

# HP Operations Orchestration

For Windows and Linux

HP OO Software Version 10.01

## Studio Authoring Guide

Document Release Date: August 2013

Software Release Date: August 2013



## Legal Notices

### Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

### Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

### Copyright Notice

© Copyright 2013 Hewlett-Packard Development Company, L.P.

### Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

AMD and the AMD Arrow symbol are trademarks of Advanced Micro Devices, Inc.

Google™ and Google Maps™ are trademarks of Google Inc.

Intel®, Itanium®, Pentium®, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft®, Windows®, Windows NT®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

## Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

**<http://h20230.www2.hp.com/selfsolve/manuals>**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

**<http://h20229.www2.hp.com/passport-registration.html>**

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

# Support

Visit the HP Software Support Online web site at:

**<http://www.hp.com/go/hpsoftwaresupport>**

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

**<http://h20229.www2.hp.com/passport-registration.html>**

To find more information about access levels, go to:

**[http://h20230.www2.hp.com/new\\_access\\_levels.jsp](http://h20230.www2.hp.com/new_access_levels.jsp)**

## Disclaimer for PDF Version of Online Help

This document is a PDF version of the online help. This PDF file is provided so you can easily print multiple topics from the help information or read the online help in PDF format.

**Note:** Some topics do not convert properly to PDF, causing format problems. Some elements of online help are completely removed from the PDF version. Those problem topics can be successfully printed from within the online help.

# Contents

Studio Authoring Guide .....	1
Contents .....	6
Welcome to HP Operations Orchestration Studio Authoring Guide .....	9
Visual Overview of HP OO Studio .....	10
Getting Started with HP OO Studio – Major Steps in the Workflow .....	20
Adjusting the Appearance of the HP OO Studio Window .....	21
Best Practices .....	23
General Best Practices .....	23
Best Practices for Sharing Content .....	24
Best Practices for Naming .....	26
Best Practices for Flows .....	28
Best Practices for Operations .....	29
Best Practices for Steps .....	31
Best Practices for Transitions .....	33
Best Practices for Inputs .....	34
Best Practices for Debugging .....	34
Best Practices for Configuring Studio .....	35
Best Practices for Descriptions .....	35
Best Practices for Source Control Management .....	42
Working with Different Languages in HP OO Studio - Localization .....	43
Changing the Current Studio Locale .....	43
Studio Display Language .....	44
Content Pack Localization .....	45
Projects Localization .....	45
Creating a New Content Pack .....	45
Working with Projects .....	46
Managing Projects .....	46
Managing Folders in the Project Pane .....	52
Working With Source Control .....	53

Creating an Initial Source Control Repository .....	53
Working with Source Control in HP OO Studio .....	54
SCM Changes Panel .....	57
Working with Multiple Authors .....	59
Reference Material .....	64
Working with Content Packs .....	68
Importing Content Packs to a Project .....	68
Managing Content Packs in a Project .....	70
Managing Configuring Items .....	74
Configuring Categories .....	74
Configuring Domain Terms .....	76
Configuring Group Aliases .....	79
Configuring Scriptlets .....	82
Configuring Selection Lists .....	85
Configuring System Accounts .....	88
Configuring System Filters .....	90
Configuring System Properties .....	95
Authoring a Flow – Basics .....	99
Creating a Flow – Step-by-Step .....	99
Creating a New Flow .....	103
Creating Steps in a Flow .....	106
Adjusting the Appearance of a Flow .....	112
Modifying a Flow .....	115
Creating Input .....	120
Specifying the Input Source .....	127
Evaluating Input Data .....	134
Creating Transitions .....	135
Setting Responses .....	139
Creating Outputs and Results .....	149
Setting Operation Outputs .....	150
Setting Step Results .....	153

Filtering Output and Results .....	158
Working with Variables .....	175
Creating Return Steps .....	181
<b>Advanced Authoring .....</b>	<b>185</b>
Creating a Subflow Within a Flow .....	185
Creating a Flow with Parallel Split Steps .....	188
Creating a Flow with Multi-Instance Steps .....	192
Using Scriptlets in a Flow .....	200
Using Regular Expressions in a Flow .....	204
<b>Validating Content .....</b>	<b>209</b>
Validating Flows in the Problems Pane .....	209
Testing and Debugging a Flow .....	210
Debugging Complex Flows .....	222
Debugging Central with Studio .....	222
What do you want to do? .....	223
<b>Exporting a Content Pack .....</b>	<b>225</b>
<b>Managing Flows and Operations .....</b>	<b>227</b>
Creating Operations .....	227
Finding a Flow or Operation .....	235
Copying Flows and Operations .....	240
Changing Between Hard Copy and Soft Copy .....	242
Replacing a Plugin in a Hard Copy .....	242
Finding Out How Flows and Operations are Used .....	243
Generating Documentation about Flow and Operations .....	245
Managing Version History of Flows and Operations .....	250
Bookmarking Flows and Operations .....	252
<b>Troubleshooting for Those Upgrading from HP OO 9.x .....</b>	<b>256</b>
Where's the Studio User Interface Item? .....	256
Comparison of versions HP OO 9.x and 10.00 .....	257

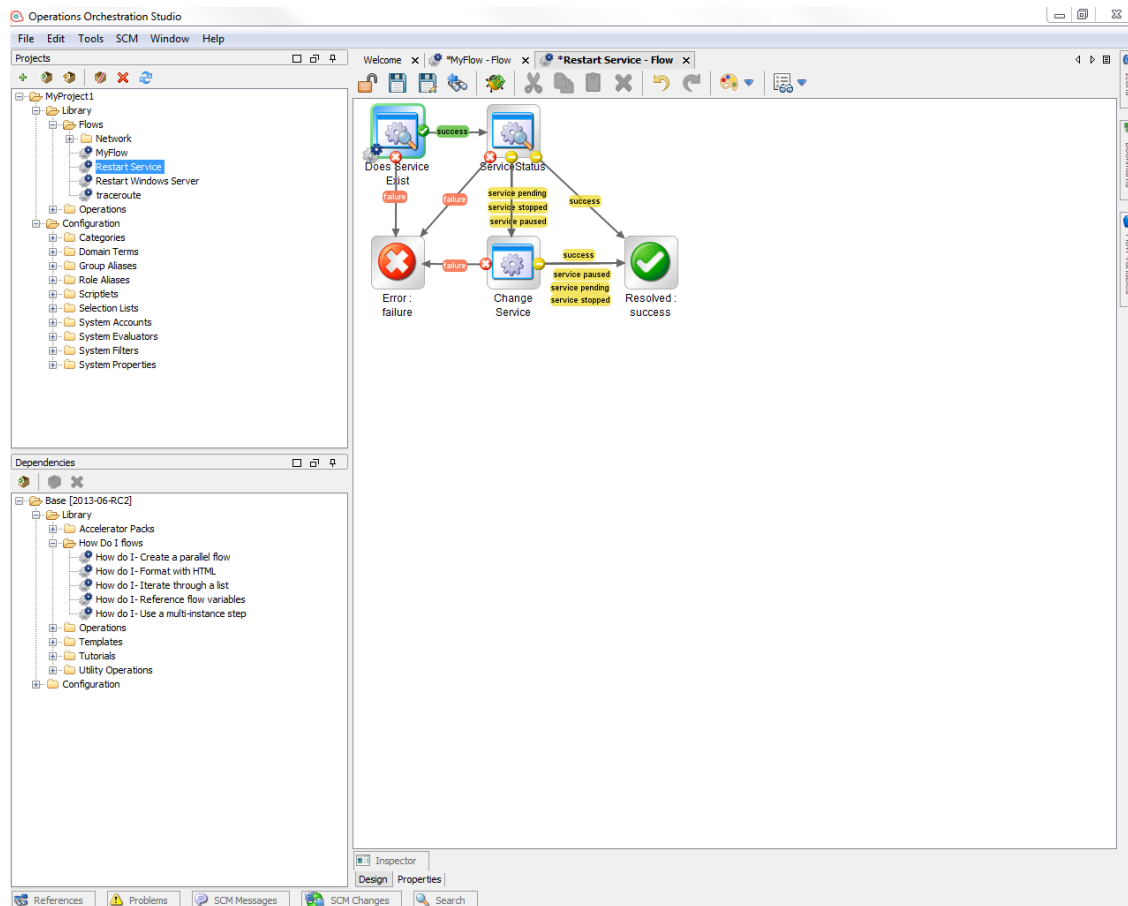


# Welcome to HP Operations Orchestration Studio Authoring Guide

HP OO Studio is a standalone authoring program used for creating, modifying, and testing flows.

