. Use the **Permissions** options to specify a set of viewing permissions and editing permissions for the task instances. Apply the permissions in the step that defines the task instances.

You can use permissions to assign data to users based on the users' roles or areas of expertise. For example, you might define a series of steps in a Human task so that different users can edit different columns of workflow data. Or, you might decide to hide sensitive data, such as salaries or Social Security numbers, from one or more users.

Consider the following rules and guidelines when you set the permissions:

- You can define different permissions in each step in a Human task. An Analyst tool user who reviews a task instance might see a different data set to a user who worked on the data in an earlier task instance.
- The permissions apply to the audit trail data in addition to the record or cluster data in each task instance.
- If a business administrator transfers a task instance to another user, the permissions that you set for the step apply to the user that the business administrator identifies.
- The viewing permissions that you set on a step take precedence over the editing permissions. If you grant editing permissions on a column and you do not grant viewing permissions on the column, Analyst tool users cannot edit the column data.
- If you find and replace multiple values in a task instance, the Analyst tool restricts the operation to the data that the viewing permissions specify for the instance.
- Analyst tool users can perform any action to update the status of a record, cluster, or task instance regardless of the viewing or editing permissions that you set.

Related Topics:

- “Permissions Options” on page 73

CHAPTER 7

Mapping Task

This chapter includes the following topics:

Mapping Task Overview

A Mapping task runs a mapping in a workflow.

When you add a Mapping task to a workflow, you select a single mapping for the task to run. You can select any mapping that has physical data objects as input and output. When you change the mapping, the Model Repository Service tracks the effects of these changes on all Mapping tasks that include the mapping.

When you run a workflow, the workflow passes input data to a Mapping task in parameters and variables. The Mapping task uses the input data to run the mapping. When the task finishes running the mapping, the Mapping task passes output data back to the workflow in variables.

When you configure a Mapping task, you specify the input data for the task and the output data. You can set parameters that create a run-time context for the mapping task. Set task properties for the location and file name for the mapping log file. Set other properties at the mapping task level, such as the optimizer level, default sort order, and Java classpath.

After you configure a Mapping task, you can select a different mapping for the task to run.

Multiple Mapping Tasks that Run the Same Mapping

You can include multiple Mapping tasks that contain the same mapping in a single workflow.

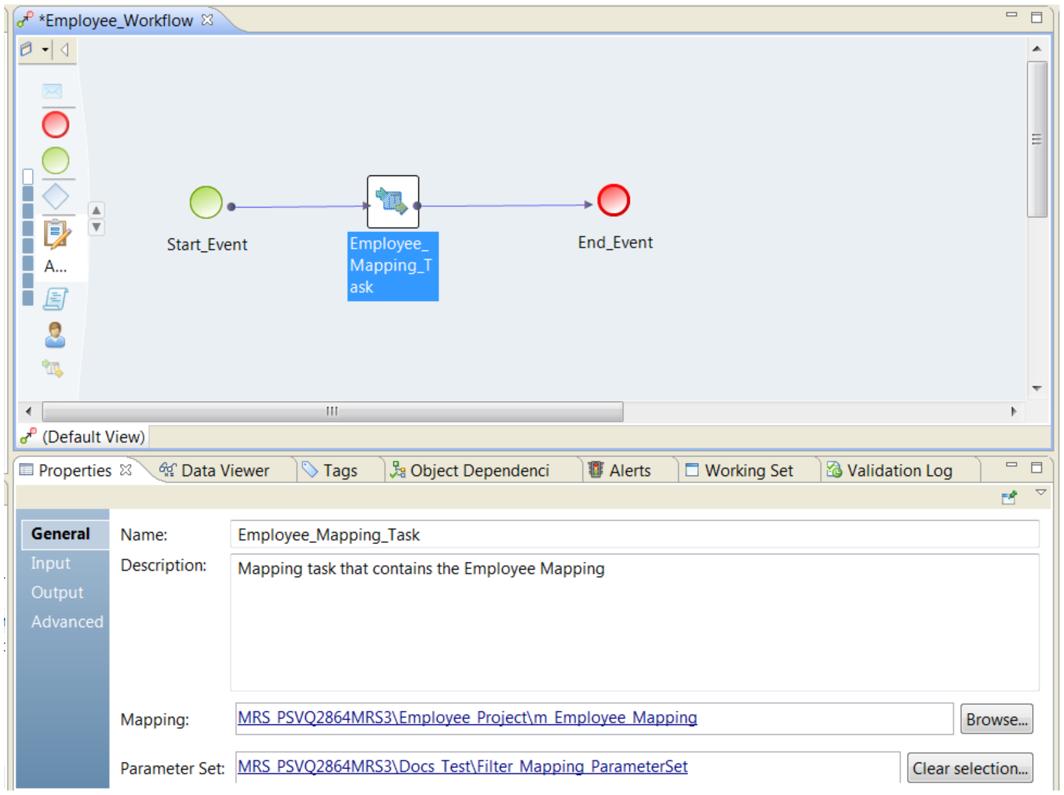
When a workflow includes multiple Mapping tasks that contain the same mapping, you can configure the Mapping tasks to run the mapping differently. You can also assign a workflow parameter or variable to a user-defined mapping parameter in a Mapping task so that you can run the mapping with distinct parameter values.

For example, you add two Mapping tasks that each contain MappingA to a single workflow. You configure the first Mapping task to run MappingA with a connection to the Sales database. You configure the second Mapping task to run MappingA with a connection to the Finance database.

Mapping Task General Properties

You can change the mapping in a Mapping task. You can also add a parameter set to the mapping task.

The following image shows the General view for a Mapping task:



Name: Employee_Mapping_Task

Description: Mapping task that contains the Employee Mapping

Mapping: MRS_PSVQ2864MRS3\Employee_Project\m_Employee_Mapping

Parameter Set: MRS_PSVQ2864MRS3\Docs_Test\Filter_Mapping_ParameterSet

To view the **General** view, click the Mapping task in the workflow. Click the **General** view.

You can change the following fields in the **General** view:

Name

Name of the Mapping task.

Description

Description of the Mapping task.

Mapping

The name and the path to the mapping to include in the Mapping task. Click **Browse** to select a different mapping in the repository.

Parameter Set

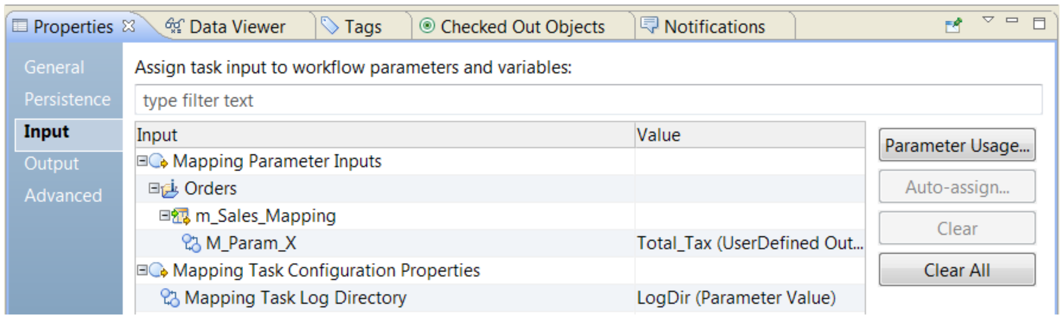
Name and path of the parameter set to use in the Mapping task. Click **Browse** to search for a parameter set. Click **Clear Selection** to remove the parameter set. To view the parameter set, click the link to the parameter set.

Mapping Task Input

Mapping task input is the data that passes into a Mapping task. A Mapping task can receive input data from mapping parameters or from Mapping task configuration properties.

Assign values to Mapping task input on the Mapping task **Input** tab.

The following image shows the Mapping task **Input** tab:



Assign task input to workflow parameters and variables:

type filter text

Input	Value
Mapping Parameter Inputs	
Orders	
m_Sales_Mapping	
M_Param_X	Total_Tax (UserDefined Out...
Mapping Task Configuration Properties	
Mapping Task Log Directory	LogDir (Parameter Value)

You can assign values to the following types of Mapping task inputs:

Mapping Parameter Inputs

Mapping parameters are parameters that you define at the mapping level. The **Input** tab lists all the parameters for a mapping, including the parameters that you did not assign to a field in the mapping.

Mapping task configuration properties

Mapping task configuration properties are properties that you configure on the **Advanced** tab. When you need to assign a parameter to a Mapping task configuration property, you can select the **Assign to Task Input** option for the property. When you choose this option, the configuration property appears on the **Input** tab. You can then assign a value to it.

To assign a value to Mapping task inputs, click the **Value** column for a parameter or configuration property. A list of the mapping outputs, workflow variables, and parameters appears. You can select an output, variable, or parameter from the list, or you can create a new value.

Related Topics:

- "Assigning Workflow Parameters to Task Input" on page 39
- "Assigning Variables to Task Input" on page 32

Mapping Parameters and Mapping Tasks

The Data Integration Service can use user-defined mapping parameter values when it runs a mapping within a workflow. You can use a parameter set or a parameter file for the workflow.

When you run a workflow, you can specify a parameter set at run time. The parameter set contains user-defined parameter values. You must add a parameter set to the application when you deploy the workflow. You can include multiple parameter sets when you deploy the workflow and run the workflow with a different parameter set for each run.

You can also add a parameter set to a Mapping task. When you add a parameter set to a Mapping task, the parameter set overrides the mapping parameter values or the workflow parameter values that you defined.

When you run a workflow, you can specify a single parameter file for the workflow. You define user-defined mapping parameter values within a mapping or mapplet element in the parameter file. Define workflow parameter values within a workflow element in the parameter file.

When you run the workflow with a parameter file or a parameter set, the Data Integration Service applies all the user-defined mapping parameter values and workflow parameter values.

Override Mapping Parameters During a Workflow Run

You can override user-defined mapping parameter values during a workflow run by assigning user-defined mapping parameters to workflow parameters or variables in the Mapping task **Input** tab.

You cannot assign system mapping parameters used in the mapping to workflow parameters. If the mapping has a user-defined mapping parameter assigned to a flat file directory field, cache file directory field, or temporary file directory field, then you can assign the user-defined mapping parameter to a workflow string parameter.

You might want to override user-defined mapping parameter values for the following reasons:

Use run-time data for the user-defined mapping parameter value.

Assign a user-defined mapping parameter to a workflow variable when you want to use workflow run-time data for the parameter value. For example, use a workflow string variable to override a mapping string parameter that defines a flat file name. Use an Exclusive gateway to evaluate a condition and then run Assignment task A or Assignment task B. Assignment task A sets the workflow variable to FlatFileA.txt. Assignment task B sets the workflow variable to FlatFileB.txt. In the Mapping task input, assign the workflow variable to the mapping string parameter. When the Mapping task runs the mapping, the mapping uses the value dynamically assigned to the workflow variable for the source file name.

Assign distinct values to a user-defined mapping parameter that is used multiple times in the workflow.

When you include multiple Mapping tasks that run the same mapping in a single workflow, assign a user-defined mapping parameter to a distinct workflow parameter for each Mapping task. Define a distinct value for each workflow parameter in a parameter file. When you run the workflow with the parameter file, the mapping run by each Mapping task uses the value of the assigned workflow parameter.

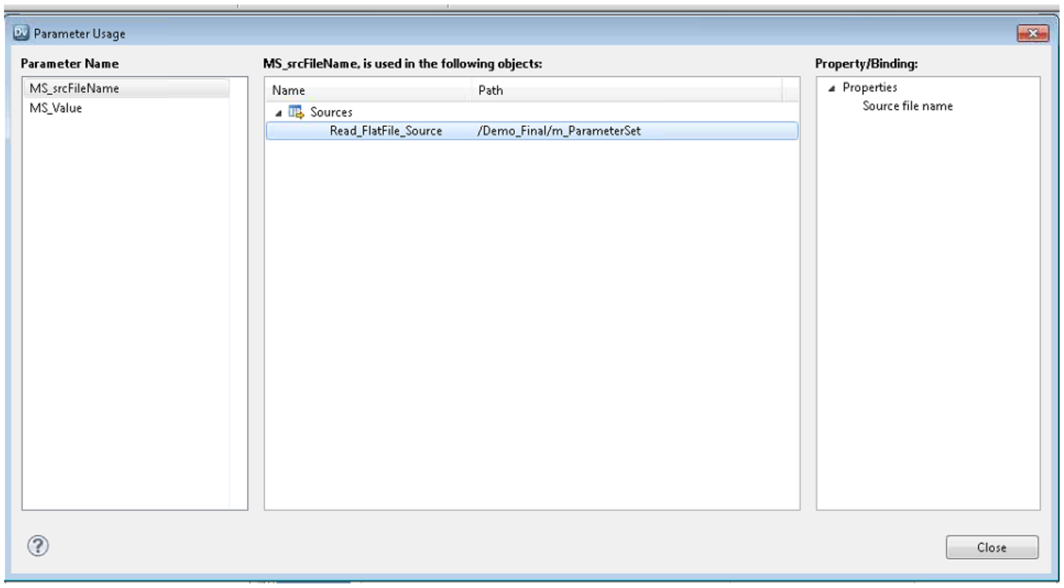
For example, a workflow includes two Mapping tasks that contain the same mapping that uses the user-defined mapping parameter SourceConnection. When the workflow runs, you want to use a different source connection for each mapping. You assign the mapping parameter SourceConnection to a distinct workflow connection parameter for each Mapping task. For MappingTask1, you assign SourceConnection to the workflow parameter WF_SourceConnection1. For MappingTask2, you assign SourceConnection to the workflow parameter WF_SourceConnection2. Assign a parameter set to Mapping task. In the parameter set, configure distinct values for the workflow parameters WF_SourceConnection1 and WF_SourceConnection2.

Parameter Usage in a Mapping Task

You can view a list of the objects in a Mapping task that use a mapping parameter.

To view the objects in a Masking task that use a mapping parameter, click **Parameter Usage** from the **Input** view of a Mapping task. The **Parameter Usage** dialog box shows all the mapping parameters in a Mapping task. Select a mapping parameter to view the objects that use the parameter.

The following image shows the **Parameter Usage** dialog box:



MS_srcFileName
MS_Value

Name	Path
Sources	
Read_FlatFile_Source	/Demo_Final/m_ParameterSet

Properties
Source file name

The **Parameter Usage** dialog box contains the following fields:

Parameter Name

The names of the mapping parameters in the mapping.

Name

Object type and the name of the object that uses the mapping parameter.

Path

The path to the object in the repository. Includes the folder name and the mapping name.

Property/Binding

Shows which property is bound to the mapping parameter. The mapping parameter might be bound to a user-defined parameter or to a specific field. When a mapping parameter is bound to a field or to a user-defined parameter, the mapping uses the mapping parameter value instead of the other value.

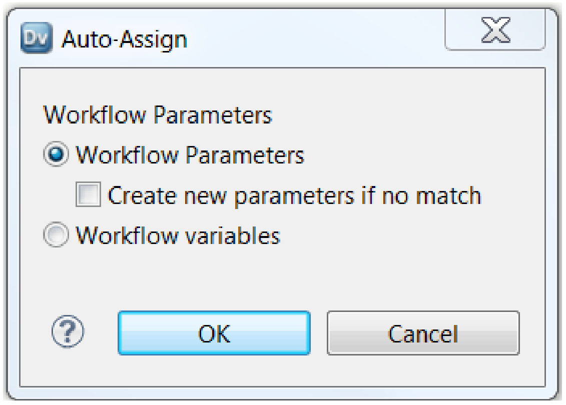
Auto-Assign Mapping Input Parameter Values

You can automatically assign mapping parameters to workflow parameters or workflow variables on the Mapping task **;** tab.

Choose the **Auto-Assign** option to generate a workflow parameter or a workflow variable to bind to a mapping parameter in one step. You can also assign mapping parameters to workflow parameters or to workflow variables that you created previously.

To auto-assign a mapping parameter, select the mapping parameter in the **Input** column and click **Auto-Assign**.

The following image shows the Auto-Assign dialog box:



Choose from the following options to assign the mapping parameter to a workflow parameter or to a workflow variable:

Workflow Parameters

Assign the mapping parameter to a workflow parameter. A workflow parameter name must exist with the same name as the mapping parameter name. The Developer tool does not match the parameters by parameter type. If the workflow has no parameter to assign the mapping parameter to, you can choose to create new parameters if no match.

Create new parameters if no match

Generate a workflow parameter with the same name as the selected mapping parameter. Workflow parameters are string types or connection types. If you create a workflow parameter from a mapping parameter that is not a string or a connection type, the Developer tool creates a string workflow parameter. The Developer tool binds the string workflow parameter to the mapping parameter.

Workflow Variables

Assign the mapping parameter to a workflow variable. The workflow variable must exist and the workflow variable name must be the same as the mapping parameter name. You cannot use the Auto-Assign option to create a workflow variable.

Clear Assignments

You can clear the value assignments for a mapping input. The value assignment is the assignment of a mapping input to a workflow parameter, workflow variable, or system parameter.

To clear the value assignment for one mapping input, select the mapping parameter in the **Input** column. Click **Clear**. Confirm that you want to remove the value assignment. The Developer tool clears the value column for the mapping input you selected.

To remove all the mapping input assignments, click **Clear All**. Confirm that you want to remove all the mapping input value assignments. The Developer tool clears the value column for all the mapping inputs.

Mapping Task Output

A mapping output in a Mapping task is a value that provides information about the Mapping task run.

You can bind a mapping output to a workflow variable to use the output in another workflow task. You can also persist the value of the output to use the value as an input parameter the next time that the Mapping task runs.

Mapping tasks return the following types of output:

General Outputs

The start time, end time, and whether the task succeeded.

System-defined Outputs

The number of error rows, source rows, and target rows that the mapping processed.

User-defined Outputs

Mapping outputs that you define to aggregate fields or expressions from each row in the mapping. Persisted user-defined outputs are values that the Data Integration Service saved to the repository from a previous workflow run. Current values are mapping output values from the current workflow run.

Perform the following tasks with mapping outputs:
Persist the mapping output in the repository.

You can configure a Mapping task to persist a mapping output value in the repository, or you can assign a persisted mapping output value to a Mapping task input.

For example, you can configure the Mapping task to return the last sequence number that it generated. You can persist the mapping output that stores the sequence number in the repository. The next time the Mapping task runs, you can use the last sequence number as the starting sequence number.

Bind the output to a workflow variable.

You can bind mapping outputs from the current Mapping task run to workflow variables. If you bind the output to a workflow variable, the workflow passes the value of the output to other tasks in the workflow. You can also bind persisted mapping outputs from a previous Mapping task run to workflow variables in the current run.

RELATED TOPICS:

- "Task Output" on page 27
- "Assign a Value with Task Output" on page 31

System-Defined Mapping Outputs

A mapping or transformation can return system-defined outputs. You can use the mapping output in a subsequent run of the workflow or by the next task in the workflow.