Visual Overview of HP OO Studio .....	10
Getting Started with HP OO Studio – Major Steps in the Workflow .....	20
Adjusting the Appearance of the HP OO Studio Window .....	21
Best Practices .....	23
General Best Practices .....	23
Best Practices for Sharing Content .....	24
Best Practices for Naming .....	26
Best Practices for Flows .....	28
Best Practices for Operations .....	29
Best Practices for Steps .....	31
Best Practices for Transitions .....	33
Best Practices for Inputs .....	34
Best Practices for Debugging .....	34
Best Practices for Configuring Studio .....	35
Best Practices for Descriptions .....	35
Best Practices for Source Control Management .....	42
Working with Different Languages in HP OO Studio - Localization .....	43
Changing the Current Studio Locale .....	43
Studio Display Language .....	44
Content Pack Localization .....	45
Projects Localization .....	45
Creating a New Content Pack .....	45

## Visual Overview of HP OO Studio



The main elements of Studio are the:

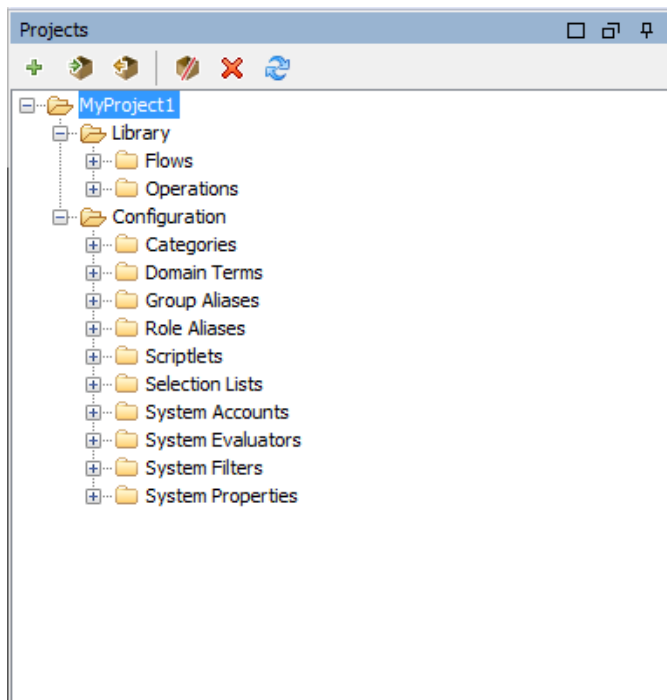
- **Projects** pane (on the left), which shows the project you're working in, and displays the editable flows, operations, and other HP OO objects that you can use in the project.
- **Dependencies** pane (on the left), which includes the imported content packs. In this pane, you can import, delete and close the content packs.
- Authoring pane (in the center). When a flow is opened in the authoring pane, the following three tabs are available at the bottom of the authoring pane:
  - **Design** tab, where you can work on the flow diagram
  - **Properties** tab, where you can set the properties for flows, operations, and configuration objects
  - **Inspector** tab, where you can set the properties for individual steps and transitions (only available when the **Design** tab is open)

- **Icons** pane (on the right), which contains collections of icons that you use for operations or steps. Open this pane by clicking the **Icons** tab.
- **Bookmarks** pane (on the right), where you can store shortcuts to favorite operations and flows. Open this pane by clicking the **Bookmarks** tab.
- **Flow Variables** pane (on the right), which shows the flow variables that are used in the flow and lists and describes how each flow variable is used. Open this pane by clicking the **Flow Variables** tab.
- **References** pane (on the bottom), which shows how flows and operations are used in existing flows. Open this pane by clicking the **References** tab.
- **Problems** pane (on the bottom), which displays problems with a selected flow or operation. Open this pane by clicking the **Problems** tab.
- **SCM Messages** pane (on the bottom), displays source control related messages. Open this pane by clicking the **SCM Messages** tab. See [Working with Source Control](#) for more information.
- **SCM Changes** pane (on the bottom), displays the latest source control changes. Open this pane by clicking the **SCM Changes** tab. See [Working with Source Control](#) for more information.
- **Search** pane (on the bottom), which enables you to search for a flow or operation. Open this pane by clicking the **Search** tab.

## Projects pane

The **Projects** pane contains the project tree—a hierarchical folder structure containing the editable content of your project:

- The **Library** folder, which holds flows and operations.
- The **Configuration** folder, which holds other HP OO objects (filters, scriptlets, system properties, and so on) that you can use to process operation results, create reports, and facilitate the running of flows.

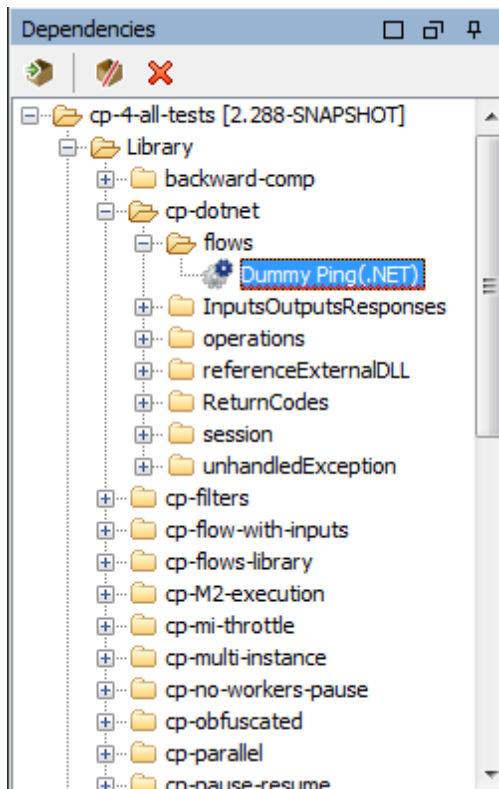


GUI item	Description
<b>New Project</b>	Creates a new project.
<b>Import Project</b>	Browse to and import an existing project from a different workspace.
<b>Create Content Pack</b>	Creates a content pack from the selected project.
<b>Delete</b>	Permanently deletes the selected project from the workspace.
<b>Open</b>	Opens the currently selected closed project.
<b>Close</b>	Closes the currently selected project, so that it is grayed out.
<b>Refresh</b>	Refreshes the files in the currently select project.

For information about working with projects, see ["Working with Projects" on page 46](#).

## Dependencies pane

The **Dependencies** pane displays the available content packs, with folders that contain the operations and flows.

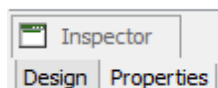


## Authoring pane

The authoring pane is the central area in Studio, where you work on flow diagrams, adding steps and the connections between them, and setting properties that determine how flows work.

When a flow is opened in the authoring pane, the following three tabs are available:

- **Design** tab, for working on the flow diagram, adding steps and the connections between them
- **Properties** tab, to display the **Properties** sheets, where you can set the properties for flows and operations, as well as configuration objects such as selection lists, filters, and scriptlets.
- **Inspector** tab, to display the Inspector, where you can set the properties for individual steps and transitions







## Authoring pane toolbar

When a flow is opened in the authoring pane, and the **Design** tab is open, the authoring pane toolbar is available.

The authoring pane toolbar buttons provide shortcuts for a number of tasks.



Button	What it does
<b>Step Palette</b> 	Opens the <b>Step</b> palette to drag step objects onto the canvas
<b>View Options</b> 	Opens the <b>View Options</b> palette
<b>Find this object in Library</b> 	Expands the Library tree to select the flow or operation that you're working on
<b>Debug Flow</b> 	Opens the Debugger and starts a run of the current flow





For information about working with the authoring pane, see ["Authoring a Flow – Basics" on page 99](#).





## Step palette

The **Step** palette contains buttons for dragging return steps, parallel split steps, multi-instance steps, and callouts onto the flow. Display the **Step** palette by clicking the **Step Palette** button

 from the authoring pane toolbar.



Button	Description
<b>Success</b> 	Lets you drag a <b>Success</b> return step to the flow.
<b>Diagnosed</b> 	Lets you drag a <b>Diagnosed</b> return step to the flow.
<b>No Action Taken</b> 	Lets you drag a <b>No Action Taken</b> return step to the flow.
<b>Failure</b> 	Lets you drag a <b>Failure</b> return step to the flow.


Button	Description
<b>Parallel Split Step</b> 	Lets you drag a parallel split step to the flow.
<b>Multi-instance Step</b> 	Lets you drag a multi-instance step to the flow.
<b>Callout</b> 	Lets you drag a callout to the flow, providing information to users.
<b>Docking bar</b> 	Click to dock and undock the palette.

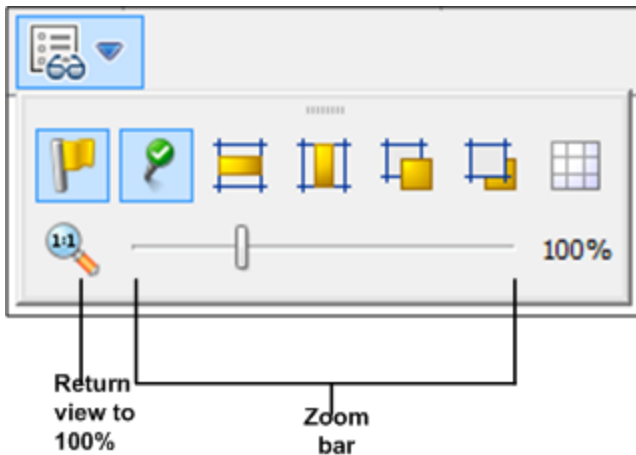
For information about working with return steps, see ["Creating Return Steps" on page 181](#).



For information about parallel split steps and multi-instance steps, see ["Advanced Authoring" on page 185](#).







## View Options palette

The **View Options** palette contains buttons for changing the appearance of the flow on the

authoring pane. Display the **View Options** palette by clicking the **View Options** button  from the authoring pane toolbar.



Button	Description
<b>Show/Hide Labels</b> 	Shows or hides response labels on objects
<b>Show/Hide Connected Response Icons</b> 	Shows or hides response icons on objects

Button	Description
<b>Align Selection Horizontally</b> 	Aligns selected steps horizontally
<b>Align Selection Vertically</b> 	Aligns selected steps vertically
<b>Bring to Front</b> 	Moves the selected object to the front of the stack
<b>Send to Back</b> 	Moves the selected object to the back of the stack
<b>Show/Hide Grid</b> 	Reveals the authoring pane grid, which you can use for arranging steps. When you stop dragging a step, it snaps to the nearest position on the grid.
<b>Docking bar</b> 	Click to dock and undock the palette.

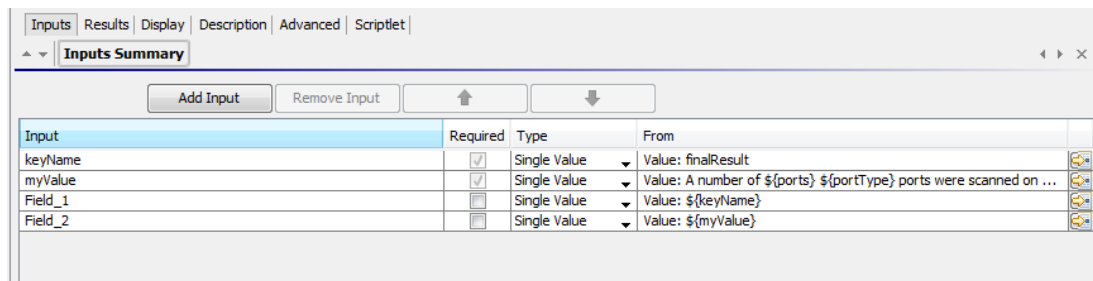
For information about working with the view options, see ["Adjusting the Appearance of a Flow" on page 112](#).

## Object Properties sheets

The **Properties** sheets for flows, operations, and configuration objects are the editors in which you add, remove, or change values for the objects. For most objects in the Library, the **Properties** sheet is the interface you use to work with the object. In addition to the fields that you can edit, **Properties** sheets provide the UUID and information about the version of the object.

After you modify the properties of an operation in the **Properties** sheet, the changes affect all steps that you create from this operation, including steps that were created earlier from this operation.

- To display the **Properties** sheet for a flow, open the flow in the authoring pane and click the **Properties** tab.
- To display the **Properties** sheet for an operation or configuration object, right-click the operation or object in the Library and select **Properties**.



Input	Required	Type	From
keyName	<input checked="" type="checkbox"/>	Single Value	Value: finalResult
myValue	<input checked="" type="checkbox"/>	Single Value	Value: A number of \${ports} \${portType} ports were scanned on ...
Field_1	<input type="checkbox"/>	Single Value	Value: \${keyName}
Field_2	<input type="checkbox"/>	Single Value	Value: \${myValue}

For information about working with the **Properties** sheet, see ["Creating Input" on page 120](#) and ["Setting Operation Outputs" on page 150](#).



## Step Inspector

The Step Inspector is similar to the **Properties** sheet for an operation, but it relates to a single step in a flow. If you modify the properties of a step in the Step Inspector, the changes only affect this step, which is an instance of the operation.

The screenshot shows the 'Step Inspector' window with the 'Inputs Summary' tab selected. The 'Step Name' is 'Set Inactive Users By Group'. Below the tabs, there are buttons for 'Add Input', 'Remove Input', and arrows for reordering. A table lists the inputs for the step:

Assign To Input	Required	Type	From
host	<input checked="" type="checkbox"/>	Single Value	Value: \${dnsServer}
baseDN	<input checked="" type="checkbox"/>	Single Value	Value: \${domainDN}
DN	<input checked="" type="checkbox"/>	Single Value	Value: CN=Users,\${domainDN}
username	<input type="checkbox"/>	Single Value	Value: \${altuser}
password	<input type="checkbox"/>	Single Value	Value: *****
daysOld	<input checked="" type="checkbox"/>	Single Value	Value: 30
groupDN	<input checked="" type="checkbox"/>	Single Value	Value: CN=Domain Admins,CN=Users,\${domainDN}

For information about working with the Step Inspector, see ["Creating Input" on page 120](#).

## Transition Inspector

The Transition Inspector is used to configure the transitions between steps. To display the Transition Inspector, right-click the line that leads between two steps and select **Properties**.

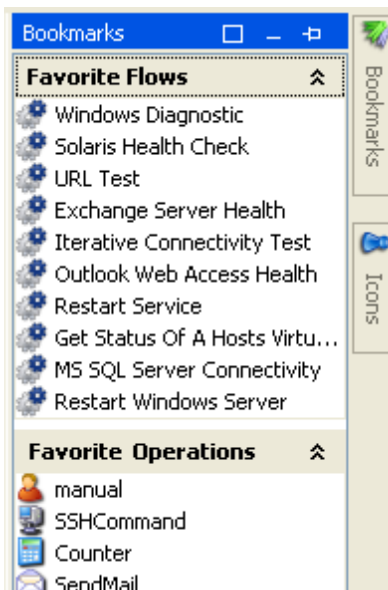
The screenshot shows the 'Transition Inspector' window. The 'Transition Name' is 'service pending'. It is a 'Gated Transition'. The 'Required Role Alias' is set to '<None>'. There is a checkbox for 'Hand off flow run after this transition' which is currently unchecked. The 'Transition ROI Value' is 0.0. The 'Description' field contains the text '\${service} is pending'.

For information about working with the Transition Inspector, see ["Creating Transitions" on page 135](#).

## Bookmarks pane

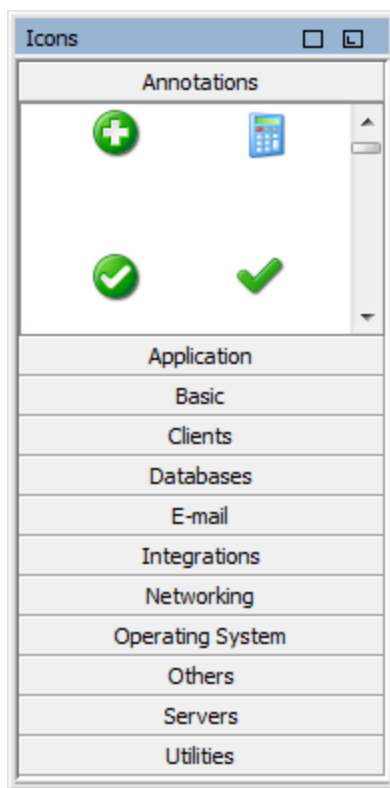
The **Bookmarks** pane, which you open with the **Bookmarks** tab in the upper-right of the Studio window, makes it easier to find and use the operations and flows that you use frequently.

You can add flows and operations to the **Bookmarks** pane by dragging them from the Library. For more information about bookmarks, see ["Bookmarking Flows and Operations" on page 252](#).



## Icons pane

The **Icons** pane, which you open with the **Icons** tab in the upper-right of the Studio window, contains libraries of icons that you can use to clarify what a step does. You can use one of these icons to replace the default icon on a flow or step.



For information about working with the **Icons** pane, see ["Modifying a Flow" on page 115](#).

## Flow Variables pane

The **Flow Variables** pane, which you open with the **Flow Variables** tab in the upper-right of the Studio window, lists the flow variables used in the flow and describes how they are used.

Name	#		
-destination	2		
Step Inputs Without User Prompts	2		
Assign From "destination" to Input "destination" in "Basic Notify"	-		
Assign Input "destination" to "destination" in "Basic Notify"	-		
-from	2		
Step Inputs Without User Prompts	2		
Assign From "from" to Input "from" in "Basic Notify"	-		
Assign Input "from" to "from" in "Basic Notify"	-		
-list	2		✓
Step Inputs With User Prompts	2		✓
Assign From "list" to Input "list" in "List Iterator"	-		✓
Assign Input "list" to "list" in "List Iterator"	-		✓

For information about working with the **Flow Variables** pane, see ["Working with Variables" on page 175](#).

## References pane

The **References** pane, which you open with the **References** tab on the lower edge of the Studio window, shows how an operation or flow is used in existing flows. The pane can display two kinds of references:

- **What uses this?** - Identifies flows that have a step created from the operation or flow.
- **What does this use?** - Identifies objects (selection lists, permissions, system filters) that the operation or flow makes use of. In the case of flows, this includes operations and subflows from which the flow's steps were created.

References from /My_Project/Content/Library/Health Check/Integrity Check for Table	
Object	Path
-Integrity Check for Table	/My_Project/Content/Library/Health Check/Integrity Check for Table
Operation: Error [return]	/My_Project/Content/Library/Operations/Error
Operation: Resolved [return]	/My_Project/Content/Library/Operations/Resolved

For information about working with the **References** pane, see ["Finding Out How Flows and Operations are Used" on page 243](#).

## Problems pane

The **Problems** pane, which you open with the **Problems** tab on the lower edge of the Studio

window, lets you check whether a selected flow or operation is valid. This pane displays problems with a selected flow or operation, with their locations and descriptions.

Problems				
Source Type	Name	Description	Location	
Step	Get Row Index by Condition	User Input: hasHeader references missing selection list	/My_Project/Content/Library/Get Cell by...	
Step	Is rowsCount Greater Than 0	Operation cannot be found	/My_Project/Content/Library/Get Cell by...	
Step	SQL Query	User Input: host references missing selection list	/My_Project/Content/Library/Health Ch...	
Transition	success	Transition source step has no operation linked to it	/My_Project/Content/Library/Get Cell by...	
Transition	failure	Transition source step has no operation linked to it	/My_Project/Content/Library/Get Cell by...	

For information about working with the **Problems** pane, see ["Validating Flows in the Problems Pane" on page 209](#).

## Search pane

The **Search** pane, which you open with the **Search** tab on the lower edge of the Studio window, enables you to search for a flow or operation. The Studio Search engine uses the Apache Lucene syntax.

Search				
Search:	<all fields>	for	SQL	<input checked="" type="checkbox"/> Exact
38 hits				
Rank	Name	Type	Path	Description
****	SQL Restart	web	/HPOO-MicrosoftSQLServer [10.00.00-713]/Co...	f2dcfd53-a114-4aad-91c4-ce55b...
***	SQL Manager Statistics	flow	/My_Project/Lib/Health Check	c90a9714-1aa2-4ed4-99a4-891c...
***	SQL Manager Statistics	flow	/My_Project/Content/Library/Health Check	c90a9714-1aa2-4ed4-99a4-891c...
***	Is a SQL Server	web	/HPOO-MicrosoftSQLServer [10.00.00-713]/Co...	28b273de-2dff-4e01-8545-4f3b...
***	Is An SQL Server	flow	/HPOO-MicrosoftSQLServer [10.00.00-713]/Co...	86262296-de06-437d-9544-f795...
***	SQL Event Log Diagnostic	flow	/HPOO-MicrosoftSQLServer [10.00.00-713]/Co...	a4ab3782-6a22-482e-9875-f006...
**	SQL Server Diagnostic	flow	/HPOO-MicrosoftSQLServer [10.00.00-713]/Co...	c8341530-8c94-48dd-84de-a5c7...