You can persist the mapping output in the repository or bind the mapping output to a workflow variable. You can use the mapping output in a subsequent run of the workflow or pass the value to another task in a workflow, such as a Notification task or an Exclusive Gateway task.

For instance, a transformation can return a system-defined output, numberOfErrorRows, that contains the number of records that were invalid. You might review this information to monitor a process or to understand the quality of the data that a mapping processes.

The following table describes the output data produced by the Exception transformation in the mapping:

Transformation	Output Data	Datatype	Description
Exception	exceptionLoadCount	Integer	Number of records that contain unresolved data quality issues and require a manual review. The records may contain errors, or they may contain duplicate information. The Developer tool lists a single instance of the exceptionLoadCount output even if the mapping contains multiple Exception transformations. Assign the exceptionLoadCount output to a workflow variable if the mapping contains a single Exception transformation. Otherwise, unexpected behavior may occur.

Persist Mapping Outputs

You can persist mapping outputs to save the output values in the repository and use the values in subsequent runs of the same Mapping task. You can also assign persisted mapping outputs from a previous Mapping task run to workflow variables for the current Mapping task run.

When you click the **Persistence** tab in the Mapping task **Properties** view, the Developer tool displays all the mapping outputs for the mapping. To persist any mapping output, enable **Persist** for the mapping output and select the type of aggregation to perform to return a persisted value.

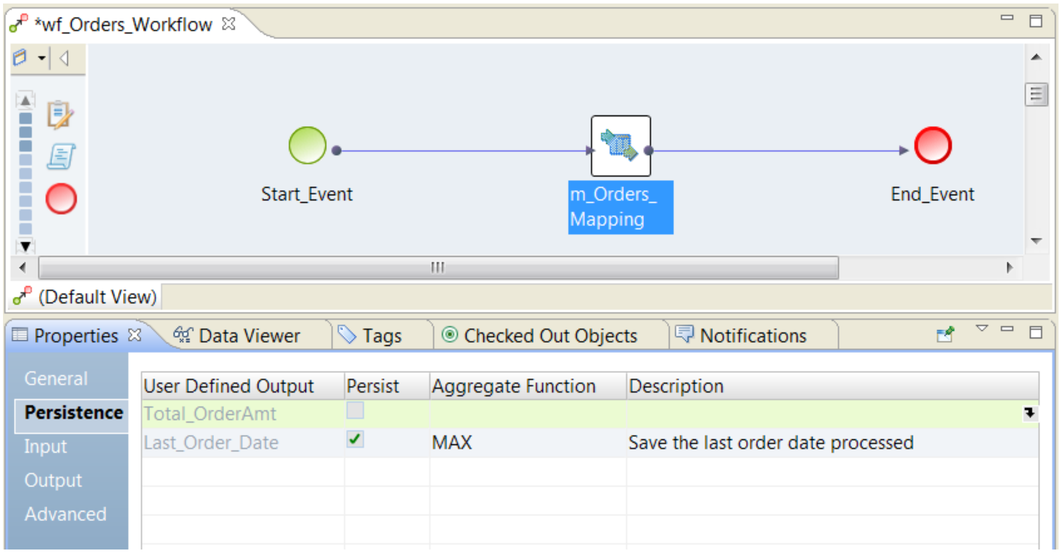
When the Data Integration Service persists a mapping output in the Model repository, the Data Integration Service saves the mapping output with the Mapping task name as a key. For example, if a workflow contains four mapping tasks, each running the same mapping, the Data Integration Service saves four outputs in the Model repository.

When you persist a mapping output, you can configure a different aggregate function for the persisted value than the aggregate function you defined at the mapping level. The Data Integration Service generates more than one mapping output value. For example, the OrderDate mapping output might contain the MIN OrderDate. The persisted OrderDate mapping output might contain the MAX OrderDate.

You can bind the mapping output from a Mapping task to the input parameter of the Mapping task the next time it runs. You must persist the mapping output in a Mapping task to bind it to an input mapping parameter.

A use case for binding is to persist the latest order date that the mapping processes. The next time the Mapping task runs, the input parameter to the mapping is the last date processed. The mapping can filter the parameter source rows to include the rows with an order date greater than the last order date processed.

The following image shows the **Persistence** tab on the **Properties** view of the Mapping task:



User Defined Output	Persist	Aggregate Function	Description
Total_OrderAmt			
Last_Order_Date	✓	MAX	Save the last order date processed

The **Persistence** tab has the following fields:

User-Defined Output

The name of a mapping output that the mapping returns.

Persist

Enables the Data Integration Service to persist the mapping output in the repository.

Aggregate Function

The type of aggregation to perform on the mapping output to persist. Select MIN, MAX, or SUM. The default is the value from the mapping output that you define in the mapping properties. You can change the aggregate function type of the persisted mapping output. You can persist a different value in the repository than the mapping output value that you pass to workflow variables.

Description

Describes the mapping output to persist in the repository.

Maintaining Persisted Outputs

You can list, update, and reset the persisted mapping outputs in the repository.

You can run the following infacmd commands for persisted mapping task values:

listMappingPersistedOutputs

Lists the persisted mapping outputs and their values for a Mapping task instance in a workflow.

setMappingPersistedOutputs

Updates or resets the persisted mapping outputs for a specific Mapping task instance in a workflow. When you reset the values, you remove the persisted values from the repository. To set mapping outputs enter space-separated name-value pairs of mapping outputs in the command line. To reset mapping outputs use the -reset option with a space-separated list of mapping outputs.

For more information about infacmd, see the *Informatica Command Reference*.

Persisted Mapping Outputs and Deployment

When you redeploy a workflow or you change a mapping output, you can affect the state of persisted mapping outputs.

Consider the following rules and guidelines for persisted mapping outputs:

- When you deploy a workflow as an application for the first time, you do not have to perform any additional tasks if a Mapping task has persisted mapping outputs.
- When you redeploy an application, you can choose whether to retain the state information or to discard it. If you choose to retain the state information, the mapping output values do not change in the repository when you redeploy the application. Otherwise the mapping outputs state is removed from persistence.
- The mapping outputs state is not backed up when you back up and restore a mapping or workflow.
- If you rename or recreate a mapping output, you cannot use the persisted mapping output value from a previous workflow run.

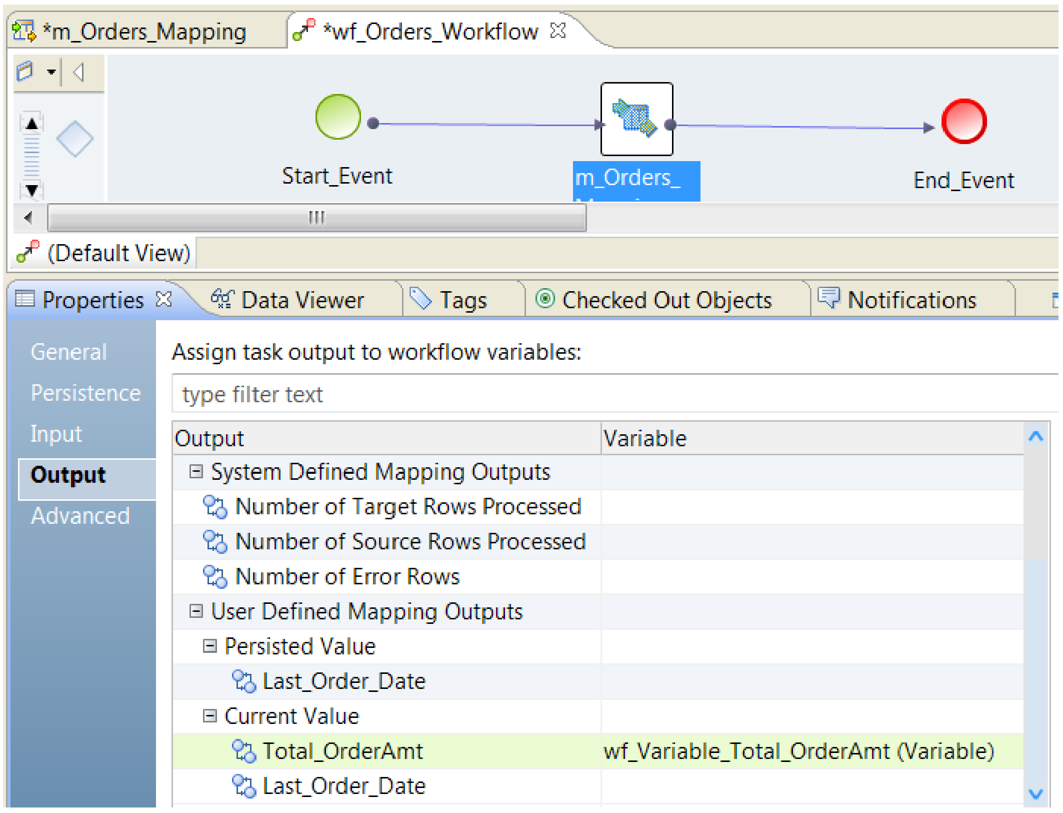
Bind Mapping Outputs to Workflow Variables

Bind mapping outputs to workflow variables and pass the values to other tasks in the workflow.

You might want to bind a mapping output to a workflow variable if you want the Data Integration Service to evaluate the mapping output and determine which object to run next. Or, you might want the Data Integration Service to use the mapping output value in a field in the next task.

Configure the Mapping task output values that you want to assign to workflow variables on the **Output** tab of the Mapping task Properties view. The Data Integration Service copies the Mapping task output values to workflow variables when the Mapping task completes.

The following image shows some mapping outputs on the Mapping task **Output** tab:



Assign task output to workflow variables:

type filter text

Output	Variable
⊟ System Defined Mapping Outputs	
Number of Target Rows Processed	
Number of Source Rows Processed	
Number of Error Rows	
⊟ User Defined Mapping Outputs	
⊟ Persisted Value	
Last_Order_Date	
⊟ Current Value	
Total_OrderAmt	wf_Variable_Total_OrderAmt (Variable)
Last_Order_Date	

The **Output** column contains the following types of mapping outputs:

System-Defined Mapping Outputs

Built-in mapping outputs that transformations return to the mapping. The system-defined mapping outputs contain the number of source rows, the number of target rows, and the number of error rows that the mapping processed.

User-Defined Mapping Outputs

Persisted mapping output values and current mapping output values. You can bind the output values to workflow values.

Persisted Values

The user-defined mapping output values from the previous workflow run. The persisted value is a value that is in the repository from the last time the Mapping task ran. The persisted value is not the value that the current mapping aggregates.

Current Values

The user-defined mapping output values from the current Mapping task.

To assign a workflow variable to a Mapping task output, click the Variable column. Choose a workflow variable to assign the output to. In the example image, the Total_OrderAmt mapping output from the current workflow run is bound to the workflow variable `wf_Variable_Total_orderAmt`.

If a task fails, the Data Integration Service copies the general task output values but not the task specific output values to workflow variables. If the task aborts, the Data Integration Service does not copy any task output value to workflow variables.

Note: If you assign a mapping output to a workflow variable and the mapping processes no rows, the output is NULL. The mapping task does not change the value of the workflow variable. The variable retains the same value as it was before the Mapping task ran.

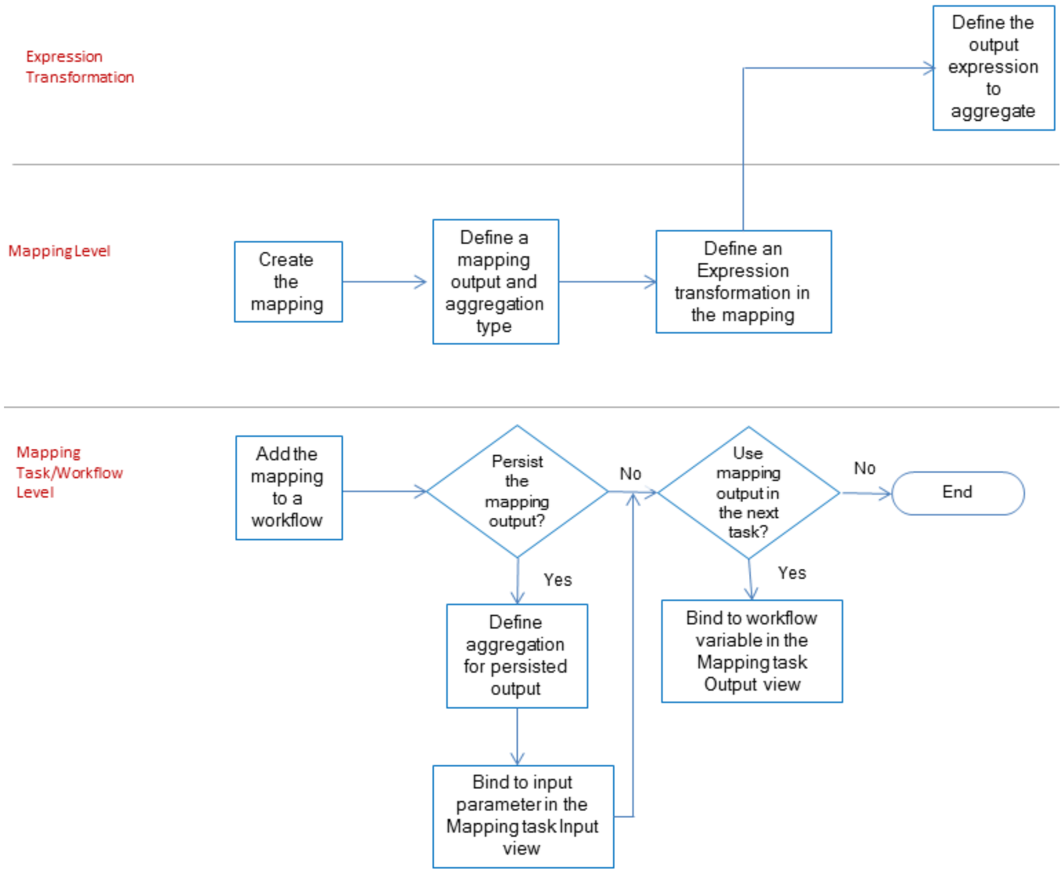
Example

A Mapping task includes a number of error rows output value that captures the number of rows that the mapping failed to process. To use the data in the remainder of the workflow, you assign the number of error rows output to a workflow variable named MappingErrorRows. Then you can add the MappingErrorRows workflow variable in an expression for an outgoing sequence flow from an Exclusive gateway. If MappingErrorRows contains a value greater than zero, the gateway takes one branch. If the value equals zero, the gateway takes the other branch.

How to Configure Mapping Outputs

When you configure mapping outputs, define the mapping outputs at the mapping level, configure the expressions to aggregate at the transformation level, and persist the results at the Mapping task level.

The following image shows the process to configure mapping outputs:


Expression Transformation
Define the output expression to aggregate
Mapping Level
Create the mapping
Define a mapping output and aggregation type
Define an Expression transformation in the mapping
Mapping Task/Workflow Level
Add the mapping to a workflow
Persist the mapping output?
No
Use mapping output in the next task?
No
End
Yes
Define aggregation for persisted output
Yes
Bind to workflow variable in the Mapping task Output view
Bind to input parameter in the Mapping task Input view

To configure mapping outputs, perform the following steps:

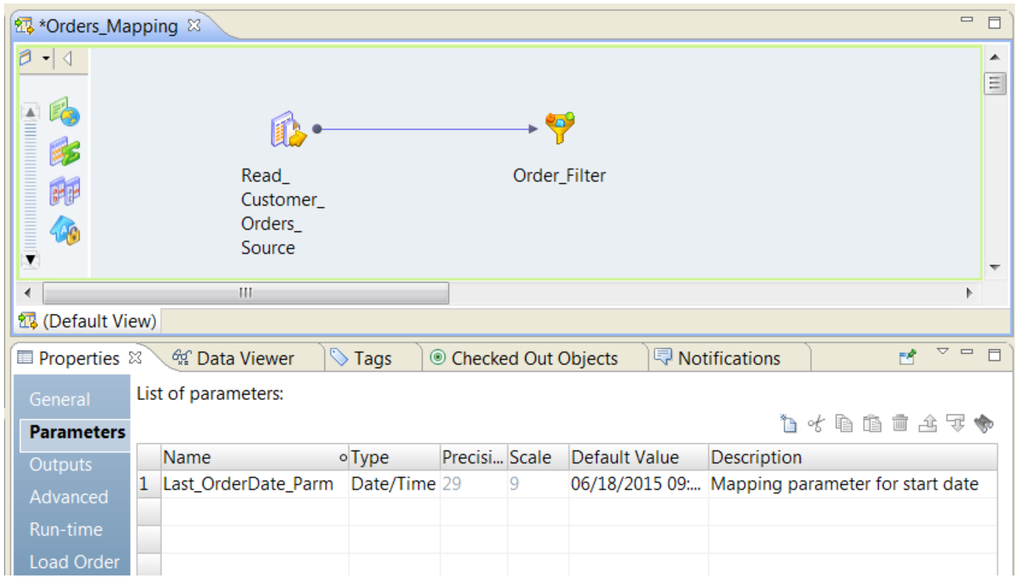
1. Create the mapping.
2. In the **Outputs** view of the mapping, define the mapping output name and type of aggregation.
3. Add an Expression transformation to the mapping and configure the mapping output expression in the Expression **Mapping Outputs** view.
4. To create a Mapping task, add the mapping to a workflow .
5. Persist the mapping output in the Mapping task **Persistence** view and configure the aggregation function type for the persisted value.
6. Assign the persisted mapping output to an input parameter in the Mapping task.
7. If you want to use the mapping output in another workflow task, assign the mapping output to a workflow variable .

Creating a Mapping

Create a mapping that contains a reusable Filter transformation. The Filter transformation filters rows that have order dates less than a specific date. The filter expression includes a parameter called Last_Order_Date_Parm.

1. Create a mapping to process order data from a Customer_Order file.
2. In the mapping Properties view, click the **Parameters** tab.
3. Add a date/time mapping parameter called Last_Order_Date_Parm.