For information about working with the **Search** pane, see ["Finding a Flow or Operation" on page 235](#).

# Getting Started with HP OO Studio – Major Steps in the Workflow

This topic briefly describes the major steps involved in working with HP OO Studio. Click the links to see more detailed information about each step.



1. **Create a new project** - Create a project to contain the flows, operations, folders, and configuration items for a business purpose.

See ["Managing Projects" on page 46](#).

2. **Import a content pack** - Import any content packs that you need, so you will be able to copy

the relevant content into your project.

**Note:** The first two steps do not have to be performed in this order. It is possible to import a content pack before creating the project.

See ["Importing Content Packs to a Project" on page 68](#).

3. **Create a flow** - Put together the operations, inputs, transitions, responses, and return steps that make up your flow.

See ["Creating a Flow – Step-by-Step" on page 99](#) and ["Advanced Authoring" on page 185](#).

4. **Run and debug the flow** - Validate your flow in the Debugger.

See ["Testing and Debugging a Flow" on page 210](#).

5. **Release the content, packaged into a content pack** - Package your project into a content pack, containing the flows, operations, actions, and configuration items, in order to promote it to HP OO Central.

See ["Exporting a Content Pack" on page 225](#).

## Adjusting the Appearance of the HP OO Studio Window


You can set the panes in HP OO Studio to the following states:

- **Docked** - set in a permanent position in the Studio window
- **Floating** - free to be repositioned in the Studio window
- **Pinned** - hidden at the side of the Studio window so that only the tab is visible, and you have more room for your workspace

## What do you want to do?

### Float a pane


By floating a pane, you make it possible to move it to a different position in the Studio window.

1. Click the **Float** button  in the upper-right-hand corner of a docked pane.
2. Move the pane to a new position in the Studio window.


### Dock a pane

If a pane has been floated to a new position in the Studio window, docking returns it to its


permanent position in the Studio window.

Click the **Dock** button  in the upper-right-hand corner of the floating pane. The pane returns to its docked position.


### Maximize a pane

To maximize a pane, so that it expands to the size of the entire HP OO window, click the **Maximize** button .


### Restore a pane to its original size

To restore the pane to its size before you maximized it, click the **Restore** button .

### Pin a pane to the side of the Studio window

Click the **Pin** button  to pin the pane to the side of the Studio window so that only the tab is visible. You can display the pane by clicking the tab.

### Unpin a pane

After a pane has been pinned, click the **Pin** button  again to unpin it. Unpinning a pane returns it to its open, docked position in the Studio window.

### Adjust the size of a pane

Drag the edge of a pane to make it larger or smaller.

### Reset the Studio window to the default layout

To return the Studio window to the default layout, select **Window > Reset Window Layout**.

## Best Practices

The following practices are recommended, especially when multiple authors are creating flows.

General Best Practices .....	23
Best Practices for Sharing Content .....	24
Best Practices for Naming .....	26
Best Practices for Flows .....	28
Best Practices for Operations .....	29
Best Practices for Steps .....	31
Best Practices for Transitions .....	33
Best Practices for Inputs .....	34
Best Practices for Debugging .....	34
Best Practices for Configuring Studio .....	35
Best Practices for Descriptions .....	35
Best Practices for Source Control Management .....	42

## General Best Practices

### ***Folder Structure***

Make sure that the folder structure is well-defined and consistent between projects, so that other authors will be able to locate your flows, operations, and utilities.

### ***Rename Within Studio***

If you need to rename a project, flow, operation, or other HP OO entity, you should do so within Studio. Do not rename entities in a file browser, such as MS Explorer.

### ***Best Practices Document***

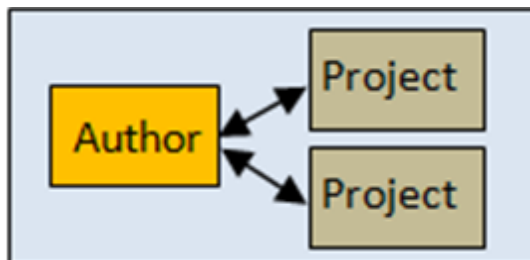
Create a document describing the naming conventions, folder structure, and other guidelines that you want flow authors to follow. For examples, see ["Best Practices for Naming" on page 26](#).

## Best Practices for Sharing Content

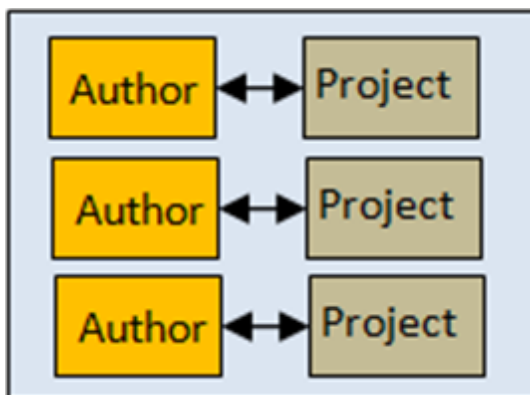
### *Identify your Environment*

There are multiple ways for authors to work together on projects. Before starting to work, you should think about how your authors will be working. For example:

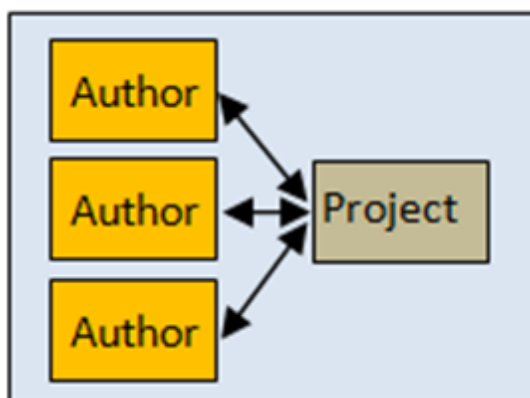
- One author, with one or more projects



- Multiple projects, each belonging to one author

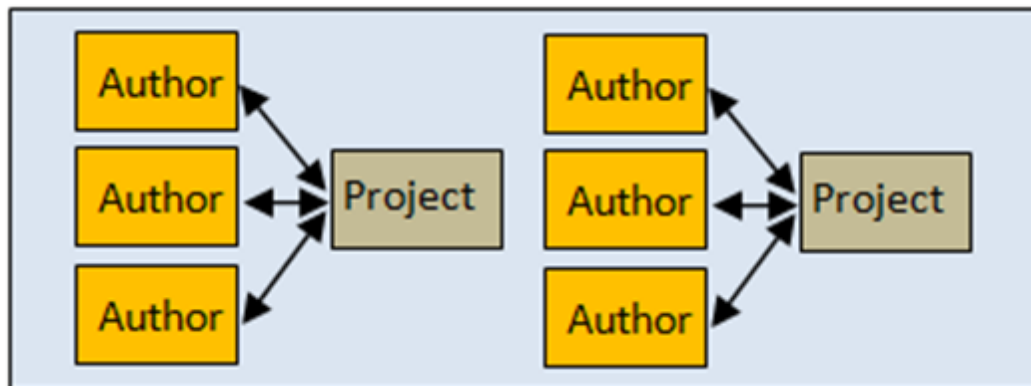


- Single project, multiple authors

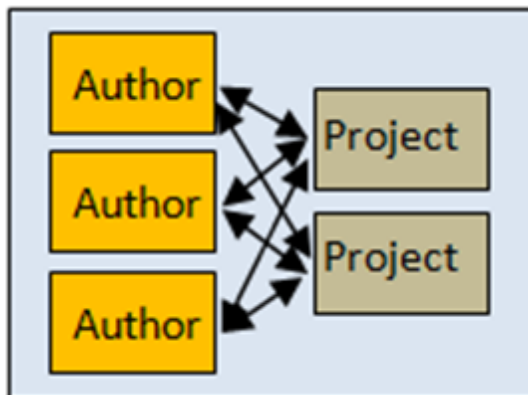




- Multiple teams, each with one project



- Multiple authors working in parallel on multiple projects



This document includes recommended best practices for the different types of environment.

## ***Community Interaction***

Community interaction involves sharing content within your organization and with other organizations.

The following are suggestions for community interaction, when authoring:

### **1. Download content from the community**

In addition to HP content and content packs that were developed by authors in your organization, it is possible to download content that was contributed to the community by other organizations. This content resides on HPLN.

For more information, see ["Importing Content Packs to a Project" on page 68](#).

### **2. Discuss issues with the community**

As part of the flow development, you can consult the community on issues such as problems in the flow, content recommendations, best practices, and so on.

For this, you should search existing community discussions for relevant keywords, create a new discussion, and track it.

In addition, you can contribute to the community knowledge via active involvement in discussions or through publishing best practice documents. For such documents, you may want to consider whether to make them open for editing or read only.

### 3. **Contribute content to the community**

When you have finished developing and validating the new content, and promoting it to the production environment, you may want to contribute it to the community as free or paid content. During the life span of that content in the community, you may decide to update it, remove it, or open it for other customers to update.

## **Best Practices for Naming**

### ***Flow Types***

- Classify different types of flows according to type, to make them easy to identify.
- Flow types should be stored in separate folders, to make them easy to locate.
- It may also be helpful to use specific icons for each flow type.

### ***Naming Conventions: Case***

Be consistent about case in the names of different types of objects. For example, use title case when naming the created flows and camel case for inputs, outputs, results, and flow variables.

**Note:** Camel case means first letter in lower-case, subsequent first letters of words contained in the name are upper-case, and no spaces within the name. For example, **serverName**.

Title case means first letter capitalized for all words, except helper words like 'a', 'the', 'and', 'by', 'for', 'to', 'from', and so on. For example, **Reboot a Server**.

### ***Naming Conventions for Flow Types: Prefixes***

Use naming conventions for different flow types. For example, add prefixes to flow names, based on the flow type:

- **User Interface** flows - UI
- **Infrastructure** flows - IF

- **Utility** flows - UT

## ***Naming Conventions for Variables: Prefixes***

Use naming conventions for different types of flow variable. For example, add prefixes to variable names, based on the variable type:

- Flow input - FI
- Step input - SI
- Operation input - OI
- Local variable - LV
- Global variable - GV

## ***Intuitive Names***

- Use self-explanatory names for flows, so that they describe the purpose of the flow.
- Rename steps if this will enhance the clarity of the flow. For example, a step named **String Comparator** is less intuitive than **Validate <inputName>**.
- If you are renaming operations and steps, make sure that the name clearly describes the purpose of the operation or step.
- Rename transitions if this will enhance the clarity of the flow.
- For flows and operations that run a single task, use a "<Verb> <Noun>" name format. For example, **Send Mail**, **Create Snapshot**.
- For sample flows, use the word "Sample" in the name. For example, **Send Mail Sample**, **Create Snapshot Sample**.
- For flows that check whether something is the case, use the question being answered as the name. For example, **Is Computer Account Enabled**.
- For health check flows that collect information about a system or environment, include "Health check" in the flow name (except for cases where you have a special **Health Check** folder). For example, **Solaris Health Check**.

## ***Word Use***

Some common input names occur across many operations and steps. Note that the following input names are used in HP OO content:

- **host** – For Windows, the host is the machine on which the operation works (for example, the host from which you are getting a performance counter or on which you are restarting a service). For secure shell (SSH) operations, the host is the machine on which the command is running.
- **username** – The name of the account to use for logging on to the machine.
- **password** – The password to use to log on to the machine.

Other common input names include:

- **mailHost** - The host machine from which an email is sent.
- **target** – When the host affects another system, the system that is affected by the host should be called the target. For example, if you SSH to server1 to run a ping against server2, then the host is server1 and the target is server2.

## Best Practices for Flows

### *Plan the Flow*

Plan the structure of the flow you want to create, before starting to build it.

### *Keep Flows Simple and on One Screen*

A flow should fit on the canvas on a 1024 x 768 screen with Studio maximized and a 1:1 view magnification. Larger flows are not strictly prohibited, but if a flow is larger, examine it carefully to see whether you can break down some of its sequences of steps into subflows.

### *Reuse Flows*

Plan your flows for reuse. Create a bank of simple flows that can be reused as substeps in more complex flows.

### *Check Where a Flow/Operation is Used Before Modifying*

Before making changes to a flow or operation, use **References > What uses this?** to check whether other flows use it.

### *Copy a Flow Before You Modify*

Always make a copy of a flow before modifying it. Even if you don't need both flows and the original is not being used by anything else, keep the original as backup, just in case the modifications are not successful. After you have completed the copied flow, you can delete the original.

## ***Be Consistent***

Design different flows using the same start and end locations.

## **Best Practices for Operations**

### ***Be Careful When Modifying Operations***

When you modify the properties of an operation (in the **Properties** sheet), remember that this will affect all the flows that use this operation as a step, including steps that were created earlier from this operation. Changing an operation's properties can break other flows that use it. It is recommended to create a copy of the operation and modify the copy. If the changes are for a single use, modify the step rather than the operation.

### ***Deep Copy if You May Need to Modify Operations***

If you are copying a flow and you think that you may need to modify the properties of the operations, it is best to use the **Copy Deep** command. This also makes a copy of the operations, so that you can modify them without affecting the originals.

### ***Create a Folder for Deep Copies***

If you are planning to copy a flow using the **Copy Deep** command, it is recommended to create a new folder for the flow and its operations.

### ***Use Original Operations When Not Customizing***

If you don't need to customize operations, use the original versions rather than copying them. Avoid cluttering your folders with unnecessary copies of operations.

### ***Keep Original Names and Messages for Integrations***

Integrations may come with their own rules and best practices. When working with integrations, keep the operations as similar as possible to the product they are integrated with:

- Keep the original names from the API being used, for flows, operation, inputs, and so on.
- Do not change the Error/Info/Success messages that come from the product you integrate with.
- Always mention the API version used for the operation at folder level. If possible, provide the location of the API as well.

## ***Copy Steps Rather than Sealed Operations***

Do not make copies of sealed operations, such as those in the **Operations** folder. Instead, make changes to the steps that you have created from sealed operations.

## ***Assign Values to Flow Variables***

By default, operations should use and set flow variables for inputs that are used repeatedly in a particular flow. For example, multiple operations in a flow might need the host, user name, and password inputs to get information from a server or the port of a mail server. Assigning those values to flow variables that are used in the various steps that require such data simplifies maintenance of the flow and makes it easier to adapt to different situations.

In contrast, the subject line of an email is probably different for each step that requires an email subject line. Therefore, the subject line is probably not a good candidate for being provided from a flow variable.

## ***Avoid Multiple Operations that Run the Same Command***

Avoid creating multiple operations that run the same command. For example, you can get both packet loss and maximum latency from a ping operation. Rather than create multiple operations that use the ping command, a better practice is to capture both pieces of information in one step by using multiple outputs of one ping operation.

Exceptions to this principle are operations that are extremely generic, such as an operation that runs a WMI command. It is better to create WMI command operations that are specific to particular functions, instead of a single operation that has a very generic input for the WMI command and very generic outputs.

## ***Use Result Filters Rather than Scriptlets***

For capturing data from the output stream of a command, using result filters is better than using a scriptlet. There are several reasons:

- Result filters are accessible and immediately visible on the Results tab editor rather than residing separately, as scriptlets do on the Scriptlets tab.
- Scriptlets are more difficult for non-programmers to maintain.
- If one of the operation's results is removed, the result filters are automatically invalidated. Any scriptlets that the author fails to remove after deleting the result that the scriptlet manipulates remain and can cause bugs in the flow.
- If you need a scriptlet for the desired processing of the result data, you can use a scriptlet filter.

## ***Only Use the Responses You Need***

Most operations should have only two responses—success and failure. Using a small number of responses makes flows easier to create and understand. Multiple responses based on different types of failures should only be used when there are obvious distinct paths to follow or there are circumstances where an outcome may only be a failure because of the situation (such as a redirect response to an HTTP Get).

However, don't force this principle when it doesn't make sense. For example, an operation that gets data and checks a threshold may require three responses (none of which is a success response)—failure, over threshold, and under threshold.

## ***Default Response is Failure***

The default response for an operation should be failure. This way, an incomplete operation shows as a failure during flow debugging and points the author to the problem before the flow goes into production.

## **Best Practices for Steps**

### ***No Description***

Steps do not generally require descriptions, because the transition description of the step's response tells what happened in the step.

### ***Callouts***

Consider using callouts to provide information about a step. Callouts can greatly enhance the usability of a flow.

### ***Start Step***

The start step should be located in the upper-left corner of the flow, with the following exceptions:

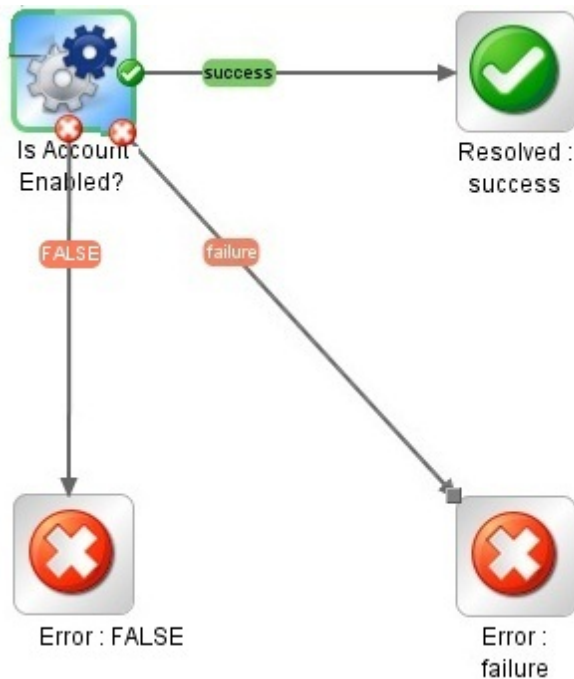
- The start step has many responses, each of which leads to another step.
- Placing the start step in the upper-left corner would cause excessive visual complexity, such as the crossing of transitions.

### ***Rename Return Steps***

If you have multiple return steps of the same type in a flow, rename the return steps to include the cause. For example, **Error: failure** and **Error: threshold not met**.

## ***Distinguish Between Failure and Negative Result***

Avoid confusing a failed operation with a negative result. For example, if an operation asks a question for which the answer may be TRUE or FALSE, a FALSE answer is not the same as a failure of the operation. In such a case, you need two **Error** return results, one for a FALSE result and one for a failure of the operation.



## ***Distinguish Between No Action Taken and Successful Gathering of Information***

Avoid confusing the following:

- The **No Action Taken** 🟡 return step is used when a remediation flow gathers data but cannot determine any diagnosis or remediation.
- A flow or operation that is intended solely to gather data should return **Resolved** ✅ when it is complete, rather than **No Action Taken** 🟡.

## ***Use Results to Assign Values to Flow Variables***

To assign information to a flow variable, use the step's **Results** tab. Filters on the results greatly enhance your flexibility in obtaining data from step results.



## ***Be Careful with Flow Variables***

Be aware that flow variables are accessible to the entire flow. Pay attention to flow variable manipulation, because data can accidentally be altered in one step and then used incorrectly in the flow's subsequent steps.