Enter a default date for the starting parameter.
The following image shows the mapping parameter:

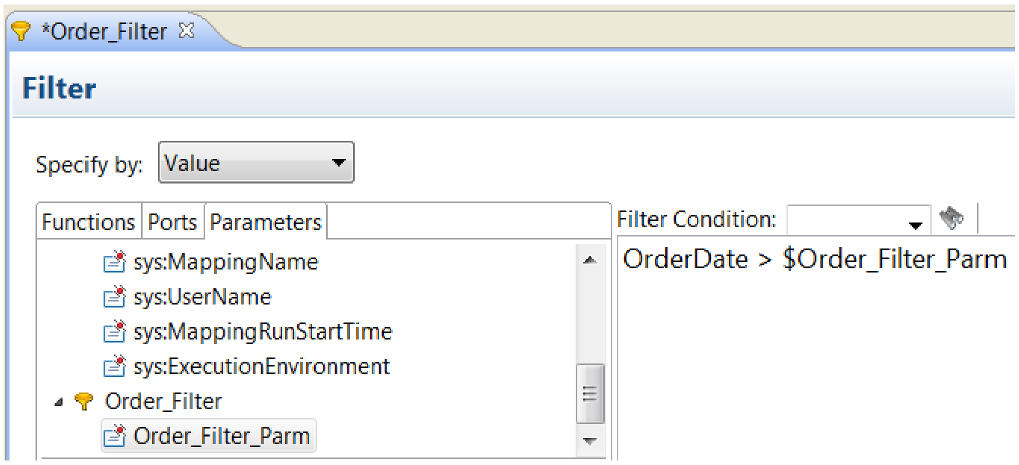


List of parameters:

	Name	Type	Precisi...	Scale	Default Value	Description
1	Last_OrderDate_Parm	Date/Time	29	9	06/18/2015 09:...	Mapping parameter for start date

4. Create a reusable Filter transformation to filter Customer_Order rows.
5. Define a parameter in the Filter transformation called Order_Filter.

   Enter a default date for the starting parameter.
6. Add a filter expression to find order dates that are greater that the parameter:



Specify by: Value

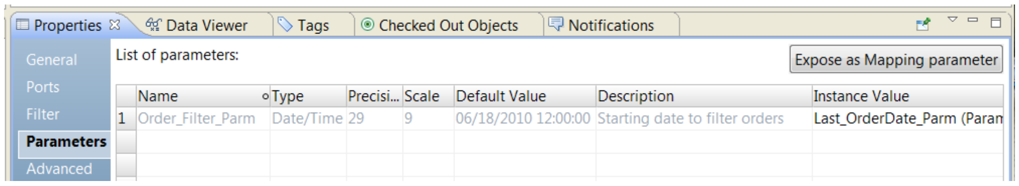
Filter Condition:

OrderDate > $Order_Filter_Parm

7. Add the Filter transformation to the mapping.
8. Click the Filter transformation to display the transformation **Properties** view.
9. Click the **Parameters** tab.
10. To bind the Order_Filter_Parm transformation parameter to the Last_Order_Date mapping parameter, click the **Instance Value column** for the Order_Filter_Parm.

11. Select Last_Order_Date.

    The following image shows where the mapping parameter is bound to the transformation parameter:



List of parameters:

	Name	Type	Precisi...	Scale	Default Value	Description	Instance Value
1	Order_Filter_Parm	Date/Time	29	9	06/18/2010 12:00:00	Starting date to filter orders	Last_OrderDate_Parm (Paran

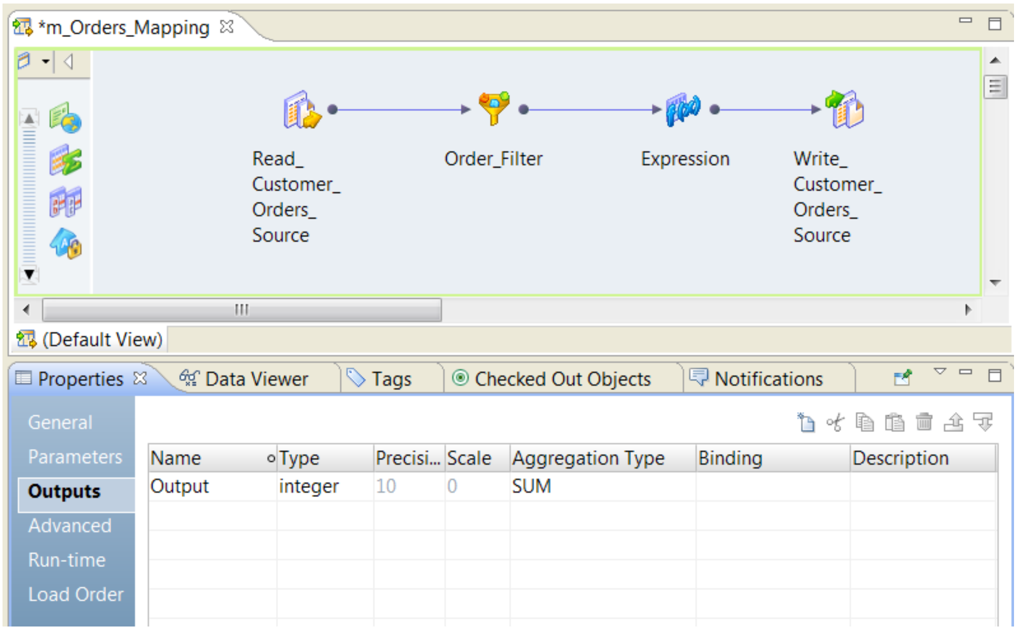
Defining Mapping Outputs

Create a mapping and define the mapping outputs in the mapping **Properties**. Each mapping output definition describes what type of aggregation to perform and the data type of the results.

1. After you create a mapping, click the editor to access the mapping **Properties**.
2. Click the **Outputs** view.
3. Click **New** to create a mapping output.

    The Developer tool creates a mapping output with default field values.

    The following image shows the mapping output default values in the **Outputs** view:



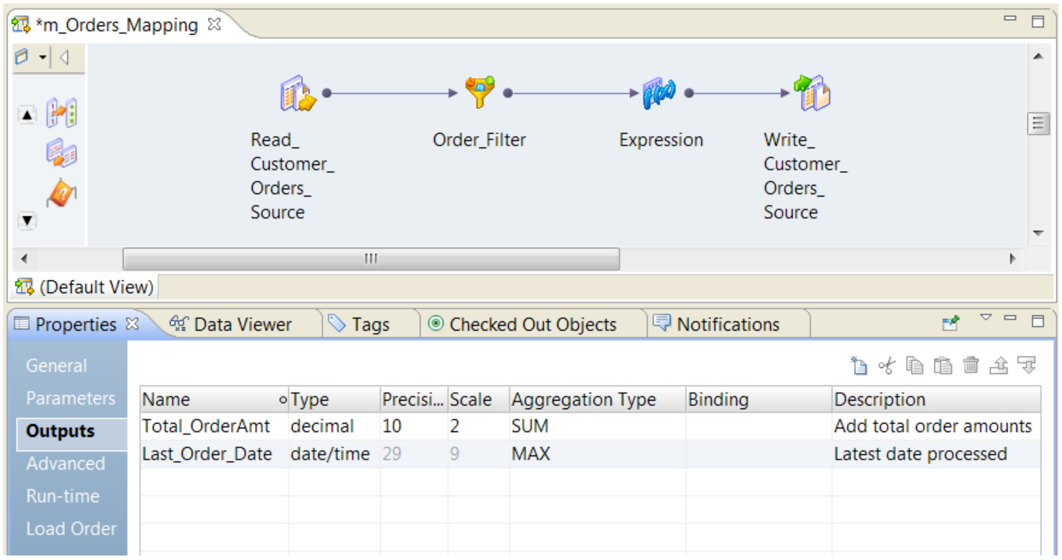
Name	Type	Precisi...	Scale	Aggregation Type	Binding	Description
Output	integer	10	0	SUM		

4. Change the name that identifies the mapping output.
5. Select a numeric or date mapping output type. Enter the precision and scale.
6. Choose the aggregation type for the mapping output.

    You can summarize the output expression or you can find the minimum or maximum expression value that the mapping processed. Default is SUM.
7. Click **File** > **Save** to save the mapping output.

    You must save the mapping output before you can create a mapping output expression in the Expression transformation.

The following image shows a mapping output that contains the sum of a decimal field and a mapping output that contains a maximum date value:



Name	Type	Precisi...	Scale	Aggregation Type	Binding	Description
Total_OrderAmt	decimal	10	2	SUM		Add total order amounts
Last_Order_Date	date/time	29	9	MAX		Latest date processed

Configuring the Mapping Output Expression

In the Expression transformation, configure the expression to aggregate for each row that the mapping processes.

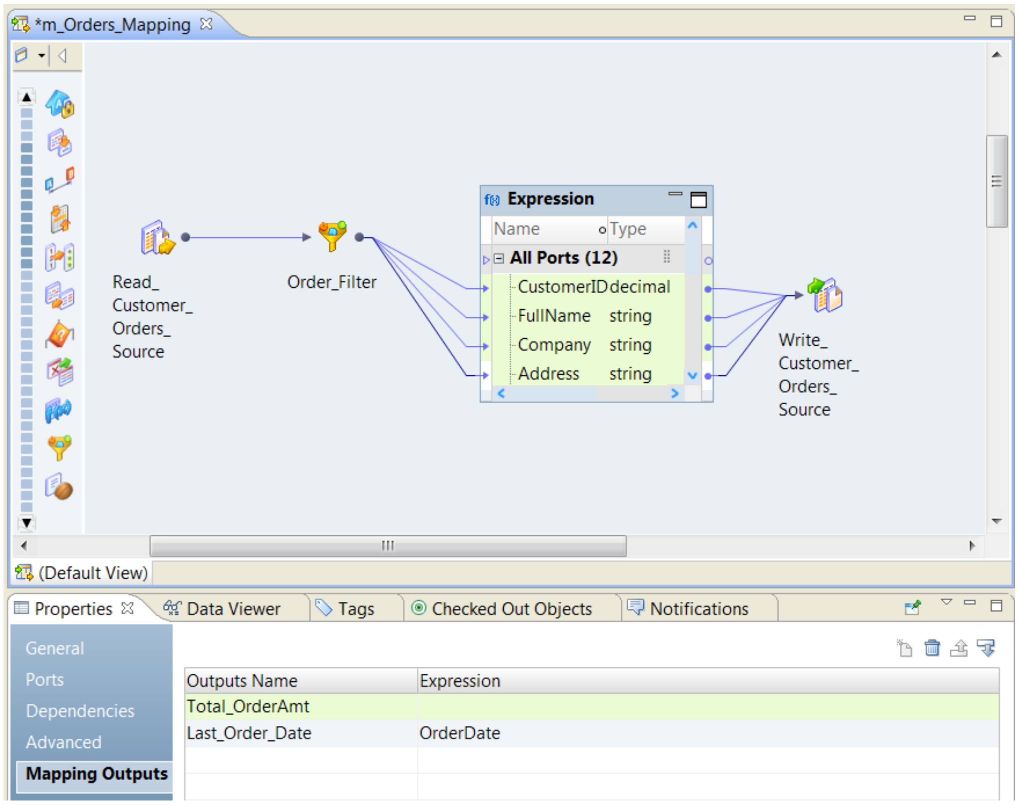
1. Add an Expression transformation to the mapping.

   Consider the mapping logic before you decide where to place the transformation. The mapping output contains an aggregation of the rows that the Expression transformation receives.

2. In the Expression transformation, click the **Mapping Outputs** view.
3. Click **New** to add a mapping output expression.

   The Developer tool creates a mapping output with a output name that matches one of the mapping outputs you created at the mapping level. If you have more than one mapping output in the mapping **Properties**, select the appropriate mapping output name to use.

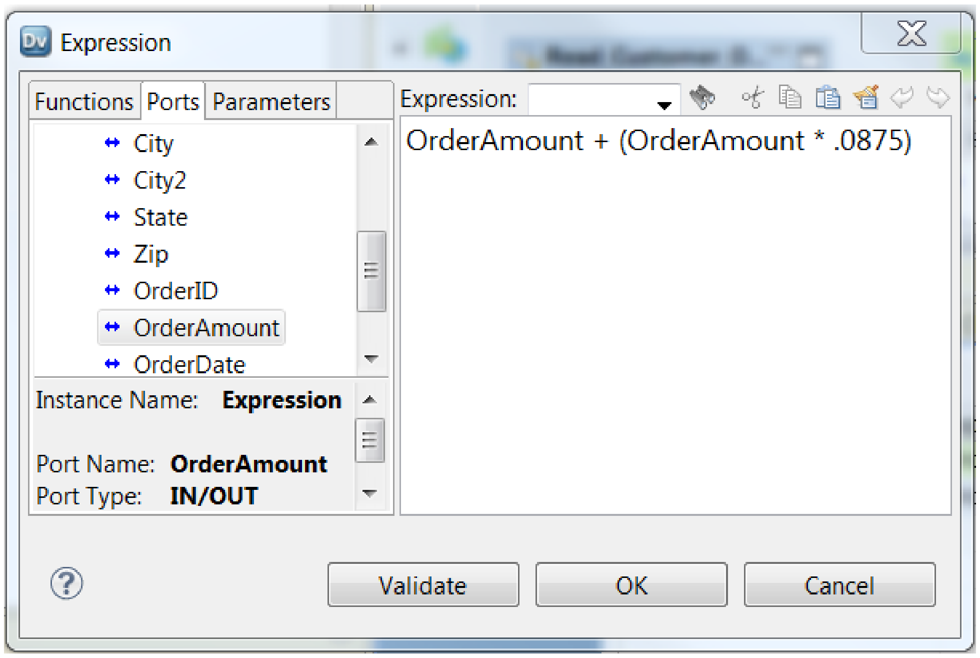
   The following image shows the **Mapping Outputs** view in the Expression transformation:



Outputs Name	Expression
Total_OrderAmt	
Last_Order_Date	OrderDate

4. Click the **Expression** column to enter an expression in the Expression Editor.

   The expression can contain just a port name or it can contain functions, ports, and parameters. The following image shows an expression to calculate the Total_OrderAmt in the Expression Editor:



```
OrderAmount + (OrderAmount * .0875)
```

Instance Name: **Expression**

Port Name: **OrderAmount**

Port Type: **IN/OUT**

5. Click **Validate** to verify that the expression is valid.
6. Click **OK** to save the expression.

The expression appears in the **Expression** column for the mapping output.

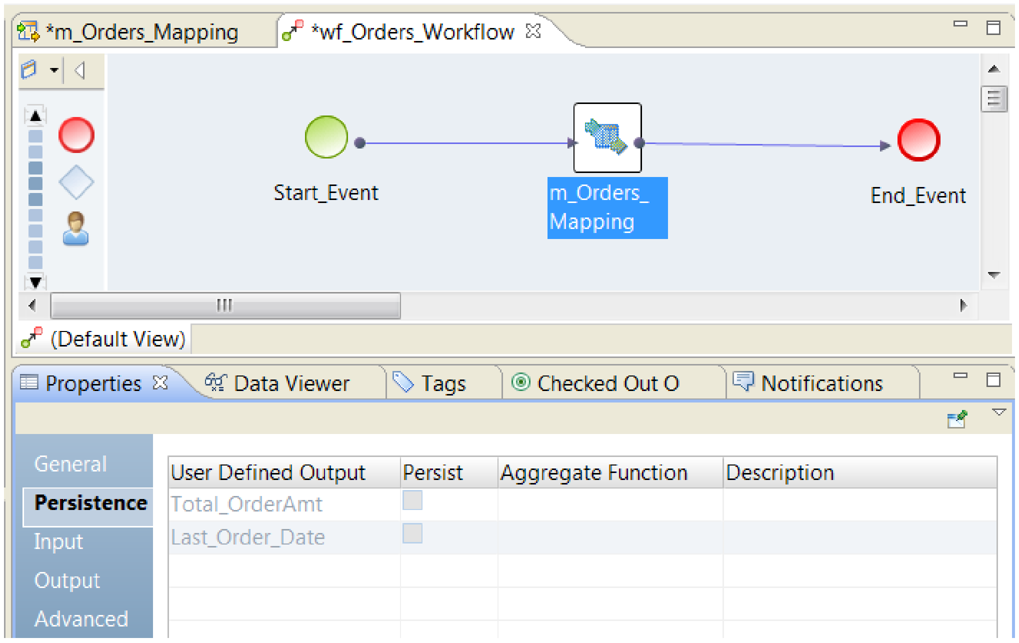
7. Click **File** > **Save** to save the Expression transformation.

Persisting Mapping Outputs

After you add the mapping to a workflow, you can persist mapping outputs from the Mapping task. You can use persisted mapping outputs as input to the Mapping task the next time it runs.

1. Add the mapping to a workflow to create a Mapping task.
2. Click the Mapping task icon in the workflow to view the Mapping task **Properties**.
3. Click the **Persistence** view.

   A list of the user-defined mapping outputs appears in the **Persistence** view.

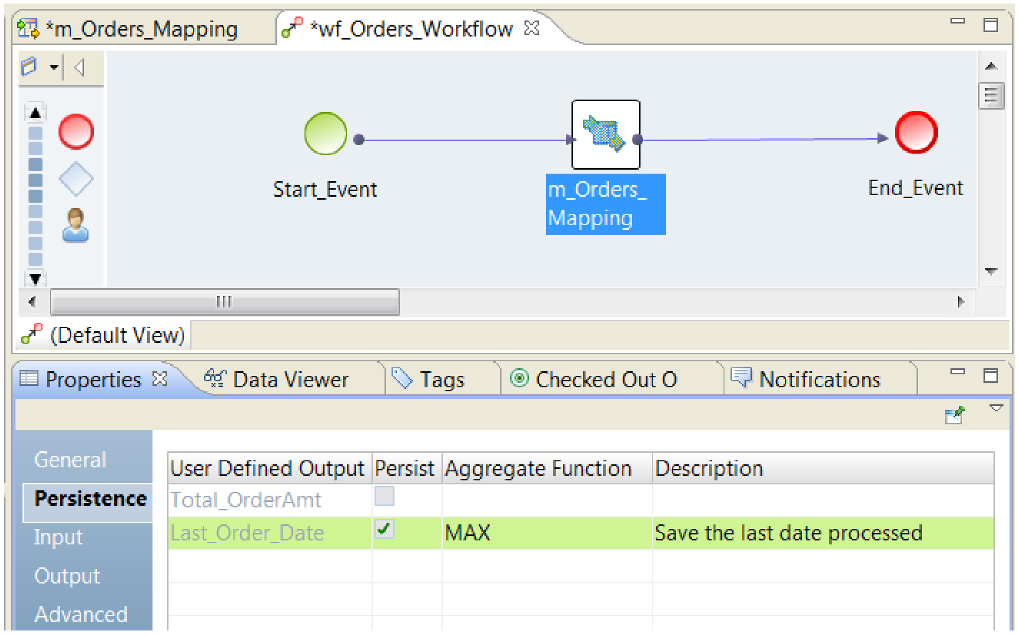


User Defined Output	Persist	Aggregate Function	Description
Total_OrderAmt	☐		
Last_Order_Date	☐		

4. Enable **Persist** to save the mapping output after the Mapping task runs.

5. Optionally, change the aggregation type and enter the description.

   The following image shows the Persistence view for a Mapping task:



User Defined Output	Persist	Aggregate Function	Description
Total_OrderAmt	☐		
Last_Order_Date	✓	MAX	Save the last date processed

   The Last_Order_Date mapping output is persisted. The aggregate function is MAX, so the Data Integration Service saves maximum order date value in the repository.