## ***Don't Add Unnecessary Results***

The operation or flow that a step is based on may provide several outputs. But when adding step results, make sure to only use the outputs that you need in the flow. Too many results may affect performance, by slowing down the flow with unnecessary data.

## ***Create Results that Capture Error Code***

If a step or transition needs the exact error that came back from an operation, create a step result that captures the error code, and assign the error code to a flow variable.

# **Best Practices for Transitions**

## ***Keep Transitions Neat and Tidy***

- As much as possible, transition lines should not cross.
- Use straight transitions, if possible. You should only use curved transitions when this is necessary for the flow layout.
- When possible, position steps so that transitions are horizontal, vertical, or at a 45-degree diagonal.
- Collapse multiple transitions from one step to another so that a single line represents all of the transitions.
- Position transition labels so that they do not overlap the step labels or each other.
- Place transition labels toward the outside of the flow when possible. For example, if two steps are at the top of the flow canvas, the transition labels should be above the transition lines. If the steps are at the bottom of the canvas the labels should be below the transition lines.

## ***Use the Grid***

Activate the grid, to keep your steps aligned.

## Best Practices for Inputs

### *Delete Unnecessary Inputs*

Delete optional inputs in steps, if these are not required.

### *Do Not Disable Required Inputs in an Operation*

Do not disable a required input in an operation, because it will disable it in all instances of that operation; you should modify the individual step.

### *Add Inputs According to Ordering Rules*

Add inputs in a consistent order, according to ordering rules. For example:

- By intuitive or logical grouping
- By importance (required inputs first)
- In alphabetic order

**Note:** Input that uses another input must be specified in the correct order. For example, if you want the following inputs:

```
input_a="The first input"
```

```
input_b=$(input_a)+
```

You must define **input\_a** before **input\_b**.

### *Assign Data to Flow Inputs*

Ideally, input values used by flow steps are supplied by flow inputs and passed to the steps by flow variables.

In general, flow authors should assume that a user will begin a flow and then start another task while the flow is running. Assigning as much data as possible to flow inputs also simplifies making changes to the flow.

## Best Practices for Debugging

### *Debugging Subflows*

It is best practice to debug subflows before debugging their parent flows.

## Best Practices for Configuring Studio

### *System Properties*

Be cautious about creating system properties or changing their values, because they have global scope, becoming part of any flow run's context when the run is begun. As a result, changing a system property's value can break existing operations and flows.

### *Configuration Items*

Before you delete a configuration item (system filter, scriptlet, selection list, and so on), it is recommended to use the **What Uses This** function to check that other items do not depend on it. For more information, see ["Finding Out How Flows and Operations are Used" on page 243](#).

## Best Practices for Descriptions

### *All Descriptions*

- Use the **Description** tab to enter a detailed description of the purpose of the operation, step, or flow, so that other authors will know what it is supposed to do and why you designed it this way.
- The description should include search words to help you or others find the item, and should describe all inputs, responses, and outputs.
- For longer descriptions, separate each paragraph by a single line.
- Do not add spacing at the start of the paragraph.
- Sentences should be short and concise.
- Avoid using long phrases. Instead, split them into separate sentences.
- All sentences must start with a capital letter and end with a period.
- Use present tense and active voice. For example, instead of "If the input is not specified, it will default to 1534", write "If you do not specify a value for this input, it defaults to 1534".

### *Folder Descriptions*

If you create multiple flows or operations that interact with the same technology, group them into a single folder and provide this information in the folder's **Description** area. To access the Description area, right-click on the folder and select **Properties**.

## Flow Descriptions

- Include any special requirements or changes that are necessary for the flow to run automatically (for example, on a schedule).
- Include any limitations to the flow's usage, such as:

```
This flow only works:  
- On Windows 2003 or later  
- If the Windows Telnet Service is enabled
```

- Specify which of the flow's inputs are required, and include information about where authors can find the data that the inputs require and the required format for the data.
- Include the flow's responses, including the meaning of each response.
- Include the result fields, including a description of the data supplied in each result field.
- Include any additional implementation notes, such as:
  - Supported platforms or applications, including version information.
  - Application or Web service APIs that the flow interacts with.
- A flow that performs triage, diagnosis, or remediation should first verify that a problem exists.
- A flow that sends a notification to the user should use notification subflows, which enable the flow author to choose from several means of notifying the user. For example, the **Web site Health Check** flow uses the **Notify** subflow. Once the user configures the **Web site Health Check** flow to their email and ticketing systems, all flows that use this flow will send notifications correctly.
- When creating a subflow, always start by adding a description. This simplifies implementation, as explaining the purpose and inputs/outputs of a flow leads to a cleaner and clearer flow design.

You should annotate (supply a description for) all transitions in a top-level parent flow. These transition descriptions should describe what happened in the step that preceded the transition.

## Operation Descriptions

- Include a description of what the operation does.
- List inputs that the operation requires, including where authors can find the data that the inputs require and the required format for the data.
- Include responses, including the meaning of each response.

- Include result fields, including a description of the data supplied in each result field.
- Include any additional implementation notes, such as:
  - Supported platforms or applications, including version information.
  - Application or Web service APIs that the flow interacts with.
  - Other environmental or usage requirements.
- Use the following template as the basis for your operation descriptions:

```
Description of what the operation does.  
  
Inputs:  
  
Input1 - info about this input  
Input2 - info about this input  
Input3 - info about this input  
  
Responses:  
  
Response1 - info about this response  
Response2 - info about this response  
  
Result:  
  
The primary result of the flow/operation  
  
Extra Results:  
  
Result1 - The first additional result  
Result2 - The second additional result
```

## ***Input Descriptions***

- Include meaningful descriptions of required inputs.
- Include sample data needed for inputs, in correct format.
- Differentiate between optional and required inputs.
- Make sure that the order in which inputs are listed in the description is the same as the order in which they appear in the **Inputs** tab.
- Describe the necessary syntax, if any is required, of the data that the input takes.
- Use the same syntax for section headings everywhere in the description. (For example, use

“Examples” not “Example” or “ex:”).

- The formats for section headings are: "Value format", "Examples", "Default values", and "Valid values". These are required whenever the information is applicable and available.
  - The values from "Value format", "Examples", "Default values", and "Valid values" that contain only a punctuation mark should be between apostrophes (for example, ',', '.').
- Leave four spaces before each input name.
  - Use the following template as the basis for your input descriptions:

```
Inputs:
    inputName - Name of the first input

Value format: domain\user
Valid values: us.east.1a, us.east.1b
Default value: us.east.1b, ','
Examples: valueString, 31241423, one, two, three
    secondInputName - Name of the second input

Value format: text
Valid values: one, two, three
Default value: one
Examples:
```

## ***Transition Descriptions***

- Supply a description for all transitions on a top-level parent flow. These transition descriptions should describe what happened in the step that preceded the transition.
- You need not add descriptions of transitions in a subflow unless the data is critical to see during a run.

## ***Output Descriptions***

- Make sure that the order in which outputs are listed in the description is the same as the order in which they appear in the **Outputs** tab.
- Leave four spaces before each output name.

- Include any environmental limitations of the operation that limit the circumstances in which it can run.

## ***Result Descriptions***

- The primary output must be the first result in the **Results** list and should contain the following text: "This is the primary output".
- List enumerations and table columns on separate lines for better visibility.
- Leave four spaces before each result name.
- Use the following template as the basis for your result descriptions:

```
Results:

    returnResult - This is the primary output.

    firstResult - Your EC2 instances in a table having the following columns:
(The following is
an example of a result in a table format.)

Instance Id
AMI ID Machine type

Status - The possible values are: "Public DNS", "Key pair name", and "Ramdisk
ID",
```

## ***Response Descriptions***

- In the description, use the word: "Responses".
- For "success" and "failure", which are the most frequently used responses, the following phrases are recommended:
  - Success - "The operation completed as stated in the description" instead of "The operation completed successfully".
  - Failure - "The operation completed unsuccessfully. See the Notes for troubleshooting help." instead of "Something went wrong".
- Leave four spaces before each response name.
- Use the following template as the basis for your response descriptions:

```
Responses:
```

```
success - The operation completed as stated in the description.  
failure - The operation completed unsuccessfully. See the Notes for troubleshooting help.  
no more values -
```

## ***Notes Descriptions***

- The "Notes" section can be left empty.
- For more complex operations, which need to contain more information, organize the "Notes" section into subsections so that it is easy to read and understand.
- Use the "Prerequisites" subsection for all configurations, settings, environment setups, and so on that are mandatory for the operation to work successfully.
- Use the "Extra settings" subsection to list all of the special use cases of the operation or flow such as security and networking settings. This section is useful for operations that have special use cases such as multiple settings.
- Use the "Troubleshooting" subsection to describe errors that do not have self-explanatory messages and to provide possible fixes.
- Use the "Other" subsection to include information that does not fit in the above sections.
- Leave the order of the subsections in the "Notes" section as they are shown.
- These sections are optional. If you only need to include the "Other" subsection heading, do not include the subsection heading but keep numbering the issues. This means that the heading remains "Notes" and the "Other" section is included under "Notes".
- If the "Notes" section applies to multiple operations or flows, put the notes at the folder level. In the individual operation or flow, refer to the notes at the folder level. For example, "See the Notes section in the folder FolderName".
- Use the following template as the basis for your notes descriptions:

```
Prerequisites:  
1. The first prerequisite  
The first prerequisite description  
    1.1. Do the first step. (4 spaces)  
    1.2. Enter one of the following comments:(4 spaces)  
        - first comment. (11 spaces)
```



```
- second comment. (11 spaces)

2. The second prerequisite
The second prerequisite description
Extra settings:
1. abc
2. def

Troubleshooting:
1. abc
2. def

Other:
1. abc
2. def
```

Example of a complete description of the **Get Database Availability** group:

Obtains the list of servers that are members of a database availability group (DAG). You can also use it to view real-time status information about a DAG, such as: PrimaryActiveManager, OperationalServers, ReplicationPort, NetworkNames, StartedMailboxServers, StoppedMailboxServers., etc.

Inputs:

host - The Exchange 2010 server host.

username - The username to use when connecting to the server.

password - The password to use when connecting to the server.

authType - Specifies the mechanism that is used to authenticate the user's credentials.

Valid values: Default, Basic, Credssp, Digest, Kerberos, Negotiate, and NegotiateWithImplicitCredential.

Default value: Default.

dagName - The name of the DAG.

Examples: oodag.

delimiter - The delimiter used in the result for separating the properties.

Default value: ', '.

#### Results:

`returnResult` - This is the primary output. Returns a list of DAG properties. Each property is on a new line and the property name is separated from its value using the delimiter.

`servers` - A list of servers that are members of the DAG.

`operationalServers` - A list with the operational servers from the DAG.

`primaryActiveManager` - Represents the node which owns the cluster core resources group.

`distinguishedName` - The DAG's DN from active directory.

`isValid` - Whether the DAG is valid or not.

#### Responses:

`success` - A list of DAG properties was retrieved from the exchange server.

`failure` - Failed to obtain DAG information. Unable to connect to the server (maybe wrong credentials or unsupported authType).

The cluster is not available and the cluster service is not running.

#### Notes:

1. For information related to Powershell remote connection, consult the description of the "Exchange 2010" folder.

2. Supported version: 2010.

## Best Practices for Source Control Management

- To avoid conflicts with other authors, always lock your files before working on them. Make sure to lock the file first and then edit it, rather than starting to work and then locking the file before committing it. Someone else may have started editing the file, while you were working on it. Note that some source control management tools don't indicate that a file is locked, until you try to lock it.
- Make sure you update an item before locking, modifying and committing.
- When a folder is locked, this also locks the files within it.

- After you modify one of the following items, It is recommended that you commit the entire project, even though you can also commit part of the tree.
  - Flow, operation, or configuration item description
  - Input prompt message (if Prompt is selected)
  - Step prompt message (in the Display tab)
  - @Actions and IActions are not stored in the project. They are stored in the author's Studio repository and linked to, from the operations that use them. To use @Actions or IActions that were created by another author, you will need to manually copy the Jar/DLL files to your Studio repository. If you have created @Actions and IActions, it is recommended to create a content pack containing the plugins, for other authors to import.

## Working with Different Languages in HP OO Studio

### - Localization

Localization refers to the adaptation of software for non-native environments. HP Operations Orchestration Studio can be localized into the following languages:

- fr - French
- de - German
- ja - Japanese
- es - Spanish
- zh - Chinese

The following Studio text strings are localized:

- Flow descriptions and callouts
- Transition descriptions
- Configuration item descriptions
- Step descriptions and prompt texts (from the Display tab)
- Input prompts
- Folder descriptions

## Changing the Current Studio Locale

By default, Studio is installed in the default language of the computer.

You can override the current locale by changing its configuration:

1. Open the **Studio\conf\studio.properties** file.
2. Search for the following lines:
  - `user.language=`
  - `user.region=`
3. Add the language locale to these fields. The following codes are valid:

Language locale code	Region (optional)	Language
fr		French
de		German
ja		Japanese
es		Spanish
zh	CN	Regional Chinese

**Note:** In order to implement a different language interface, you must assign the same valid language locale to both the **user.language** field and **user.region** fields (where relevant). If an invalid value is assigned, or these fields are left blank, Studio uses the default system locale.

## Studio Display Language

Studio takes the display language from the `user.language` property, this property must be valid (`user.region` must be valid as well), if not valid studio takes the system locale.

Currently Studio support the following display languages:

Language	Language code
French	en
German	de
Japanese	ja
Spanish	es
Regional Chinese	zh

In addition, the content packs can also be localized. For details, see ["Content Pack Localization" on the facing page](#)

## Content Pack Localization

All text strings contained in content pack flows and operations should be shown in the current Studio locale.

The localization files are saved with the name **cp\_<locale>.{<region>}.properties**.

(The region parameter is optional.)

For example, the default content pack properties file is named **cp.properties**, and the Chinese file is named **cp\_zh\_CN.properties**.

## Projects Localization

Projects have only one properties file which is saved as the default file (cp.properties) during the packaging of the project into a content pack.

This has two implications:

- Project texts are not really localized, they are always loaded/written from/to the same **cp.properties** file.
- If a content pack is unpacked, and loaded as a project in Studio, the localized properties file will not be used, only the default file. If one does not exist, it will be created and all text strings will be empty.

**Note:** When working in different languages, it is recommended to create a separate project for each language.

## Creating a New Content Pack

When creating a content pack from a project in HP OO Studio, the resource bundle folder contains only the cp.properties file.

**Note:** After changing the configuration of a content pack, you must restart Studio.

When the project is created/saved, the **cp.properties** file is saved in the project file system location under **\resource-bundles\cp.properties**.

You can also add your own resource files to the resource-bundles directory. These files will be bundled along with the other resource files into the content pack's jar file.

# Working with Projects

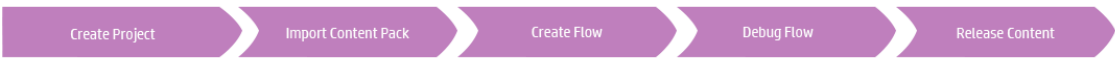
A project is a logical unit that can include flows, operations, folders, selection lists, and other items for a business purpose. A project may be developed by a single author or as a collaboration project.

Projects in Studio use a file directory structure, like that of an operating system, which enables easy integration with a source control application.

Managing Projects .....46

Managing Folders in the Project Pane .....52

## Managing Projects



Before you can start creating flows in HP OO Studio, you must be working in a project.

**Note:** Creating the project does not have to be the first step in the workflow. It is possible to import content packs before creating the project.

In Studio 10.00, projects are stored locally on the author’s file system. In order to have source control capabilities, you should use an external source control management tool. For more information about source control management, see [Working with Source Control](#).

## What do you want to do?

### Configure a workspace

A workspace is a directory that you create to house your projects. When you create projects, you store them in the **Workspace** directory.

This step is optional. If you do not configure a **Workspace** folder , Studio will create one in a default location in your user folder, when you create a project.

1. If you are working with a source control management tool, create a **Workspace** folder in your shared repository. For more information, see ["Managing Projects using a Source Control Tool" on page 1](#).
2. Open the file `Users\[USER_NAME]\.oo\10.00\opstudiorc.xml` in a text editor.
3. Where the text shows `<entry key="workspace" value="C:\Users\[USER_NAME]\.oo\Workspace">`, replace the value with the path where you want to locate your **Workspace** folder.

This step tells Studio to use this location by default when you create new projects.

4. Open Studio at least once, to create the **Workspace** folder.
5. If you are working with a source control management tool, check out the shared **Workspace** folder to your local **Workspace** folder. For more information, see ["Managing Projects using a Source Control Tool" on page 1](#).

## Create a project

1. Select **File > New Project**.

**Note:** Alternatively, you can click the **New Project**  button in the **Projects** pane.

2. In the **Name** box, enter a name for the project.
3. In the Create Project dialog box, verify that the path in the **Location** box is the correct path to the **Workspace** folder that you set up in the previous task. If the path does not lead to the **Workspace** folder, browse to and select the **Workspace** folder.
4. Click **OK**.
5. If you are working with a source control management tool, add the project folder to your shared repository. For more information, see ["Managing Projects using a Source Control Tool" on page 1](#).

## Import a project

To open a project that was created in a different location, you need to import it to Studio. Once a project has been imported, it appears in the **Project** pane.

If you are working with a source control tool, you may need to check out the project folder, so that you have a local working copy, and then import the project to Studio. For more information, see ["Managing Projects using a Source Control Tool" on page 1](#).

1. Select **File > Import Project**.


**Note:** Alternatively, you can click the **Import Project**  button in the **Projects** pane.

2. In the Select Project Directory dialog box, browse to locate the project that you want to import.
3. Click **OK**. The project appears in the **Projects** pane in Studio.

## Close an open project

If you close a project, it is visible in the **Projects** pane, but is grayed out and not available.

1. Select the project that you want to close.
2. Select **File > Close Project**.

**Note:** Alternatively, you can click the **Close**  button in the **Projects** pane.

## Open a closed project

After a project has been closed, you can open it, to work with it again.

1. Select the closed (grayed out) project that you want to open.
2. Select **File > Open Project**.

**Note:** Alternatively, you can click the **Open**  button in the **Projects** pane.

## Delete a project

Deleting a project is different from closing it, in that a deleted project is permanently removed from the workspace.

**Note:** When you delete a project, it is deleted from the workspace, but is not deleted from the file system. If required, you can import it again.

1. Select the project that you want to delete.
2. Select **File > Delete Project**.

**Note:** Alternatively, you can click the **Delete**  button in the **Projects** pane.

3. Click **Yes** in the confirmation dialog box.

## Copy flows and operations from a content pack into a project

The flows and operations in the **Content Packs** pane are read-only. You can create editable copies of these flows and operations by copying them into the project.

1. In the **Content Packs** pane, select the flow or operation that you want to copy.
2. Select **Edit > Copy**.
3. Select the location in the project tree where you want to paste the copy, and select **Edit > Paste**. The flow or operation is treated as a new object and is detached from the content pack it arrived with.

**Note:** If you also want copy the operations that make up a flow into the project, select **Edit > Copy Deep**. For more information, see ["Copying Flows and Operations" on page 240](#).