Assigning Persisted Outputs to Mapping Task Input

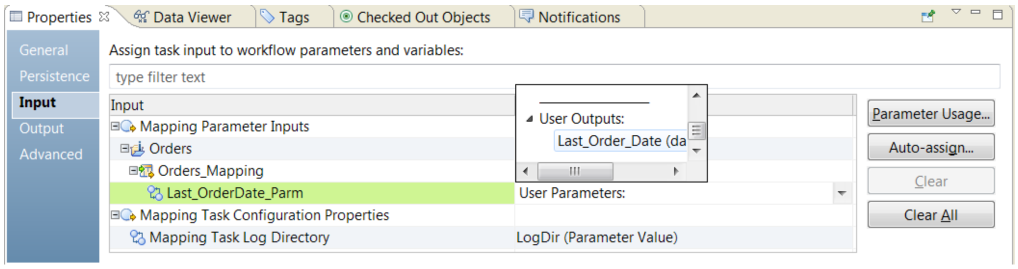
You can bind persisted mapping outputs from a Mapping task to the input parameters of the same Mapping task for the next time the workflow runs.

Assign the persisted latest order date value from the Mapping task as the input parameter to the same Mapping task. Configure a Filter transformation that uses a Last_OrderDate_Parm parameter to select which orders to process. The filter expression to select input rows is `Order_Date > Last_OrderDate_Parm`.

1. Click the Mapping task icon in the workflow to view the Mapping task **Properties** view.

   A list of the Mapping task input parameters and a list of the parameterized Mapping task configuration properties appears. The mapping must have a mapping parameter to assign the mapping output to.
2. Locate the mapping input parameter that you want to bind the mapping output to. Double-click the **Value** column to view the selection arrow.
3. Click the selection arrow to view a list of the parameters and variables that you can assign to the input parameter.
4. Scroll to the **User Outputs** section of the list and choose the persisted mapping output to use.

   The following image shows Last_OrderDate_Parm mapping parameter on the Mapping task **Input** view:



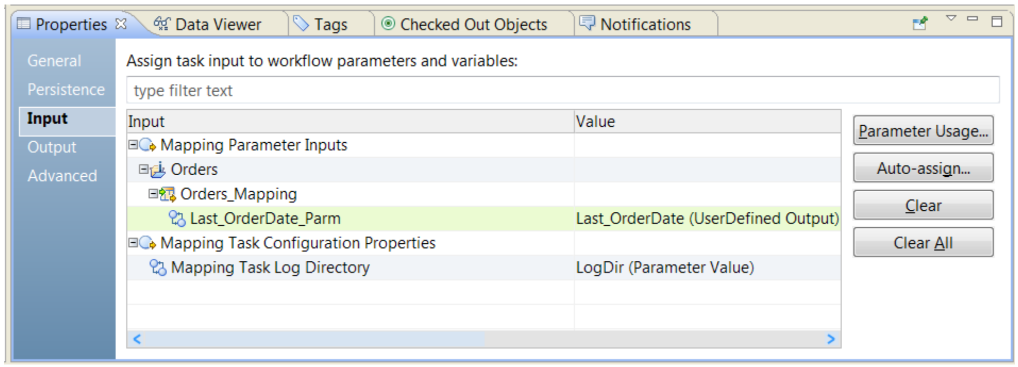
Assign task input to workflow parameters and variables:

Input	
Mapping Parameter Inputs	
Orders	
Orders_Mapping	
Last_OrderDate_Parm	User Parameters:
Mapping Task Configuration Properties	
Mapping Task Log Directory	LogDir (Parameter Value)

User Outputs:
Last_Order_Date (da

5. Select the mapping output to assign to the parameter.

   The mapping output name appears in the value column for the input parameter.



Assign task input to workflow parameters and variables:

type filter text

Input	Value
Mapping Parameter Inputs	
Orders	
Orders_Mapping	
Last_OrderDate_Parm	Last_OrderDate (UserDefined Output)
Mapping Task Configuration Properties	
Mapping Task Log Directory	LogDir (Parameter Value)

6. Click **File** > **Save** to save the Mapping task.

   The Last_OrderDate_Parm is bound to the persisted order date value from the repository.

Binding Mapping Outputs to Workflow Variables

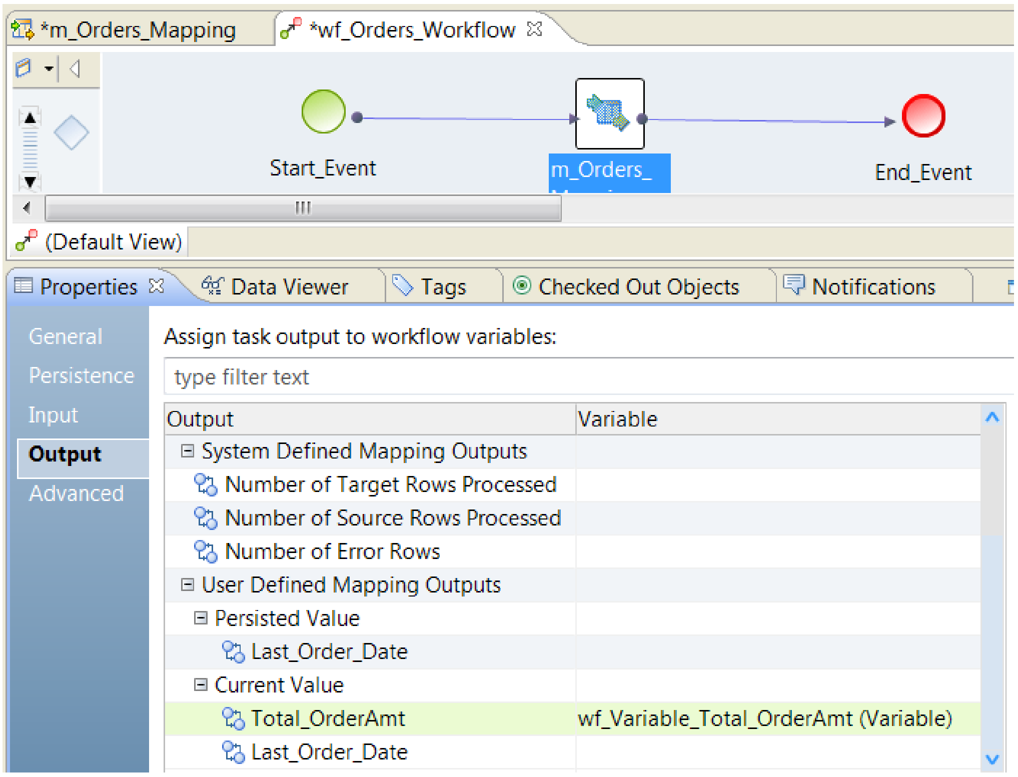
You can bind mapping outputs to workflow variables and pass the values to other tasks in the workflow.

To pass the mapping output value to another task, bind the mapping output to a workflow variable in the Mapping task **Output** view. You can bind mapping outputs from the current Mapping task or you can bind the persisted mapping outputs from the previous Mapping task run.

1. Add the mapping with the mapping outputs to a workflow.
2. Click the Mapping task icon in the workflow to view the Mapping task **Properties**.
3. In the Mapping task **Properties**, click the **Output** view.

   The Mapping task **Output** view shows the data that you can pass from the task into workflow variables.
4. Find the mapping output that you want to bind to a variable.
5. Double-click the **Variable** column to access the selection arrow and view a list of workflow variables.

   The following image shows where to bind the Total_Order_Amt mapping output to the wf_Variable_Total_OrderAmt workflow variable in the Mapping task **Output** view:



Assign task output to workflow variables:

type filter text

Output	Variable
System Defined Mapping Outputs	
Number of Target Rows Processed	
Number of Source Rows Processed	
Number of Error Rows	
User Defined Mapping Outputs	
Persisted Value	
Last_Order_Date	
Current Value	
Total_OrderAmt	wf_Variable_Total_OrderAmt (Variable)
Last_Order_Date	

6. To create a workflow variable, click the **New Variable** option from the list of workflow variables in the **Value** column.

   Enter the variable name, type, and default value.

Mapping Task Logs

You can view Mapping task logs to troubleshoot Mapping task problems or to see information about the mapping run.

The Data Integration service writes a new log file for each Mapping task run. The log file contains information about the events in the Mapping task. Log events are lines of text that contain a timestamp, thread identifier, a severity code, and the log message. The message can contain general information or it can contain an error message.

The following text shows the Mapping task log message format:

```
2015-02-20 12:49:24 <DTMLoggerThread_2> INFO: READER_1_1_1,    DBG_21430,    Reading
data from input source file [C:\Source\Logging_Source_1.txt]
2015-02-20 12:49:24 <DTMLoggerThread_2> INFO: READER_1_1_1,    BLKR_16019,    Read [200]
rows, read [0] error rows for source table [read_src2] instance name [read_src2]
2015-02-20 12:49:24 <DTMLoggerThread_2> INFO: LKPDP_2,    TE_7212,    Increasing [Data
Cache] size for transformation [Rel_Lookup] from [59652322] to [59654144].
2015-02-20 12:49:24 <DTMLoggerThread_2> INFO: READER_1_1_1,    BLKR_16008,    Reader run
completed.
2015-02-20 12:49:24 <DTMLoggerThread_2> INFO: WRITER_1_*_1,    WRT_8167,    Start
loading table [Router_Target_Default] at: Fri Feb 20 12:49:24 2015
```

When you set the tracing level to verboseData, the Mapping task log shows the parameters and parameter values for the mapping run.

The following text shows some Mapping task log messages that contain parameter values:

```
Integration Service will use override value [C:\Source] for parameter [ff_SrcDir] in
transformation [map_AllTx\read_src1].
Integration Service will use override value [8] for parameter [exp_Int] in
transformation [map_AllTx\Expression].
Integration Service will use override value [Mapping_New] for parameter [exp_String] in
transformation [map_AllTx\Expression].
Integration Service will use override value [C:\Source] for parameter [ldo_SrcDir] in
mapping \ mapplet [map_AllTx\DO_Lookup\DO_FF_REL_SRC_Read_Mapping].
```

After you run a mapping on the Spark engine on a Hadoop cluster, you can view the total number of cluster nodes used to execute the mapping in the mapping task log. On the Blaze engine, you can view the number of healthy cluster nodes used by the Grid Manager in the mapping task log.

Mapping Task Log File Name

The default name of each Mapping task log file is based on whether you are archiving the log files by timestamp or you are archiving the log files by run number.

The default name of the Mapping talk log file consists of the Mapping task name, an UID number, a timestamp, and the .log suffix. If you choose to archive the logs by run number, the default log file name consists of the Mapping task name, the run number, and the .log suffix.

In the Mapping task **Advanced** properties, configure the **Mapping Task Log Save Type** to indicate whether you want to archive files by timestamp or by run number. If you choose to archive by run number, you can enter the number of runs to keep before the Data Integration Service overwrites a log. If you choose to archive by timestamp, the Data Integration Service keeps an unlimited number of log files.

You can enter a different Mapping task log file name if you do not want to use the default name.

Mapping Task Log File Directory

You can configure the directory where the Data Integration Service writes the Mapping task log. You can parameterize the mapping task log file directory.

By default, the Data Integration Service writes the Mapping task log file in the directory defined by the system parameter, LogDir. An administrator can change the LogDir path in the Administrator tool.

You can configure a different directory for the Mapping task log file in the Mapping task **Advanced** properties. If you enter a directory name that the Data Integration Service cannot access, the Mapping task fails.

When you run a workflow on a grid, the mapping task log directory is created on the machine where the Data Integration Service master node is running. Configure the mapping task log file directory on a shared directory that all nodes on the grid can access.

Mapping Task Advanced Properties

The **Advanced** tab for a Mapping task includes properties that the task uses to run the mapping.

You can select values for the configuration properties. Or, you can assign a configuration property to task input. When you choose to assign the property to task input, you can assign a workflow parameter or a variable to the property. Assign a workflow or a variable to the property in the Mapping task **Input** tab. You can use a parameter set or a parameter file to change the value of the property.

You must select a value for the task recovery strategy. You cannot assign the task recovery strategy to task input.

Configure the following advanced properties for a Mapping task:

Task Recovery Strategy

Determines whether the Data Integration Service reruns or skips a task that is interrupted or that encounters an error. If the workflow is not enabled for recovery, the Data Integration Service ignores the task recovery strategy.

By default, the task has a restart recovery strategy.

When you use the Create Cluster task to create a cluster workflow, set the Mapping task recovery strategy to **Restart task**.

Cluster Identifier

Cluster Identifier string identifying the cluster where the Mapping task will run.

Set the Cluster Identifier property only for Mapping tasks that are contained in cluster workflows.

Choose one of the following options:

Value	Description
Blank (no value)	Run the mapping runs on the cluster configured in the Hadoop connection associated with the mapping.
AutoDeploy	Run the mapping on the cluster that the workflow creates. When you choose this option, it also populates the Cluster Identifier property in the Create Cluster task with the value Set to AutoDeployCluster. Default is AutoDeploy.
(Assign to task input)	Select this option to accept input from another source than the Create Cluster task. If you choose this option, enter a parameter value in the Cluster Identifier property of the Mapping task Input properties tab.

Default Date Time Format

The date time format that the Data Integration Services uses to convert data between date and string data types. Select one of the predefined formats, or type a valid date format string.

The property has a string datatype. Default is MM/DD/YYYY HH24:MI:SS.

Optimizer Level

Controls the optimization methods that the Data Integration Service applies to a mapping.

- 0 (None). The Data Integration Service does not optimize the mapping.
- 1 (Minimal). The Data Integration Service applies the early projection optimization method to the mapping.
- 2 (Normal). The Data Integration Service applies the early projection, early selection, pushdown, and predicate optimization methods to the mapping.
- 3 (Full). The Data Integration Service applies the early projection, early selection, pushdown, predicate, cost-based, and semi-join optimization methods to the mapping.

The property has an integer data type. Default is 2 (Normal).

High Precision

Runs the mapping with high precision.

High precision data values have greater accuracy. Enable high precision if the mapping produces large numeric values, for example, values with precision of more than 15 digits, and you require accurate values. Enabling high precision prevents precision loss in large numeric values.

The property has a boolean data type. Default is true.

Sort Order

The order in which the Data Integration Service sorts character data in the mapping.

The property has a string data type. Default is Binary.

Override Tracing Level

Sets the override tracing level used for each transformation in the mapping. The tracing level determines the amount of information that the Data Integration Service sends to the mapping log files.

Choose one of the following tracing levels:

- none. The Data Integration Service does not override the tracing level that you set for each transformation.
- terse. The Data Integration Service logs initialization information, error messages, and rejected data notifications.
- normal. The Data Integration Service logs initialization information, status information, errors encountered, and skipped rows from transformation row errors. It summarizes mapping results, but not at the level of individual rows.
- verboseInitialization. In addition to normal tracing, the Data Integration Service logs initialization details, the names of the index files and the data files, and detailed transformation statistics.
- verboseData. In addition to verbose initialization tracing, the Data Integration Service logs each row that passes into the mapping. The Data Integration Service includes the names and the values of the parameters in the mapping run.The Data Integration Service also notes where it truncates string data to fit the precision of a column and it returns detailed transformation statistics. The Data Integration Service writes row data for all rows in a block when it processes a transformation.

The property has a string datatype. Default is normal.

Mapping Task Log Directory

Directory for the mapping task log. Use a directory local to the Data Integration Service to save the task log. If you do not enter a path, the Data Integration Service writes the log file to the directory from the LogDir system parameter. The default directory is `<Informatica installation directory>/tomcat/bin/disLogs/builtinhandlers`. An administrator can change this path in the Administrator tool. The system parameter is in the Data Integration Service service process properties.

Mapping Task Log File Name

Name of the mapping task log. You can enter a file name or the file name with the directory name in this field. The Data Integration Service appends the file name to the information in the **Masking Task Log File Directory** field. It appends the log file name to a UID and time stamp or to a mapping run number, based on how you choose to save the log file.

Java Classpath

The Java classpath to add to the beginning of the system classpath when the Data Integration Service runs the mapping task. Use this option if you use third-party Java packages, built-in Java packages, or custom Java packages in a Java transformation.

Mapping Task Log Save Type

Saves the Mapping task log file by timestamp or by the number of mapping task runs. The suffix of the mapping task log file name reflects the option you select.

When you configure a Mapping task log file, you can restrict the number of log files to keep at one time or you can choose to create an unlimited number of log files. Select one of the following save types:

- Mapping task timestamp. Default. Saves log files by time stamp. The Integration Service saves an unlimited number of logs and labels them by time stamp. Each Mapping task log file name has the following format: `<MappingTaskLogFileName>_<UID>_<Timestamp>.log`
- Mapping task runs. Saves a specific number of Mapping task logs. Configure the number of log files to save in the **Save Mapping Task Logs for These Runs** property. Each Mapping task log file name has the following format:`<MappingTaskLogFileName>_<Run#>.log`

Save Mapping Task Logs For These Runs

The number of log files to save when you save the log file by the number of mapping runs. Each log file name contains the run number. If you choose to save 10 Mapping task logs, the Data Integration Service saves logs numbered 0 to 9.

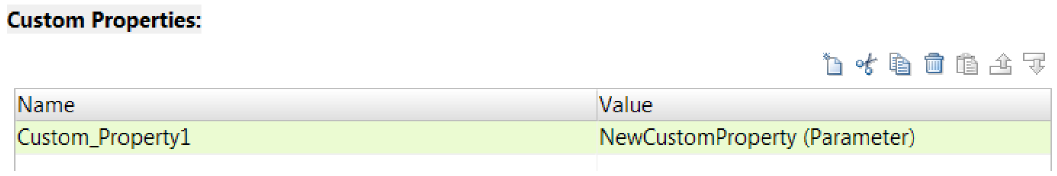
Custom Properties

You can define custom properties for a Mapping task and configure the property values.

You might need to apply custom properties in special cases. Define custom properties only at the request of Informatica Global Customer Support.

To define a custom property, click **New** and enter the property name with an initial value. You can enter a workflow parameter or a workflow variable for the custom property value. You can also create a new workflow parameter or variable.

The following image shows the Custom Properties panel:



Custom Properties:

Name	Value
Custom_Property1	NewCustomProperty (Parameter)

Variable and Parameter Values for Configuration Properties

If you assign a Mapping task configuration property to a workflow variable or parameter in the **Input** tab, you must define a valid variable or parameter value for the property.

The following table lists the valid variable and parameter values that you can define for Mapping task configuration properties:

Property	Workflow Variable Value	Workflow Parameter Value
Default date time format	String variable with a valid date format	String parameter with a valid date format
Optimizer level	- Integer variable with a value of 0 for None, 1 for Minimal, 2 for Normal, or 3 for Full. - String variable with a value of "0" for None, "1" for Minimal, "2" for Normal, or "3" for Full. - Boolean variable. The Data Integration Service converts a value of true to 1 (Minimal). The service converts a value of false to 0 (None).	String parameter with a value of "0" for None, "1" for Minimal, "2" for Normal, or "3" for Full.
High precision	- Boolean variable. - String variable with a value of "true" or "false." - Integer variable. The Data Integration Service converts a value of zero as false. The service converts any non-zero value as true.	String parameter with a value of "true" or "false."

Property	Workflow Variable Value	Workflow Parameter Value
Sort order	String variable with one of the following case sensitive values: - albanian - arabic - armenian - belarusian - bengali - binary - bulgarian - catalan - croatian - czech - danish - dutch - english - estonian - finnish - french - german - germanPhonebook - greek - hebrew - hindi - hungarian - indonesian - italian - japanese - kazakh - korean - latvian - lithuanian - macedonian - norwegian - pashto - persian - polish - portuguese - romanian - russian - serbian - simplifiedChinese - slovak - slovenian - spanish - swedish - thai - traditionalChinese - traditionalSpanish - turkish - ukranian - vietnamese	String parameter with one of the same case sensitive values that are valid for variables.
Override tracing level	String variable with one of the following case-sensitive values: - terse - normal - verboseInitialization - verboseData	String parameter with one of the same case-sensitive values that are valid for variables.

Property	Workflow Variable Value	Workflow Parameter Value
Mapping Task Log Directory	String variable that contains a valid path that the Data Integration Service can access. Default is LogDir.	String parameter that contains a valid path that the Data Integration Service can access. Default is LogDir.
Mapping Task Log File Name	String variable that contains a valid file name.	String parameter that contains a valid file name.
Java Classpath	String variable that contains a valid path to a third party or custom Java package.	String parameter that contains a valid path to a third party or custom Java package.

Related Topics:

- "Workflow Variable Datatype Conversion" on page 35
- "Workflow Parameter data type Conversion" on page 43

Changing the Mapping that the Task Runs

After you configure a Mapping task, you can select a different mapping for the task to run.

If you select a different mapping, the Developer tool removes the following information configured for the Mapping task:

- User-defined mapping parameters assigned to workflow parameters or variables in the **Input** tab.
- Transformation output data assigned to workflow variables in the **Output** tab.

1. Select the Mapping task in the editor.
2. In the **Properties** view, click the **General** tab.
3. Click **Browse**, select a different mapping, and click **OK**.

   The **Confirm Change** dialog box appears.
4. Click **Yes** to confirm that you want to change the mapping.

   Configure the Mapping task input and output for the selected mapping.

Mapping Tasks in a Cluster Workflow

A cluster workflow is a workflow that automates the creation of a cluster using a Create Cluster task, and then runs one or more mappings on that cluster. A cluster workflow must contain one or more Mapping tasks.

Before you add a mapping to a Mapping task, set the Run-time *Connection* property in the mapping to designate where to run the mapping. You can choose to run the mapping on the cluster that the workflow creates or on another cluster.

For more information about cluster workflows, see the *Data Engineering Integration User Guide*.

CHAPTER 8

Notification Task

This chapter includes the following topics:

Notification Task Overview

A Notification task sends a notification to specified recipients during the workflow.

You can send an email notification from a Notification task. For example, you want to track how long a mapping takes to run from a workflow. You include a Notification task in the workflow to send an email containing the time and date that the Mapping task starts and completes.

Before you configure Notification tasks to send emails, an administrator must enable and configure the Email Service in the Administrator tool.

When you add a Notification task to a workflow, you specify the recipients and configure the notification properties. Recipients include users and groups in the Informatica domain that receive the notification. For an email notification, the properties include email addresses and email content. You can use workflow parameters and variables to dynamically determine the recipients, email addresses, and email content during the workflow.

When a Notification task finishes running, the task passes output data back to the workflow in variables.

When you configure a Notification task, you specify the recipients, the email addresses, the email content, and the output data that the task passes to the workflow. You also set the advanced properties that the task uses when it runs.

Recipients

Recipients include users and groups in the Informatica domain that receive the notification during the workflow. Select the recipients in the **Notification** tab.

You can select users and groups from native and LDAP security domains. The Developer tool displays selected users and groups using the following format:

```
<security domain name>\<user login name or group name>
```

When you configure a user to receive an email notification, the Notification task sends an email to the user at the email address that the user account properties specify. When you configure a group to receive an email notification, the Notification task sends an email to all users in the group at the email addresses that the user account properties specify.

If a user account email address is not valid, the Notification task runs and reports back to the domain that the email address is undeliverable. The Notification task sends an email to the address that the Email Service specifies to indicate the delivery failure.

You can use the Administrator tool to enter an email address for native user accounts. You cannot use the Administrator tool to enter an email address for LDAP user accounts. If the user account information imported from the LDAP directory service includes an email address, the Notification task can send an email to the user at that email address.

You can select a recipient multiple times by selecting a user and then selecting a group that the user belongs to. When the Notification task sends the email, the duplicated recipient receives multiple emails or a single email based on how the email server handles duplicate recipients.

When you configure the email properties for a Notification task, you can enter user and group names in the address fields and configure dynamic recipients. Dynamic recipients are users and groups that you define in workflow parameters or variables.

Selecting Recipients

Add users and groups to the recipient list when you want to send a notification to the users and groups.

You can add users and groups to the recipient list from the Notification task recipient list or from the email properties. Any users or groups that you add in one location also appear in the other location. You can type user and group names and configure dynamic recipients from the email properties only.

When you add users or groups, you can use a search filter. Enter a string to search for users or groups. You can use wildcard characters in the string. The Developer tool returns all names that contain the search string. The string is not case sensitive.

1. Select the Notification task in the editor.
2. In the **Properties** view, click the **Notification** tab.
3. Select **Recipients**.
4. Click **Choose**.

   The **Select Users and Groups** dialog box appears.
5. Enter the filter conditions to search for users or groups.

   To view the users within a group, select the group and click **View Users in Group**.
6. Select a user or group name.

   Use the Ctrl or Shift key to select multiple names.
7. Click **OK**.

Configuring Notification Types for Recipients

To configure a recipient to receive an email notification, select the email address fields for the recipient.

1. Select the Notification task in the editor.
2. In the **Properties** view, click the **Notification** tab.
3. Select **Recipients**.
4. Select a user, group, or dynamic recipient in the recipients list.
5. In the **Email** column, select To, Cc, or Bcc.

Typing Recipient Names

When you configure the email properties for a Notification task, you can type user and group names in the address fields.

When you type user and group names in address fields, use the following format:

```
<security domain name>\<user login name or group name>
```

For example, type `Native\adietrich` to specify the user in the native security domain with an adietrich login name.

If you do not specify a security domain name, the Developer tool uses the native security domain.

1. Select the Notification task in the editor.
2. In the **Properties** view, click the **Notification** tab.
3. Select **Email** under **Notification Types**.
4. In one of the address fields, enter a user or group name using the required format.

   Enter multiple recipient names separated by a semicolon.

Dynamic Email Recipients

Use workflow parameters and variables to dynamically define email recipients.

When you configure recipients in the email properties, use the **Dynamic Recipients** tab to select workflow parameters and variables that define user and group recipients. You cannot type a parameter or variable name in the address fields.

The following table describes the types of parameters and variables that you can use to define dynamic recipients:

Parameter or Variable Type	Description
Workflow parameters	Use a string workflow parameter to define the user or group name in a parameter file when you run the workflow.
System workflow variables	Use the UserName system workflow variable to send an email to the user that runs the workflow.
User-defined workflow variables	Use a user-defined string workflow variable to dynamically assign a user or group name using an Assignment task in the workflow.

When you define a user or group name in a parameter file or in an Assignment task, enter a single user or group name for the value. Use the following syntax for the value:

```
<security domain name>\<user login name or group name>
```

For example, enter `Native\Developers` to specify the Developers group in the native security domain.

If you do not specify a security domain name, the Data Integration Service uses the native security domain. The Data Integration Service first attempts to find the specified name in the list of groups. If the name is not a group, the service then attempts to find the name in the list of users.

Configuring Dynamic Email Recipients

Use workflow parameters and variables to configure dynamic email recipients.

1. Select the Notification task in the editor.
2. In the **Properties** view, click the **Notification** tab.
3. Select **Email** under **Notification Types**.
4. Click **To**.

   The **Email Properties** dialog box appears.
5. Select **Recipients**, and then click **Dynamic Recipients**.
6. Click **New**.

   The Developer tool adds an empty row to the recipients list.
7. Click in the **Recipients** column, and then select an existing parameter or variable, or a new parameter or variable.
   - Select a workflow parameter or variable name.
   - Click **New Parameter** or **New Variable**. In the **Add Parameter** or **Add Variable** dialog box, enter the name and type of the parameter or variable. Enter a default user or group name for a parameter. Or, enter an initial user or group name for a variable. The Developer tool creates the workflow parameter or variable and adds it to the dynamic recipient list.
8. In the **Email** column, select To, Cc, or Bcc.
9. Click **OK**.

   The Notification task recipient list displays the workflow parameter or variable under **Dynamic Recipients**.

Email Addresses

In addition to specifying users and groups as the email recipients, you can enter email addresses that receive an email from the Notification task. Enter email addresses in the **Notification** tab.

You can enter any valid email address. Enter multiple email addresses separated by a semicolon. You can use workflow parameters and variables to dynamically define email addresses.

If you enter an email address that is not valid, the Notification task runs and reports back to the domain that the email address is undeliverable. The Notification task sends an email to the address that the Email Service specifies to indicate the delivery failure.

Entering Email Addresses

A Notification task can send an email to any valid email address that you enter in the email properties.

1. Select the Notification task in the editor.
2. In the **Properties** view, click the **Notification** tab.
3. Select **Email** under **Notification Types**.
4. In the **Properties** area, enter a fully qualified email address in the appropriate address field.

   Enter multiple email addresses separated by a semicolon.

Dynamic Email Addresses

Use workflow parameters and variables to dynamically define email addresses.

When you enter email addresses in the email properties, use the **Dynamic Email Addresses** tab to select parameters and variables that define email addresses. You cannot type a parameter or variable name in the address fields.

The following table describes the types of parameters and variables that you can use to define dynamic email addresses:

Parameter or Variable Type	Description
Workflow parameters	Use a string workflow parameter to define the email address in a parameter file when you run the workflow.
System workflow variables	Use the UserName system workflow variable to send an email to the user that runs the workflow.
User-defined workflow variables	Use a user-defined string workflow variable to assign an email address using an Assignment task in the workflow.

When you define an email address in a parameter file or in an Assignment task, enter a single email address for the value.

Configuring Dynamic Email Addresses

Use workflow parameters and variables to configure dynamic email addresses.

1. Select the Notification task in the editor.
2. In the **Properties** view, click the **Notification** tab.
3. Select **Email** under **Notification Types**.
4. Click **To**.

   The **Email Properties** dialog box appears.
5. Select **Email Addresses**, and then click **Dynamic Email Addresses**.
6. Click **New**.

   The Developer tool adds an empty row to the recipients list.
7. Click in the **Recipients** column, and then select an existing parameter or variable, or a new parameter or variable.

- Select a workflow parameter or variable name.
- Click **New Parameter** or **New Variable**. In the **Add Parameter** or **Add Variable** dialog box, enter the name and type of the parameter or variable. Enter a default email address for a parameter, or an initial email address for a variable.

8. In the **Email** column, select To, Cc, or Bcc.
9. Click **OK**.

Email Content

Email content includes the email subject and body. Enter email content in the **Notification** tab. You can use workflow parameters and variables to dynamically define email content.

Dynamic Email Content

Use workflow parameters and variables in the email subject and body to dynamically define the content.

You can select a workflow parameter or variable name when you select **Email Content** in the email properties.

You can type a workflow parameter or variable name in the subject or body fields when you select **Email** notification type in the **Notification** tab. When you type the parameter or variable name in the subject or body, use the required syntax.

For example, you create a workflow variable named MappingErrorRows and assign the number of error rows output value for a Mapping task to the variable. You enter the following text in the body of a Notification task:

```
Mapping failed to write ${var:MappingErrorRows} rows to the target.
```

The following table describes the types of parameters and variables that you can use to define dynamic content:

Parameter or Variable Type	Description
Workflow parameters	Use a workflow parameter of any type to define email content in a parameter file when you run the workflow.
System workflow variables	Use any of the system workflow variables to include the values of the system workflow variables in email content.
User-defined workflow variables	Use a workflow variable of any datatype to include output values produced by other tasks in email content. Or, use a workflow variable of any datatype to include values assigned to the variable with an Assignment task in email content.

Related Topics:

- "Parameter Names in Expressions and Strings" on page 40
- "Variable Names in Expressions and Strings" on page 33

Entering Email Content

When you configure email content, you enter the email subject and body text. You can use workflow parameters and variables to configure dynamic email content.

1. Select the Notification task in the editor.
2. In the **Properties** view, click the **Notification** tab.
3. Select **Email** under **Notification Types**.
4. Click **Subject**.

   The **Email Properties** dialog box appears.
5. In the **Email Content** view, enter text in the subject and body fields.
6. To configure dynamic content using an existing workflow parameter or variable, select the subject or body field, and then double-click an existing workflow parameter or variable.

   The Developer tool adds the parameter or variable to the field using the required syntax.
7. To configure dynamic content using a new workflow parameter or variable, click **New Parameter** or **New Variable**.

   In the **Add Parameter** or **Add Variable** dialog box, enter the name and type of the parameter or variable. Enter a default value for a parameter, or an initial value for a variable. The Developer tool creates the workflow parameter or variable and adds it to the email content field.
8. Click **OK**.

Notification Task Output

Notification task output is the data that passes from a Notification task into workflow variables. Notification task outputs include general outputs.

When you configure a Notification task, you specify the task output values that you want to assign to workflow variables on the **Output** tab. The Data Integration Service copies the Notification task output values to workflow variables when the Notification task completes or fails. If the task aborts, the Data Integration Service does not copy task output values to workflow variables.

For example, a Notification task produces a start time output value that indicates when the Data Integration Service started running the task. The workflow cannot directly access this Notification task output data. To use the data in the remainder of the workflow, you assign the start time output to a workflow variable named NotificationStartTime. Then use the NotificationStartTime workflow variable in an expression for a conditional sequence flow. The Data Integration Service runs the next object in the workflow if the Notification task started before the specified time.

General outputs include output data produced by all tasks such as the task start time, end time, and whether the task successfully ran.

Related Topics:

- “Task Output” on page 27
- “Assign a Value with Task Output” on page 31

Notification Task Advanced Properties

The **Advanced** tab for a Notification task includes the task recovery strategy.

Configure the following Notification task advanced property:

Task Recovery Strategy

Determines whether the Data Integration Service reruns or skips a task that is interrupted or that encounters an error. If the workflow is not enabled for recovery, the Data Integration Service ignores the task recovery strategy.

By default, the task has a restart recovery strategy.

Troubleshooting Notification Tasks

The solution to the following situation might help you troubleshoot Notification tasks.

A Notification task fails with the message "Exception Reading Response."

This message originates from the SMTP mail server and might indicate that the email server properties for the Email Service are incorrectly configured in the Administrator tool. For example, the email server properties might specify that the SMTP server uses SSL security. However, the specified SMTP server port number is the TLS port number instead of the SSL port number.

CHAPTER 9

Gateways

This chapter includes the following topics:

Gateways Overview

A gateway splits a sequence flow into multiple sequence flows or it merges multiple sequence flows into a single sequence flow. The Data Integration Service evaluates the sequence flows at run time and runs the objects on the sequence flows that meet the conditions that you specify.

Add gateways to a workflow in pairs. The first gateway splits the sequence flow from an upstream workflow object and connects to multiple objects that you select. The second gateway merges the sequence flows so that you can connect a single sequence flow to the next object in the workflow. The first gateway is the outgoing gateway. The second gateway is the incoming gateway. A gateway must connect to at least two sequence flows.

Each sequence flow represents a branch that the workflow data can follow. You can connect multiple objects consecutively on a branch. Connect the final sequence flow on each branch to the incoming gateway.

The branches that you create on an outgoing gateway must merge into an incoming gateway of the same type. You can connect multiple workflow objects on each branch before the branches merge into the incoming gateway.

You can add Exclusive gateways and Inclusive gateways to a workflow. Add Exclusive gateways to run the objects on a single branch between the gateways. Add Inclusive gateways to run the objects on multiple branches in parallel. For example, you might use Exclusive gateways to determine that a workflow follows one branch if a mapping runs successfully and follows another branch if the mapping fails. You might use Inclusive gateways run a series of mappings in parallel from a single application.

You can add Exclusive gateways and Inclusive gateways to a branch between two gateways. When you add gateways to a branch between two gateways, add the gateways in pairs. For example, add an outgoing Inclusive gateway and an incoming Inclusive gateway to a branch between two gateways. Or, add an outgoing Exclusive gateway and an incoming Exclusive gateway to the branch. The gateways that you add to the branch are called nested gateways.

The branches that you create on an outgoing gateway must merge into an incoming gateway of the same type. For example, the branches from an outgoing Exclusive gateway must merge into an incoming Exclusive gateway. You can connect multiple workflow objects on each branch before the branches merge into the incoming gateway.

You define conditions on the sequence flows that you create on an outgoing gateway. When the Data Integration Service evaluates the conditions on an outgoing Exclusive gateway, the Data Integration Service runs the first sequence flow with a condition that evaluates to true. When the Data Integration Service evaluates the conditions on an outgoing Inclusive gateway, the Data Integration Service runs every sequence flow with a condition that evaluates to true. The Data Integration Service runs the sequence flows on the Inclusive gateway concurrently.

You must select a default sequence flow on each outgoing gateway. The default sequence flow ensures that data can flow from the outgoing gateway to the incoming gateway if all of the sequence flow conditions evaluate to false. The Data Integration Service runs the default sequence flow on an Exclusive gateway if the conditions on the other sequence flows evaluate to false. The Data Integration Service always runs the default sequence flow on an Inclusive gateway. You do not need to define a condition on the default sequence flow.

Exclusive Gateways

Use Exclusive gateways to create multiple branches from a sequence flow and to run the objects on a single branch. The Data Integration Service runs the objects on the first branch with a sequence flow condition that evaluates to true.

The Data Integration Service evaluates the conditions on each sequence flow in the order that you specify on the outgoing gateway properties. If a condition evaluates to true, the Data Integration Service follows the branch that the sequence flow indicates and does not evaluate any other branch. If a condition evaluates to false, the Data Integration Service skips the branch and evaluates the condition on the next sequence flow. When the objects on the branch are complete, the Data Integration Service passes the data to an incoming Exclusive gateway.

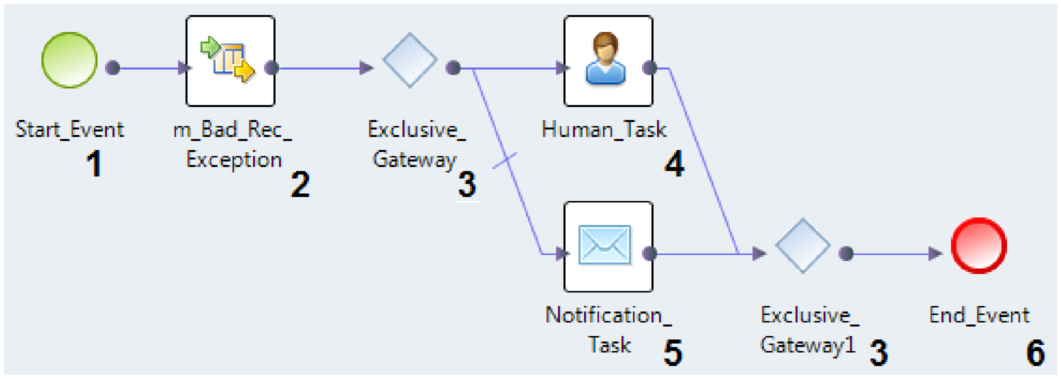
You specify a default sequence flow in the gateway properties. The Data Integration Service evaluates the default sequence flow last, regardless of the order that you set for the sequence flows in the gateway properties. If no other branch on the outgoing Exclusive gateway can run, the Data Integration Service runs the branch that the default sequence flow indicates.

Exclusive Gateway Example

Add Exclusive gateways to a workflow to create branches that run exclusively of each other. When you run a workflow that contains Exclusive gateways, the Data Integration Service runs the tasks on a single branch between the gateways. Configure conditions on the outgoing sequence flows to determine the branch that the workflow follows at run time.

You can use Exclusive gateways when the path that the workflow must follow depends on the result from an upstream workflow object. For example, you might configure a workflow with a Mapping task that identifies exception records and a Human task that distributes the exception records to data stewards. If the mapping in the Mapping task does not identify exceptions, the Human task cannot distribute records. You use an exclusive gateway to create two branches for the workflow. One branch contains the Human task that can distribute the exception records. Another branch contains a Notification task that sends an email to the data stewards if the mapping source data contains no exception records.

The following image shows a workflow that contains the branches between the Exclusive gateways:



The workflow contains the following objects:

1. Start event. The Start event starts the workflow.
2. Mapping task. The Mapping task identifies a mapping that finds the exception records in a data set. The Mapping task output includes the *exceptionLoadCount* variable value. The Human task uses the *exceptionLoadCount* value to determine the number of exception records to distribute to the data stewards.
3. Exclusive gateways. The first gateway splits the sequence flow into two branches. One branch includes a Human task. The other branch includes a Notification task. The second gateway merges the branches into a single sequence flow.

   Note: Define a condition on the sequence flow that connects to the Human task. Configure the condition to evaluate to true if the *exceptionLoadCount* variable value that the Mapping task generates is greater than zero.
4. Human task. The Human task reads the exception records that the earlier mapping identified and distributes the records to data stewards for analysis.
5. Notification task. The Notification task sends an email to the data stewards to indicate that the Human task did not run.

   The sequence flow that connects to the Notification task is the default sequence flow. The Data Integration Service runs the Notification task if the condition on the Human task sequence flow evaluates to false.
6. End event. The End event ends the workflow.

Inclusive Gateways

Use Inclusive gateways to create multiple branches from a sequence flow and to run the objects on one or more branches in parallel. The Data Integration Service runs the objects on every branch with a sequence flow condition that evaluates to true.

The Data Integration Service evaluates the conditions on each sequence flow before it runs an object on any branch. If a condition evaluates to true, the Data Integration Service follows the branch that the sequence flow indicates. The Data Integration Service runs the objects on each branch concurrently. When the objects on all branches are complete, the Data Integration Service passes the data from the incoming Inclusive gateway to the next object in the workflow.

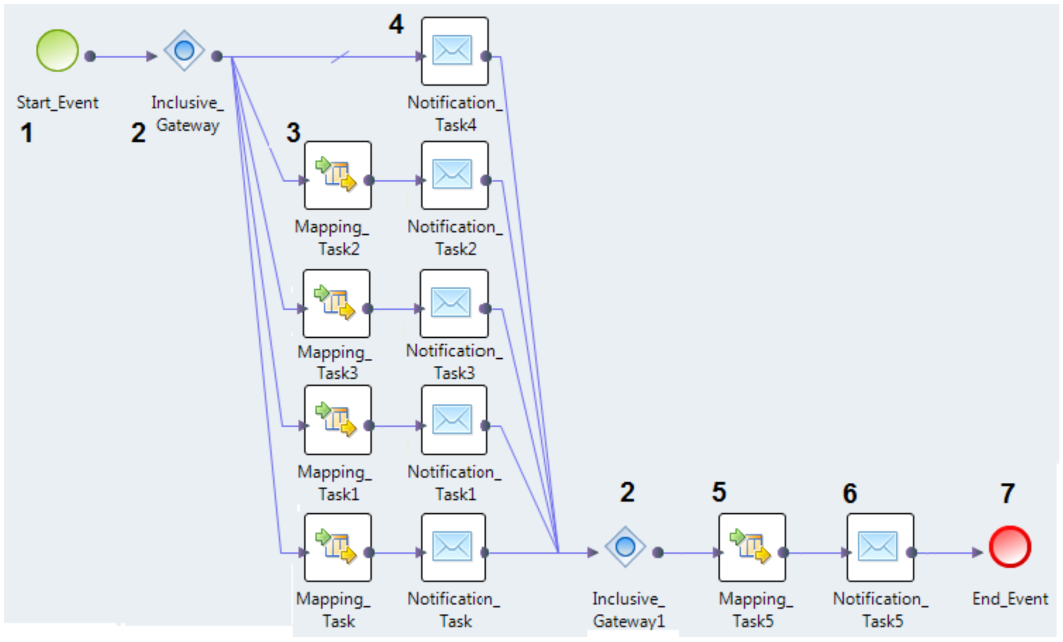
You specify a default sequence flow in the gateway properties. You do not need to define a condition on the default sequence flow. The Data Integration Service always runs the default sequence flow on an Inclusive gateway.

Inclusive Gateway Example

Add Inclusive gateways to a workflow to run multiple tasks in parallel. For example, you can configure an Inclusive gateway with multiple branches and add a Mapping task to each branch. When the workflow runs, the Data Integration Service concurrently runs the mappings that the tasks identify.

You might configure multiple Mapping tasks in parallel when you need to process very large volumes of data. The tasks might identify mappings that read data from different locations and write the data to a warehouse. Add the Mapping tasks to the branches that the Inclusive gateways define. The Data Integration Service treats each mapping as a separate job. If you configure the Data Integration Service to run on a grid, the Data Integration Service can assign the mapping jobs to different nodes on the grid. If you configure the mappings to run in a Hadoop environment, the Data Integration Service pushes the mappings to the Hadoop cluster.

The following image shows a workflow that contains multiple Mapping tasks between Inclusive gateways:



The workflow contains the following objects:

1. Start event. The Start event starts the workflow.
2. Inclusive gateways. The two gateways split the workflow into multiple branches and merge the branches into a single flow.
3. Mapping tasks. The workflow branches include a series of Mapping tasks. Each task identifies a mapping that reads a different data source. The mappings write to a common data target.
4. Notification tasks. Each branch includes a Notification task that sends an email to the data owners when the corresponding mapping runs.

   The default sequence flow does not connect to a Mapping task. The sequence flow connects to a Notification task that notifies the workflow owner that the Data Integration Service reached the outgoing gateway.
5. Mapping task. The final mapping task includes an Expression transformation that verifies the number of records that the earlier mappings wrote to the target database tables.

6. Notification task. The final Notification task sends an email to the workflow developer to indicate that the workflow tasks are complete.
7. End event. The End event ends the workflow.

Note: If a Mapping task includes a mapping that you configure to run in a Hadoop environment, do not assign the Mapping task outputs to workflow variables. Mappings that run in a Hadoop environment do not provide the total number of source, target, or error rows. When a Mapping task includes a mapping that runs in a Hadoop environment, the task outputs contain a value of zero (0).

Default Sequence Flows

When you create the sequence flows that link an outgoing gateway to other objects, you specify a default sequence flow. The default sequence flow ensures that the Data Integration Service can always identify an object to run on an outgoing sequence flow.

The Data Integration Service uses the default sequence flow in different ways for the different gateway types. On an Exclusive gateway, the Data Integration Service runs the object on the default sequence flow if the conditions on every other sequence flow evaluate to false. On an Inclusive gateway, the Data Integration Service runs the object on the default sequence flow regardless of the conditions on every other sequence flow.

You do not need define a condition on the default sequence flow. The Data Integration Service ignores any condition on the default sequence flow at run time. When a sequence flow includes a condition and you select the sequence flow as the default, the Developer tool displays the condition as read-only.

Note: The business decisions that the gateway represents might require that the Data Integration Service performs no further work on the workflow data. To enable the workflow to continue when the objects between the gateways cannot run, create a sequence flow that connects the outgoing gateway to the incoming gateway. Select the sequence flow as the default sequence flow.

Splitting a Sequence Flow into Multiple Branches

Use a gateway to split a sequence flow into multiple sequence flows. Connect the gateway to multiple objects. Each connection is a sequence flow that begins a branch in the workflow.

1. Add a gateway to the workflow.
2. Add objects to the workflow to represent the actions that the Data Integration Service might take on different branches.
3. Configure the objects.
4. Connect the gateway to each object. Use the editor, or use the **Sequence Flows** tab.
   - In the editor, select the gateway and drag it to each object.
   - In the **Sequence Flows** tab, click **New**. In the **Connect Workflow Objects** dialog box, select the objects to connect to the gateway.
5. Select the gateway.
6. In the **Properties** view, click the **Sequence Flows** tab.
7. To set the default sequence flow, select a sequence flow and click **Set as Default**.

8. Define a condition expression in each outgoing sequence flow that is not the default sequence flow.
    - To display the condition editor for a sequence flow, click the arrow in the **Condition** column.
    - You can include workflow parameters and variables in the expression. Select a workflow parameter or variable from the **Inputs** tab in the condition editor, or type the parameter or variable name in the condition in the required syntax.
9. Use the arrows to define the order in which the Data Integration Service evaluates the sequence flows.

Use another gateway to merge the branches to a single sequence flow. Use the same type of gateway to create the branches and to merge the branches.

Related Topics:

- “Conditional Sequence Flows” on page 15

Merging Branches into a Single Sequence Flow

Use a gateway to merge the branches in a workflow into a single sequence flow. Connect the final objects on each branch to the gateway that you add.

1. Add a gateway to the workflow.

    Select the type of gateway that created the branches.
2. Connect the final object on each branch to the gateway.

After you connect the objects, you can connect the gateway to a single downstream object in the workflow.

CHAPTER 10

Workflow Recovery

This chapter includes the following topics:

Workflow Recovery Overview

Workflow recovery is the completion of a workflow instance from the point of interruption. A running workflow instance can be interrupted when an error occurs, when you cancel or abort the workflow instance, or when a Data Integration Service process shuts down unexpectedly.

The Data Integration Service tries to recover the previous workflow state if the service restarts after an unexpected shutdown. By default, the Data Integration Service does not recover a workflow instance that stopped during a Command task, Mapping task, or Notification task. In addition, the Data Integration Service cannot recover a workflow instance by default if you cancel the workflow instance or cancel a running task in the workflow instance. You can configure the recovery options on the workflow to enable the Data Integration Service to recover a workflow instance in such cases.

When you configure the workflow options, you can configure the workflow for manual recovery or automatic recovery. If you configure automatic recovery, the Data Integration Service restarts the workflow from the point of interruption without any human interaction. If you configure manual recovery, you can restart the workflow.

The Data Integration Service also considers the task recovery strategy that the workflow specifies for the task that was running when the workflow stopped. A task recovery strategy determines whether the Data Integration Service completes an interrupted task during a workflow run.

Note: Some errors are not recoverable. You cannot recover a workflow instance from a workflow error. You cannot recover a workflow that is in an aborted state. If a workflow instance is recoverable and you change the workflow metadata and redeploy the workflow application, the workflow instance is no longer recoverable.

Recoverable Workflow States

You can recover a workflow instance that you canceled and a workflow instance that was interrupted by a recoverable error. To recover a workflow, first enable the workflow for recovery.

You can cancel a workflow instance from the **Monitoring** tab in the Administrator tool or from the command line. Recoverable errors include an interruption to the Data Integration Service process that runs the workflow.

Nonrecoverable Workflow States

A workflow error can interrupt a running workflow instance. Workflow errors are not recoverable.

Workflow errors occur when the Data Integration Service cannot perform the following actions:

- Read the parameter file at the start of the workflow run to resolve the parameters. For example, a workflow error occurs if the parameter assigned to the workflow tracing level property contains a value that is not valid.
- Copy workflow parameter and variable values to task input. For example, a workflow error occurs if a string parameter value is assigned to an integer task input and the string value does not contain a number.

Workflow errors require that you change the workflow definition or change a parameter value in the parameter file to fix the error. When a workflow error occurs, the Data Integration Service immediately fails the workflow instance. You cannot recover the workflow instance even if the workflow is enabled for recovery.

Note: A workflow that ends when the active sequence flow reaches a Terminate event enters an aborted state. You cannot recover the workflow instance.

Task Errors and Recovery

A task error either interrupts the running workflow instance or causes the Data Integration Service to skip the task during the workflow instance run, depending on the recovery strategy.

Task errors are recoverable or nonrecoverable. Recoverable task errors are errors that you can fix without making changes to the workflow definition. Nonrecoverable task errors require that you change the workflow definition or change a parameter value in the parameter file. You can recover a workflow instance interrupted by a recoverable task error if you enabled recovery for the workflow.

Workflow tasks can use restart or skip as the recovery strategy. If the task uses restart as the recovery strategy, a task error interrupts the workflow instance. If a task uses skip as the recovery strategy, a task error causes the Data Integration Service to skip the task.

All tasks use restart as the recovery strategy except Create Cluster tasks, Mapping tasks, and Delete Cluster tasks. Create Cluster tasks always use skip as the recovery strategy. You can configure Mapping tasks and Delete Cluster tasks to use restart or skip as the recovery strategy.

Tasks can encounter the following recoverable or nonrecoverable errors:

Command task

All Command task errors are recoverable.

Human task

All Human task errors are recoverable.

Note: When the exceptionLoadCount variable value on a Human task is less than 1, the Human task generates no task data for review. An exceptionLoadCount variable value of less than 1 does not represent a task error.

Mapping task

A Mapping task encounters a nonrecoverable error when a workflow variable or a parameter assigned to a Mapping task configuration property contains a value that is not valid. For example, a workflow string parameter with a value of 8 is assigned to the Optimizer Level configuration property.

All other Mapping task errors are recoverable.

Notification task

A Notification task encounters a recoverable error in the following situations:

- When the Email Service is not available.
- When the email server properties are not configured correctly.

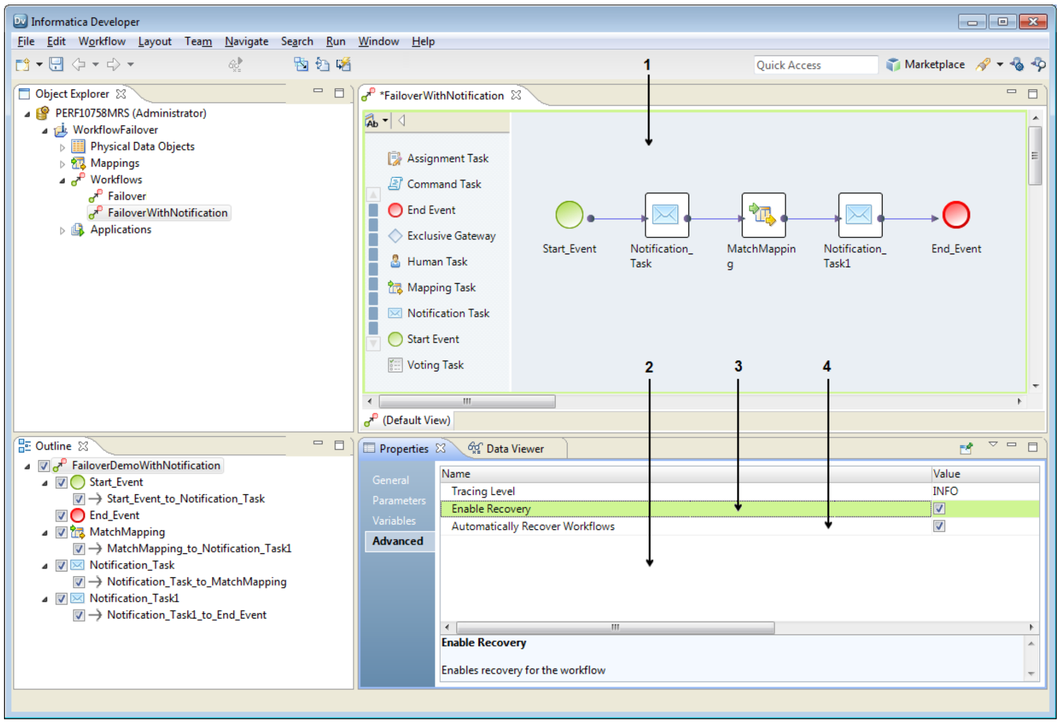
All other Notification task errors are nonrecoverable.

Workflow Recovery Strategies

When you enable a workflow for recovery, you can configure automatic recovery for any instance of the workflow that was interrupted by a recoverable error.

When you configure automatic recovery, the Data Integration Service applies the recovery strategy that you define when a recoverable event occurs. The Data Integration Service process recovers the workflow instances when the service process restarts.

The following image shows the workflow recovery options in the Developer tool:



Name	Value
Tracing Level	INFO
Enable Recovery	✓
Automatically Recover Workflows	✓

Enable Recovery

Enables recovery for the workflow

Use the following options when you define a workflow recovery strategy:

1. Workflow editor in the Developer tool
   The editor displays the workflow that you configure.
2. Advanced tab on the Properties view.
   The tab includes the Workflow recovery options.
3. Enable Recovery option.
   To enable recovery, select the option. When you select the option, you can select or clear the option to automatically recover the Workflow instances.
4. Automatically Recover Workflow option.
   To enable the Data Integration Service to automatically recover instances of the Workflow, select the option.

Workflow Recovery and Terminate Events

You cannot recover a workflow that ends in a Terminate event. A workflow enters an aborted state if the active sequence flow reaches a Terminate event.

Workflow Recovery on Grid

The status of a workflow that fails over from one node to another on a grid depends on the type of task that is running when the node fails.

A running workflow tries to continue to run during failover regardless of the workflow recovery strategy that you specify. If the currently running task is a Command task or a Mapping task at the point of failover, the workflow recovery strategy determines the workflow behavior.

If you specify an automatic recovery strategy for the workflow, the workflow reruns the Mapping task or the Command task from the start of the task. If you do not specify an automatic recovery strategy, the workflow is canceled and you must manually recover the workflow on the active node.

You can configure automatic recovery for a workflow that runs on a Data Integration Service grid. If the master service process shuts down unexpectedly, the master role fails over to another service process. The current master can automatically recover any workflow instance that was running during the failover and that you configured for automatic recovery.

Note: Do not configure a workflow for automatic recovery if the workflow includes a Human task. If the workflow is interrupted during a Mapping task that precedes a Human task, the recovered workflow reruns the Mapping task and adds conflicting data to the target database.

Task Recovery Strategies

The task recovery strategy determines whether the Data Integration Service reruns or skips a task that is interrupted or that encounters an error. Verify the recovery strategy for any Mapping task in the workflow.

A task can have one of the following recovery strategies:

Restart

When a task with a restart recovery strategy is interrupted or encounters a recoverable error, the Data Integration Service cancels the task and the workflow. The Data Integration Service restarts the task when the workflow recovers. When a task with a restart recovery strategy encounters a nonrecoverable error, the Data Integration Service aborts the task and the workflow.

When you enable recovery on a workflow, all tasks use a restart recovery strategy by default.

Define a restart strategy for any Mapping task that writes data for a Human task.

Skip

When a task with a skip recovery strategy encounters a recoverable error or a nonrecoverable error, the Data Integration Service skips the task. The Data Integration Service runs the next stage in the workflow. If the workflow does not encounter an error or an interruption, the workflow completes successfully.

If the workflow is not enabled for recovery, the Data Integration Service skips any task that encounters a task error.

Restart Recovery Behavior

When a task with a restart recovery strategy is interrupted, the task state can become Canceled, Aborted or Completed. The Data Integration Service can restart canceled tasks.

The following table describes the task and workflow instance states when a task with a restart recovery strategy is interrupted:

Interruption	Task State	Workflow State	Description
Task encounters a nonrecoverable error	Aborted	Aborted	The task and the workflow are aborted. The workflow is not recoverable.
Task encounters a recoverable error	Canceled	Canceled	The task and the workflow are canceled. The workflow is recoverable.

Interruption	Task State	Workflow State	Description
User aborts the workflow instance	Aborted	Aborted	The Data Integration Service aborts the task and workflow instance. The workflow is not recoverable.
User cancels the workflow instance	Completed/ Running	Canceled	The Data Integration Service cancels the workflow instance and finishes running the current task. The workflow is recoverable. Because the current task runs to completion, the workflow status might change to Canceled while the task is still running. When the task completes, the task enters a Completed state unless the task is a Human task. A Human task remains in a Running state until the workflow recovers. When the workflow recovers, the Human task state changes to Completed when the steps in the task are completed.
Service process shuts down unexpectedly	Canceled	Canceled	The task and the workflow are canceled. The workflow is recoverable. If the workflow is not configured for automatic recovery, the service process changes the task and workflow instance states to Canceled when the service process restarts. The interrupted task restarts when you recover the workflow. If the workflow is configured for automatic recovery, the service process recovers the workflow instance and restarts the interrupted task when the service process restarts. The service process changes the task and workflow instance states to Running. **Note:** Before the Data Integration Service restarts, the workflow state and the task state appear as Running, although the workflow and the task are no longer running. The Data Integration Service updates the workflow and task states when it restarts.

Skip Recovery Behavior

When a task with a skip recovery strategy is interrupted, the task state becomes Failed, Completed, Canceled or Aborted. When a workflow instance runs in recovery mode, the Data Integration Service skips failed and completed tasks.

The following table describes the task and workflow instance state when a task with a skip recovery strategy is interrupted:

Interruption	Task State	Workflow State	Description
Task encounters a recoverable or nonrecoverable error	Failed	Completed	The Data Integration Service fails the task. The Data Integration Service runs subsequent workflow objects if expressions in the conditional sequence flows evaluate to true or if the sequence flows do not include conditions. If the workflow instance finishes running without another interruption, the Data Integration Service updates the workflow state to Completed.
User aborts the workflow instance	Aborted	Aborted	The Data Integration Service fails the task and aborts the workflow instance. The workflow is not recoverable.

Interruption	Task State	Workflow State	Description
User cancels the workflow instance	Completed/ Running	Canceled	The Data Integration Service cancels the workflow instance and finishes running the current task. The workflow is recoverable. Because the current task runs to completion, the workflow status might change to Canceled while the task is still running. When the task completes, the task enters a Completed state unless the task is a Human task. A Human task remains in a Running state until the workflow recovers. When the workflow recovers, the Human task state changes to Completed when the steps in the task are completed.
Service process shuts down unexpectedly	Canceled	Canceled	The workflow and the task are canceled. The workflow is recoverable. If the workflow is not configured for automatic recovery, the service process changes the workflow instance state to Canceled when the service process restarts. When you recover the workflow, the service process reruns the interrupted task regardless of the skip recovery strategy. If the workflow is configured for automatic recovery, the service process recovers the workflow instance and reruns the interrupted task when the service process restarts. The service process sets the workflow instance state to Running. The service process restarts the interrupted task regardless of the skip recovery strategy. **Note:** Before the Data Integration Service restarts, the workflow state and the task state appear as Running, although the workflow and the task are no longer running. The Data Integration Service updates the workflow and task states when it restarts.

Human Task Restart Behavior

A Human task uses a restart recovery strategy. Unlike other task types, when an interrupted Human task restarts, it continues from the point at which it ceased operation. The Data Integration Service does not rerun the Human task from the beginning.

Consider the following rules and guidelines for Human task behavior:

- When a Data Integration Service runs a Human task, the service creates and distributes task instances to users and groups in the Informatica domain. Each task instance identifies a subset of the workflow data that a user must analyze in the Analyst tool.

  If the Data Integration Service does not distribute all task instances before the Human task is interrupted, the service continues to distribute the task instances when it restarts the task. The Human task does not restart from the beginning.
- If the Data Integration Service distributes all task instances before the Human task is interrupted, the Analyst tool users can work on the instances. However, if the Human task contains multiple steps, the task data cannot move from a current task instance to a task instance that another step defines. When the workflow restarts, the Human task continues from the point of interruption and the task data can move to the next step.
- If the workflow database connection fails during task instance creation, the workflow can enter one of several different states. The workflow might fail, or the Data Integration Service might not create all of the task instances that the Human task specifies.

If the workflow database connection fails before the Data Integration Service creates all of the Human task instances, stop the Data Integration Service. Verify that the workflow database connection is available, and restart the Data Integration Service. When the workflow recovers, the Data Integration Service tries to distribute any task instance that remains undistributed.

- If a workflow enters an aborted state, any Human task in the workflow enters a Completed state.

Note: If you cancel a workflow that contains a Human task and you try to redeploy the application, the Developer tool might stop responding. This might occur if you abort the workflow when the Human task is in the task creation phase. When the Human task returns control to the workflow, the Developer tool responds and deploys the application.

Rules and Guidelines for Workflow and Task Recovery

The workflow states and the task strategies that you define can influence the recovery behavior for a workflow instance.

Consider the following rules and guidelines for workflow and task recovery:

- If a task with a skip recovery strategy encounters an error, the Data Integration Service fails the task and moves to the next object in the workflow. The workflow is not interrupted.

  If the Data Integration Service is interrupted while a task with a skip recovery strategy is running, the workflow and the task enter a canceled state. When the workflow recovers, the Data Integration Service reruns the canceled task.
- If a task with a restart recovery strategy encounters a recoverable error, the Data Integration Service cancels the workflow. When the workflow recovers, the Data Integration Service restarts the task that was running when the error occurred.
- If you cancel a workflow while a task with a skip recovery strategy is running, the Data Integration Service cancels the workflow and waits for the task to finish. Upon recovery, the Data Integration Service runs the next object in the workflow.
- If a workflow enters an aborted state while a task with a skip recovery strategy is running, the Data Integration Service terminates the task and the workflow. The workflow is not recoverable.
- If the Data Integration Service that runs a task fails on a grid or in a high-availability environment, another Data Integration Service can recover the workflow. The service failover might occur before the workflow engine can register the final state of the tasks from the earlier workflow run. When the workflow recovers, the Data Integration Service might rerun one or more tasks that completed in the earlier run.

Steps to Configure Recovery

When you configure recovery, you can recover a workflow instance from the point of interruption.

To configure recovery, perform the following tasks:

1. Configure the workflow for recovery.

2. Configure a recovery strategy for each Mapping task in the workflow.

   Note: For each Mapping task with a restart recovery strategy, develop the mapping to support a complete restart of the task.

Configuring a Workflow for Recovery

When you configure a workflow for recovery, you can recover a workflow instance from a recoverable error.

1. Open the workflow in the editor.
2. In the **Properties** view, click the **Advanced** tab.
3. Select **Enable Recovery**.
4. To configure automatic recovery for instances of the workflow, select **Automatically Recover Workflows**.
5. Click **File** > **Save** to save the workflow.

Configuring a Task Recovery Strategy

By default, all tasks use a restart recovery strategy. Verify the task recovery strategy for any Mapping task in a workflow that you enable for recovery.

If the workflow is not enabled for recovery, the Data Integration Service ignores the task recovery strategy.

1. Open the workflow in the editor.
2. Select a Mapping task.
3. In the **Properties** view for the task, click the **Advanced** tab.
4. Select one of the following task recovery strategies:
   - **Restart task**. When a task with a restart recovery strategy is interrupted or encounters a recoverable error, the Data Integration Service cancels the task and the workflow. The Data Integration Service restarts the task when the workflow recovers.

     Define a restart strategy for any Mapping task that writes data for a Human task. If you configure a Mapping task with a restart recovery strategy, develop the mapping to support a complete restart of the task.
   - **Skip task**. When a task with a skip recovery strategy encounters a recoverable error or a nonrecoverable error, the Data Integration Service skips the task. The Data Integration Service runs the next stage in the workflow.
5. Repeat the steps for any other Mapping task in the workflow.
6. Click **File** > **Save** to save the workflow.

Parameter and Variable Values During Workflow Recovery

The Data Integration Service stores the current values of the workflow parameters and variables in the workflow database. When a recovered workflow runs, the Data Integration Service reuses the parameters and variables in use in the previous workflow run up to the point of the workflow interruption.

The parameters and variables have the following characteristics:

Workflow parameters

Workflow parameters use the values that the parameters had during the original workflow instance run. You cannot recover the workflow instance using different parameter values. To use different values in the parameter file, you must run another instance of the workflow using the infacmd wfs startWorkflow command.

User-defined workflow variables

User-defined workflow variables use the values that the variables had before the interrupted task began to run.

System workflow variables

The StartTime and InstanceID system workflow variables use the values that the variables had during the original workflow instance run.

The value of the UserName system workflow variable depends on whether a user recovers the workflow instance or whether the Data Integration Service recovers the workflow instance. If a user recovers the workflow instance, the UserName system workflow variable uses the name of the user who recovers the workflow instance. If the Data Integration Service recovers the workflow instance, the UserName system workflow variable continues to use the name of the user who ran the previous workflow instance.

Workflow Recovery Logs

The Data Integration Service appends log events to the current workflow log when you recover the workflow instance. When the recovered workflow instance includes a restarted Mapping task, the Data Integration Service creates a mapping log.

If the workflow instance runs on a Data Integration Service grid, the recovery of the workflow instance might run on a different node than the original workflow instance. If the recovery runs on a different node and the log directory is not shared, the Data Integration Service creates a log file with the same name on the current node.

Steps to Develop a Mapping for Restart

Mapping objects such as targets, Java transformations, and SQL transformations can affect external files or database tables. When you run a mapping multiple times, these mapping objects can cause unexpected results in the external files or database tables. You must develop the mapping to remove any external effects from the previous mapping run before the mapping restarts in a recovered workflow.

For example, the Data Integration Service stops unexpectedly while a Mapping task with a restart recovery strategy is running. The mapping writes 50 rows to the target before Data Integration Service stops. When you recover the workflow instance, the Mapping task does not recover at the point of interruption and start writing row number 51. Instead, the mapping begins again.

Before the Data Integration Service restarts an interrupted mapping, you must manually remove the rows that the earlier mapping wrote to the target. Or, configure the mapping to remove the target rows.

Note: If you opt to manually remove the target rows, do not configure the workflow for automatic recovery.

Remove Target Rows Manually

If the mapping writes to a shared table that contains data that you need to keep, you can manually remove the rows written in the original run and then recover the workflow instance.

If the mapping writes to a table that a Human task reads, you must manually remove the target rows. Multiple Human tasks can write data to the same database table. If you configure the mapping to remove target rows, you might erase data from multiple mappings. Use the workflow instance ID to identify and manually remove the rows written in the original workflow run. You can locate the workflow instance ID in the workflow properties in the Monitoring tool. Or, you can locate the workflow instance ID in the output of the infacmd wfs startWorkflow and listActiveWorkflowInstances commands.

Configure the Mapping to Remove Target Rows

If the mapping writes to a file or a table that does not contain data that you need to keep, you can configure the mapping to remove all data from the target file or table. If the mapping writes to a shared table that a Human task does not read, you can configure the mapping to run an SQL command to remove the rows written in the original run.

Configure the mapping to use one of the following methods to remove target rows:

Truncate a flat file target.

If you can remove all data from the target file, configure the mapping to truncate the target file before it writes to the file.

Truncate a relational target table.

If you can remove all data from the target table, configure the mapping to truncate the target before it loads data.

Run an SQL command on a relational target table.

If you need to keep some of the data in the target table, write an SQL command that the Data Integration Service runs to remove the rows written in the interrupted mapping run before it reads the source. For example, if the mapping writes sales data to a table that multiple departments share, write an SQL command that deletes all rows written for the Sales department. Or, if the mapping runs once daily, write an SQL command that deletes all rows written on the current date.

Configuring a Mapping to Truncate a Flat File Target

To support a complete restart of a mapping that writes to a flat file target, configure the write properties to truncate the target file before writing to the file.

1. Open the flat file data object in the editor.
2. Select the **Write** view.
3. Select the Input transformation.
4. In the **Properties** view, click the **Run-time** tab.
5. Clear **Append if Exists**.
6. Click **File** > **Save** to save the flat file data object.

Configuring a Mapping to Truncate a Relational Target

To support a complete restart of a mapping that writes to a relational target, configure the mapping to truncate the target before it loads data.

1. Select the Mapping task in the editor.
2. In the **Properties** view, click the **General** tab.
3. Click the name of the mapping that the task runs.

   The mapping opens.
4. Select the relational data object in the editor.
5. In the **Properties** view, click the **Advanced** tab.
6. Select **Truncate Target Table**.
7. Click **File** > **Save** to save the mapping.

Configuring a Mapping to Run an SQL Command

To support a complete restart of a mapping that writes to a relational target, write an SQL command that the Data Integration Service runs to delete target rows before it reads the source.

1. Select the Mapping task in the editor.
2. In the **Properties** view, click the **General** tab.
3. Click the name of the mapping that the task runs.

   The mapping opens.
4. Select the relational data object in the editor.
5. In the **Properties** view, click the **Advanced** tab.
6. In the **Value** column for the **PreSQL** property, click the **Open** button.

   The **SQL Query** editor appears.
7. Enter an SQL command and then click **OK**.

   The Developer tool does not validate the SQL.
8. Click **File** > **Save** to save the mapping.

Recovering Workflow Instances

To recover a canceled workflow instance, use the Monitoring tool or run the infacmd wfs recoverWorkflow command.

The Monitoring tool shows the status of running, completed, and interrupted workflow instances. Use the Monitoring tool to view logs of interrupted workflow instances and to investigate the cause of the interruption. After you fix any recoverable error, you can recover the interrupted workflow instance if it is enabled for recovery.

To recover a workflow instance from the command line, you must specify the workflow instance ID. You can view the workflow instance ID in the workflow properties in the Monitoring tool. Or, you can view the workflow instance ID in the output of the infacmd wfs startWorkflow command and the infacmd wfs listActiveWorkflowInstances command.

For example, the following command recovers an interrupted workflow instance with an ID of u-6-j4MwEeGxHO9AUHdw6A:

```
infacmd wfs recoverWorkflow -dn MyDomain -sn MyDataIntSvs -un MyUser -pd MyPassword -iid
u-6-j4MwEeGxHO9AUHdw6A
```

Summary of Workflow States After an Interruption

The workflow state and task state when a workflow instance is interrupted depend on the reason for the interruption and the recovery options that you select.

Recovery is not enabled.

The following table summarizes the workflow and task states when the workflow is not enabled for recovery:

Interruption	Task State	Workflow State
Workflow or task encounters any error	Failed	Completed
User aborts the workflow instance	Aborted	Aborted
User cancels the workflow instance	Completed	Canceled

Recovery is enabled. Interrupted task has a restart recovery strategy.

The following table summarizes the workflow and task states when the workflow is enabled for recovery and the interrupted task has a restart recovery strategy:

Interruption	Task State	Workflow State
Workflow encounters any error	Aborted	Aborted
Task encounters a nonrecoverable error	Aborted	Aborted
Task encounters a recoverable error	Canceled	Canceled
User aborts the workflow instance	Aborted	Aborted
User cancels the workflow instance	Completed	Canceled

Recovery is enabled. Interrupted task has a skip recovery strategy.

The following table summarizes the workflow and task states when the workflow is enabled for recovery and the interrupted task has a skip recovery strategy:

Interruption	Task State	Workflow State
Workflow encounters any error	Aborted	Aborted
Task encounters any error	Failed	Completed

Interruption	Task State	Workflow State
User aborts the workflow instance	Aborted	Aborted
User cancels the workflow instance	Completed	Canceled

Workflow Recovery Examples

The following examples describe interrupted workflow instances and describe how the Data Integration Service tries to recover each instance. In each example, the workflow is enabled for recovery and each task is a Command task, a Mapping task, or a Notification task.

The workflow encounters the following interruptions:

You cancel the workflow instance.

You cancel the workflow instance while Task 2 is running. The Data Integration Service completes Task 2 and then cancels the workflow instance. The three tasks and the workflow instance have the following states:

Task 1 (Completed) > Task 2 (Completed) > Task 3 (Not Started). Workflow is Canceled.

When the workflow recovers, the Data Integration Service passes over Task 1 and Task 2 and runs Task 3 for the first time.

You abort the workflow instance while a task with a skip recovery strategy is running.

You abort the workflow instance while Task 2 is running. Task 2 has a skip recovery strategy. The Data Integration Service aborts Task 2 and then aborts the workflow instance. The three tasks and the workflow instance have the following states:

Task 1 (Completed) > Task 2 (Aborted) > Task 3 (Not Started). Workflow is Aborted.

You cannot recover the workflow instance because you aborted the workflow.

The Data Integration Service process shuts down unexpectedly. The workflow is configured for recovery.

The Data Integration Service process shuts down while Task 2 is running. Task 2 is a Notification task. The three tasks and the workflow instance have the following states when the service process restarts:

Task 1 (Completed) > Task 2 (Canceled) > Task 3 (Not Started). Workflow is Canceled.

When the workflow recovers, the Data Integration Service passes over Task 1, restarts Task 2, and runs Task 3 for the first time.

A task with a restart recovery strategy encounters a recoverable error.

Task 2 has a restart recovery strategy and encounters a recoverable error. The service cancels Task 2 and then cancels the workflow instance. The three tasks and the workflow instance have the following states:

Task 1 (Completed) > Task 2 (Canceled) > Task 3 (Not Started). Workflow is Canceled.

When the workflow recovers, the Data Integration Service passes over Task 1, restarts Task 2, and runs Task 3 for the first time.

A task with a skip recovery strategy encounters an error. You cancel the workflow instance.

Task 2 has a skip recovery strategy. The task encounters an error and fails. The outgoing sequence flow does not contain a condition to verify that Task 2 succeeded. As a result, the Data Integration Service continues to run subsequent workflow objects. You cancel the workflow instance while Task 3 is running. Task 3 has a restart recovery strategy. The Data Integration Service cancels Task 3 and then cancels the workflow instance. The four tasks and the workflow instance have the following states:

Task 1 (Completed) >Task 2 (Failed) > Task 3 (Canceled) > Task 4 (Not Started). Workflow is Canceled.

When the workflow recovers, the Data Integration Service passes over Task 1 and Task 2, restarts Task 3, and runs Task 4 for the first time. During the recovery run, Task 3 and Task 4 complete successfully. The Data Integration Service updates the final workflow state to Completed.

A task with a skip recovery strategy encounters an error. You abort the workflow instance.

Task 2 has a skip recovery strategy. The task encounters an error and fails. The outgoing sequence flow does not contain a condition to verify that Task 2 succeeded. As a result, the Data Integration Service continues to run subsequent workflow objects. You abort the workflow instance while Task 3 is running. Task 3 has a restart recovery strategy. The Data Integration Service aborts Task 3 and then aborts the workflow instance. The four tasks and the workflow instance have the following states:

Task 1 (Completed) >Task 2 (Failed) > Task 3 (Aborted) > Task 4 (Not Started). Workflow is Aborted.

You cannot recover the workflow instance because you aborted the workflow.

A task with a skip recovery strategy encounters an error. The outgoing conditional sequence flow checks for task failure.

Task 1 has a skip recovery strategy. The task encounters an error and fails. The outgoing sequence flow contains a condition to verify that Task 1 succeeded. Because the condition returns false, the Data Integration Service stops processing subsequent workflow objects and completes the workflow instance. The three tasks and the workflow instance have the following states:

Task 1 (Failed) > Task2 (Not Started) > Task 3 (Not Started). Workflow is Completed.

You cannot recover the workflow instance because the workflow instance completed.

A task fails due to a task error in a workflow that does not support recovery.

Because the workflow does not specify a workflow recovery strategy, the Data Integration Service skips any task that fails due to a task error. The workflow can run to completion. The three tasks and the workflow instance have the following states:

Task 1 (Completed) > Task2 (Failed) > Task 3 (Completed). Workflow is Completed.

You cannot recover the workflow instance because the workflow instance completed.

A mapping in a Mapping task generates no exception data for a Human task. The active sequence flow in the workflow reaches a Terminate event.

Task 1 is a Mapping task and Task 2 is a Notification task. The Mapping task generates exception data for a downstream Human task. The Notification task sends an email that contains the number of exception rows that the mapping in the Mapping task generates. The workflow includes an Exclusive gateway that connects to the Human task and to a Terminate event.

The first sequence flow on the gateway connects to the Terminate event and includes a condition that evaluates the output from the Mapping task. The second sequence flow on the gateway connects to the Human task. Because the mapping that Task 1 specifies generates no exception data, the conditional sequence flow on the gateway triggers the Terminate event. The tasks and the workflow instance have the following states:

Task 1 (Completed) > Task 2 (Completed) > Terminate event. Workflow is Aborted.

You cannot recover the workflow because the workflow entered an aborted state by design.

CHAPTER 11

Workflow Administration

This chapter includes the following topics:

Workflow Administration Overview

After you deploy a workflow to a Data Integration Service, you can use the Administrator tool or the Monitor tool to administer and monitor the workflow job. You must have the appropriate privileges in order to perform these tasks.

You can monitor a workflow job in the following locations:

- Monitoring tool. In the Developer tool, click the **Menu** button in the **Progress** view and select **Monitor Jobs**. Select the Data Integration Service that runs the workflow and click **OK**. The Monitoring tool opens.
- Administrator tool. To monitor a workflow job in the Administrator tool, click the **Monitor** tab.

When you monitor a workflow job, you can view summary statistics or execution statistics for the job. The **Summary Statistics** view displays a graphical overview of the status of workflow jobs in the domain.

The **Execution Statistics** view displays information about workflow jobs and workflow objects.

When you select a workflow job in the contents panel on the **Execution Statistics** view, you can complete the following tasks:

- View a graph of the workflow.
- View properties for workflow objects.
- Cancel or abort the workflow instance.
- Recover an interrupted workflow instance.

- View logs for the workflow instance.

Workflow Graph

You can view the details of a workflow that you run in the Monitoring tool in a graphical form.

After you run a workflow, you can see the graphical view of the workflow in the Monitoring tool. In the workflow graph, you can see the sequential run of the mapping tasks in the workflow. The workflow graph enables you to view the failure points in a workflow at a glance.

In the workflow graph, you can view the following details of a workflow:

- Mapping tasks in the workflow
- Task details
- Recovery details

You can perform the following tasks from the workflow graph:

- Abort a running workflow
- Cancel a running workflow
- Recover a failed workflow
- View the workflow logs

Viewing a Workflow Graph

You can view a workflow graph that shows the sequential run of the mapping tasks in the workflow.

1. Click the **Execution Statistics** view.
2. In the Domain Navigator, expand an application.
3. Select the **Workflows** folder.

   A list of workflows appears in the contents panel.
4. Select the workflow that you want to view.
5. Click **Actions** > **View Workflow Graph**.

   The workflow graph appears in a new window.

View Workflow Objects

When you expand a workflow instance in the contents panel, you can view properties about workflow objects, such as the name, state, start time, and elapsed time for the object.

Workflow objects include events, tasks, and gateways. When you monitor workflows, you can monitor the tasks that run in a workflow instance. The **Monitor** tab does not display information about events or gateways in the workflow instance.

If an expression in a conditional sequence flow evaluates to false, the Data Integration Service does not run the next object or any of the subsequent objects in that branch. The **Monitor** tab does not list objects that do not run in the workflow instance. When a workflow instance includes objects that do not run, the instance can still successfully complete.

You can expand a task in the contents panel to view information about the work item run by the task. For example, if the workflow contains a mapping task, you can view throughput and resource usage statistics for the mapping run.

Viewing Summary Statistics for Workflow Objects

You can view throughput and resource usage statistics for mapping objects in workflows that run in separate local processes.

1. Click the **Execution Statistics** view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Expand an application and select the **Workflows** folder.

   A list of workflows appears in the contents panel.
4. Expand a workflow that contains a mapping object.
5. Expand the mapping task and select the mapping.
6. In the details panel, click the **Summary Statistics** view.

   The **Summary Statistics** view displays throughput and resource usage statistics for the source and target.

Optionally, you can sort the statistics in ascending or descending order. Click a column header to sort the column in ascending order. Click the column header again to sort the column in descending order.

Viewing Detailed Statistics for Workflow Objects

You can view graphs of the throughput and resource usage for mapping objects in workflows that run in separate local processes. Detailed statistics appear for jobs that run longer than one minute.

1. Click the **Execution Statistics** view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Expand an application and select the **Workflows** folder.

   A list of workflows appears in the contents panel.
4. Expand a workflow that contains a mapping object.
5. Expand the mapping task and select the mapping.
6. Click the **Detailed Statistics** view in the details panel.

   The **Detailed Statistics** view displays the throughput graph and resource usage graphs.

Optionally, you can complete the following tasks in the **Detailed Statistics** view:

Task	Description
Enlarge a graph	Move the cursor over a graph, and then click the magnifying glass icon.
Enlarge a section of an enlarged graph	Drag the cursor to select an area to enlarge.

Task	Description
Switch between rows and bytes in the throughput graph	Click the Bytes option or the Rows option.
Choose which statistics are plotted on the throughput graph	In the throughput field, select the sources and targets that you want to view.

Workflow States

When you monitor a workflow instance, you can view the state of the workflow instance. If a workflow instance recovers after a task is interrupted, the Monitor adds an entry for the task instance that runs in the recovered workflow.

A workflow instance can have one of the following states:

Aborted

A workflow instance aborts when you choose to abort the workflow instance from the Monitoring tool or using the infacmd wfs abortWorkflow command. You can also choose to abort a running workflow instance when you stop the application that contains the workflow or when you disable the workflow in the application.

Note: A workflow instance also aborts if the active sequence flow in the workflow reaches a Terminate event.

Canceled

You choose to cancel the workflow instance from the **Monitor** tab or by using the infacmd wfs cancelWorkflow command.

The workflow can also enter a canceled state if the Data Integration Service shuts down unexpectedly. If the workflow is not configured for automatic recovery, the service process changes the workflow instance state to Canceled when the service process restarts. Before the Data Integration Service restarts, the workflow state and the task state appear as Running, although the workflow and the task are no longer running. If the workflow is configured for automatic recovery, the service process recovers the workflow instance and reruns the interrupted task when the service process restarts. The service process sets the workflow instance state to Running.

Completed

The Data Integration Service successfully completes the workflow instance. A completed workflow instance might indicate that all tasks, gateways, and sequence flow evaluations either successfully completed or were in a branch that did not run.

A workflow can also enter a Completed state if a Command, Mapping, Notification, or Human task encounters a recoverable error or nonrecoverable error. When the task encounters the error, the Data Integration Service fails the task. The Data Integration Service runs subsequent workflow objects if expressions in the conditional sequence flows evaluate to true or if the sequence flows do not include conditions. If the workflow instance finishes running without another interruption, the Data Integration Service updates the workflow state to Completed.

When the task fails, the Data Integration Service continues to run additional objects in the workflow instance if expressions in the conditional sequence flows evaluate to true or if the sequence flows do not include conditions. If the workflow instance finishes running without another interruption, the Data Integration Service updates the workflow state to completed. A completed workflow instance can contain both failed and completed tasks.

Failed

A workflow instance fails when a workflow error occurs. Workflow errors can occur when the Data Integration Service reads the parameter file at the start of the workflow run, copies workflow parameter and variable values to task input, or evaluates expressions in conditional sequence flows. In addition, a workflow error occurs if an Assignment task or a gateway fails.

When a workflow error occurs, the Data Integration Service stops processing additional objects and fails the workflow instance immediately. Workflow errors are nonrecoverable.

Running

The Data Integration Service is running the workflow instance.

Workflow Object States

Workflows include tasks and gateways. When you monitor a workflow instance, you can view the state of the tasks that run in the workflow instance.

Tasks can have one of the following states:

Aborted

A task aborts in the following situations:

- The task encounters an nonrecoverable error.
- You abort the workflow instance.

  When you abort the workflow instance, the Data Integration Service first aborts the task and then aborts the workflow instance.

  If you choose to abort the workflow instance while an Assignment task is running, the Data Integration Service completes running the task. The Data Integration Service then aborts the workflow instance and does not start running other objects.

Completed

The Data Integration Service successfully completes the task.

Failed

A task fails in the following situations:

- Any task in a workflow not enabled for recovery encounters any type of error.
- An Assignment task in a workflow enabled for recovery encounters any type of error.
- A Command, Mapping, Notification, or Human task with a restart recovery strategy in a workflow enabled for recovery encounters a non recoverable error.
- A Mapping task with a skip recovery strategy in a workflow enabled for recovery encounters any type of error.

Note: A workflow can complete if a task fails. The Data Integration Service runs subsequent workflow objects if expressions in the conditional sequence flows evaluate to true or if the sequence flows do not include conditions. If the workflow instance finishes running without another interruption, the Data Integration Service updates the workflow state to Completed.

Running

The Data Integration Service is running the task.

Mapping Task Work Item States

When you expand a Mapping task, you can view the state of the mapping run. When you expand a restarted Mapping task, you can view the mapping jobs run for each recovery attempt of the workflow instance. If a workflow instance recovers after a Mapping task is interrupted, the Monitor adds an entry for the task instance that runs in the recovered workflow.

You can also view the state of the mapping run from the workflow graph of the workflow that contains the mapping task.

Mappings run by a Mapping task can have one of the following states:

Aborted

The Mapping task aborts while the mapping is running because you choose to abort the workflow instance.

Completed

The Data Integration Service successfully completes the mapping.

Failed

The mapping encounters an error. The mapping and the Mapping task appear as Failed in the Monitor. The states do not depend on the Mapping task recovery strategy.

Running

The Data Integration Service is running the mapping.

Canceling or Aborting a Workflow

You can cancel or abort a workflow instance at any time. You might want to cancel or abort a workflow instance that stops responding or that is taking an excessive amount of time to complete.

When you cancel a workflow instance, the Data Integration Service finishes processing any running task and then stops processing the workflow instance. The service does not start running any subsequent workflow objects.

When you abort a workflow instance, the Data Integration Service attempts to kill the process on any running task. If an Assignment task or a gateway is running, the Data Integration Service completes the task or gateway. After the task aborts or completes, the service aborts the workflow instance. The service does not start running any subsequent workflow objects.

You can also cancel or abort a workflow from the workflow graph.

1. Click the **Execution Statistics** view.
2. In the Navigator, expand a Data Integration Service.

3. Expand an application and select **Workflows**.

   A list of workflow instances appears in the contents panel.
4. Select a workflow instance.
5. Click **Actions** > **Cancel Selected Workflow** or **Actions** > **Abort Selected Workflow**.

Workflow Recovery

Workflow recovery is the completion of a workflow instance from the point of interruption.

When a workflow is enabled for recovery, you can recover a workflow instance if a task encounters a recoverable error, if you cancel the workflow instance, or if the Data Integration Service process shuts down unexpectedly.

View the workflow log to identify the cause of the interruption. After fixing any recoverable errors, you can recover the interrupted workflow instance if it is enabled for recovery.

You cannot change a workflow definition between the interrupted run and the recovery run. If a workflow instance has a recoverable state and you change the workflow metadata in the Developer tool and redeploy the application that contains the workflow, then the workflow instance is no longer recoverable.

The Data Integration Service tries to recover the previous workflow state if the service restarts after an unexpected shutdown. By default, the Data Integration Service does not recover a workflow instance that stopped during a Command task, Mapping task, or Notification task. In addition, the Data Integration Service cannot recover a workflow instance by default if you cancel the workflow instance or cancel a running task in the workflow instance. You can configure the recovery options on the workflow to enable the Data Integration Service to recover a workflow instance in such cases.

When you configure the workflow options, you can configure the workflow for manual recovery or automatic recovery. If you configure automatic recovery, the Data Integration Service restarts the workflow from the point of interruption without any human interaction. If you configure manual recovery, you can restart the workflow.

When a workflow instance recovers or when you recover a workflow instance, the Data Integration Service restarts the task. The service continues processing the subsequent workflow objects. If a workflow instance recovers after a task is interrupted, the Monitor adds an entry for the task instance that runs in the recovered workflow. For example, if a workflow recovers three times and restarts a Mapping task each time, the Monitor contains three entries for the Mapping task.

Recovery Properties

The read-only recovery properties display for each workflow instance. You configure the recovery properties for the workflow definition in the Developer tool. You cannot change the values of the properties for the workflow instance.

The following table describes the read-only recovery properties for a workflow instance:

Property	Description
Recovery Enabled	Indicates that the workflow is enabled for recovery.
Automatically Recover Workflows	Indicates that the Data Integration Service process tries to automatically recover workflow instances that were interrupted. The workflow recovery starts after the Data Integration Service process restarts.

Recovering a Workflow

You can recover interrupted workflow instances that are enabled for recovery.

1. Click the **Execution Statistics** view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Expand an application and select **Workflows**.

   A list of workflow instances appears in the contents panel.
4. Select the interrupted workflow instance that you want to recover.
5. Click **Actions** > **Recover Selected Workflow**.

   Monitor the state of the workflow recovery run in the contents panel.

Workflow Logs

The Data Integration Service generates log events when you run a workflow. Log events include information about workflow errors, task progress, and the setting of workflow variables. Log events also include the analyses of the links that the Data Integration Service evaluates in a sequence flow.

If a workflow instance includes a Mapping task, the Data Integration Service generates a separate log file for the mapping. The mapping log file includes any errors encountered during the mapping run and load summary and transformation statistics.

You can view the workflow and mapping logs from the Monitor tab.

When you recover an interrupted workflow instance, the Data Integration Service appends log events to the current workflow log. When the recovered workflow instance includes a Mapping task that is restarted, the Data Integration Service creates a mapping log.

If the workflow runs on a grid, the recovery of the workflow instance might run on a different node than the original workflow instance run. If the recovery runs on a different node and the log directory is not in a shared location, the Data Integration Service creates a log file with the same name on the current node.

Workflow Log Information

The information in the workflow log file represents the sequence of events that occur when the workflow runs.

The Data Integration Service writes information to the workflow log when the following types of event occur:

- The Data Integration Service starts to run a task or another object in the workflow.
- A task or another object in the workflow is in progress.
- The Data Integration Service finishes running a task or another object in the workflow.
- The Data Integration Service sets or updates a workflow variable.
- The Data Integration Service evaluates the links in a sequence flow and identifies the correct path for the workflow process.
- The workflow encounters a workflow error.

Viewing Logs for a Workflow

You can download the log for a workflow instance to view the workflow instance details.

1. In the Administrator tool, click the **Monitor** tab.
2. Click the **Execution Statistics** view.
3. In the Domain Navigator, expand a Data Integration Service.
4. Expand an application and select **Workflows**.

   A list of workflow instances appears in the contents panel.
5. Select a workflow instance.
6. Click **Actions** > **View Logs for Selected Object**.

   A dialog box appears with the option to open or save the log file.

Viewing Logs for a Mapping Run in a Workflow

You can download the log for a mapping run in a workflow to view the mapping details.

1. Click the **Execution Statistics** view.
2. In the Domain Navigator, expand a Data Integration Service.
3. Expand an application and select **Workflows**.

   A list of workflow instances appears in the contents panel.
4. Expand a workflow instance.
5. Expand a Mapping task, and then select the mapping run by the task.
6. Click **Actions** > **View Logs for Selected Object**.

   A dialog box appears with the option to open or save the log file.

INDEX