## Display the project properties

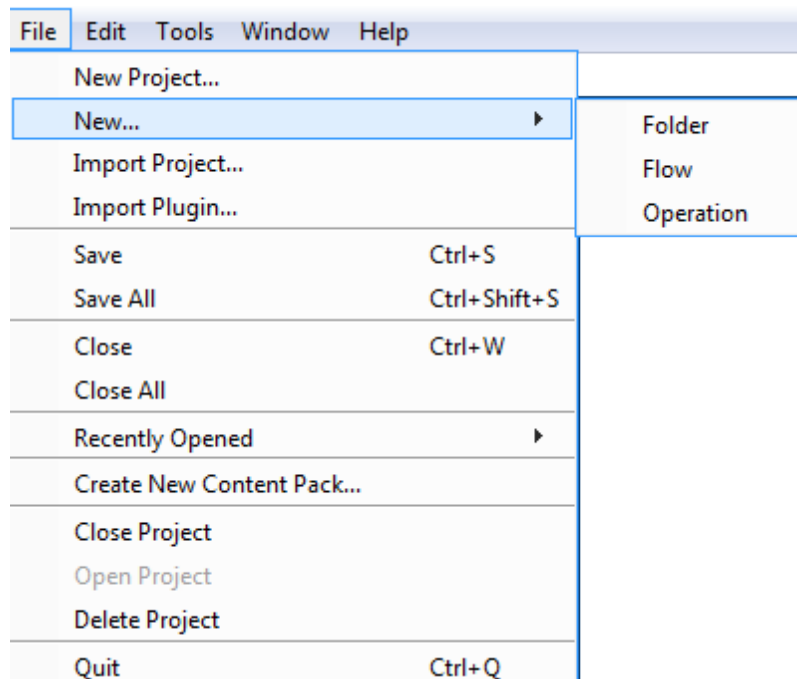
Right-click the project and select **Properties**. The Properties window displays information about the project.

## Display the properties of an object in a project

- Double-click a flow, operation or other object in the **Projects** pane. The Properties window for the object opens.
- If a flow is open on the authoring canvas, click the **Properties** tab in the lower left corner of the authoring canvas, to display the Properties window for the flow.

# Reference Material

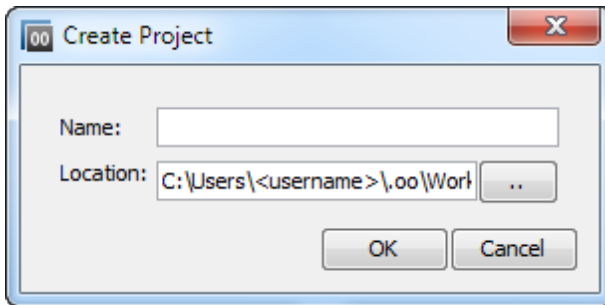
## File menu




Menu item	Description
<b>New Project</b>	Creates a new project.
<b>New</b>	Creates a new file, folder, or operation.
<b>Import Project</b>	Browse to and import an existing project from a different workspace.
<b>Import Plugin</b>	Browse to and import an action plugin, to use as the basis of a new operation. See <a href="#">"Creating Operations" on page 227</a> .

<b>Save</b>	Saves the selected element.
<b>Save All</b>	Saves all updated elements.
<b>Close</b>	Closes the currently selected flow or Properties window.
<b>Close All</b>	Closes all open flows or Properties windows.
<b>Recently Opened</b>	Lists recently opened items. This includes flows and operations that were open in the authoring canvas and Properties windows that were recently viewed.
<b>Create New Content Pack</b>	Creates a content pack from the selected project.
<b>Close Project</b>	Closes the currently selected project, so that it is grayed out.
<b>Open Project</b>	Opens the currently selected closed project.
<b>Delete Project</b>	Permanently deletes the selected project from the workspace.
<b>Quit</b>	Quits HP OO.

### Create Project dialog box

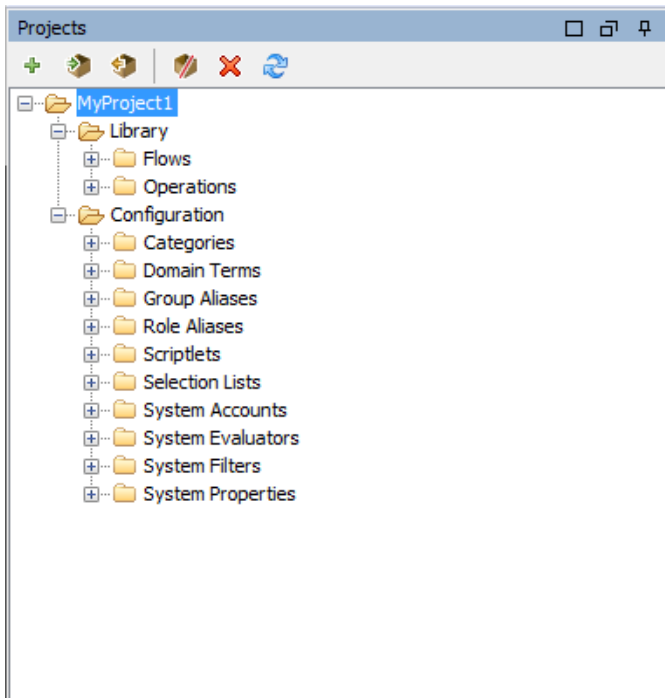


GUI item	Description
<b>Name</b>	Type a name for the new project.
<b>Location</b>	<p>Enter the location of the workspace, in which to store the new project.</p> <ul style="list-style-type: none"> <li>Type the path to the location.</li> <li>Click the browse button  to browse for the location.</li> </ul>

### Projects pane

The **Projects** pane contains the project tree—a hierarchical folder structure containing the editable content of your project:

- The **Library** folder, which holds flows and operations.
- The **Configuration** folder, which holds other HP OO objects (filters, scriptlets, system properties, and so on) that you can use to process operation results, create reports, and facilitate the running of flows.



GUI item	Description
<b>New Project</b>	Creates a new project.
<b>Import Project</b>	Browse to and import an existing project from a different workspace.
<b>Create Content Pack</b>	Creates a content pack from the selected project.
<b>Delete</b>	Permanently deletes the selected project from the workspace.
<b>Open</b>	Opens the currently selected closed project.
<b>Close</b>	Closes the currently selected project, so that it is grayed out.
<b>Refresh</b>	Refreshes the files in the currently select project.

## Managing Folders in the Project Pane

In the **Projects** pane, you manage the folders in the project—adding, deleting, copying, and renaming folders.

### ***Best Practices***

Make sure that the folder structure is well-defined and consistent between projects, so that other authors will be able to locate your flows, operations, and utilities.

If you are planning to copy a flow using the **Copy Deep** command, it is recommended to create a new folder for the flow and its operations.

We recommend using a document that describes the folder structure and other guidelines for flow authors.

## What do you want to do?

### **Create a folder**

1. Select **File > New > Folder**.
2. Type a name for the new folder and click **OK**.

**Note:** Names can be a maximum of 128 characters long, and are not case-sensitive.

### **Rename a folder**

Right-click a folder in the project tree, and select **Rename**.

### **Delete a folder**

Right-click a folder in the project tree, and select **Delete**.

### **Copy and paste a folder**

1. Right-click a folder in the project tree, and select **Edit > Copy**.
2. Right-click the position in the project tree where you want to paste the folder, and select **Edit > Paste**.

### **Drill down to subfolders**

Click the expand button  to expand a folder and display the subfolders inside it.

### **Display the folder descriptions**

Right-click a folder in the project tree, and select **Properties**. The properties of that folder are displayed.

# Working With Source Control

## What is Source Control in HP OO Studio?

HP OO Studio includes a built-in Source Control Management tool, that allows authors to work in their local environment and then synchronize the changes with the public version.

### Working Copy

The SCM repository holds all versioned data on a source control server. The Source Control Management tool in Studio manages local copies of the versioned data, known as the working copy. SCM accesses its repository across networks. Multiple users can access the repository at the same time.

### Checkout

Checkout is used to download sources from the repository to the working copy. If you want to access files from the source control server, checkout is the first operation you should perform. When you check out, a working copy is created, allowing you to edit, delete, or add contents. You can check out a file, directory, trunk or whole project. To check out, you need the source control server URL of the components you want to check out.

### Commit: Save changes to the repository

When you make changes to your local working copy, they are not automatically saved in the source control server. To make the changes permanent, commit the changes.

### Add: Add a new file to SVN repository

The **Add** command allows you to add new files or directories to the repository. The repository shows the newly added file after you commit the changes.

### Delete: Removing a file from repository

The **Delete** command deletes an item from the working copy (or repository). Files are deleted from the repository after you commit the changes.

### Move: Rename file or directory

The **Move** command moves a file from one directory to another or renames a file. The file is moved on your local sandbox immediately, as well as on the repository after committing.

### Update: Update the working copy

The **Update** command transfers changes from the repository into your working copy. It is recommended that you update your working copy before you start working, so that all the latest changes available in repository are available in your working copy.

## Creating an Initial Source Control Repository

Before you begin working with source control in Studio, you need to setup an initial repository.

**Note:** If your company already has a source control repository, this section is not relevant.

To create an SVN repository:

1. Create a shared folder in Windows, and assign it with read and write permissions, for the authors that will work on the repository.
2. Open command line (cmd) in <studio\_installation\_folder>\studio.
3. In the command window, type: **createFileSystemScmRepository.bat** and the full path to the shared folder.

For example:

```
c:\<studio_installation_folder>\Studio>  
createFileSystemScmRepository.bat SHARED_FOLDER_FULL_PATH
```

The following appears:

```
Repository was successfully created at <SHARED_FOLDER_FULL_PATH>
```

The shared repository folder is now ready for use with Studio.

## Working with Source Control in HP OO Studio

This section describes the common tasks that are used with the Source Control Management tool and more advanced tasks that the author may encounter when projects and items are shared with multiple authors.

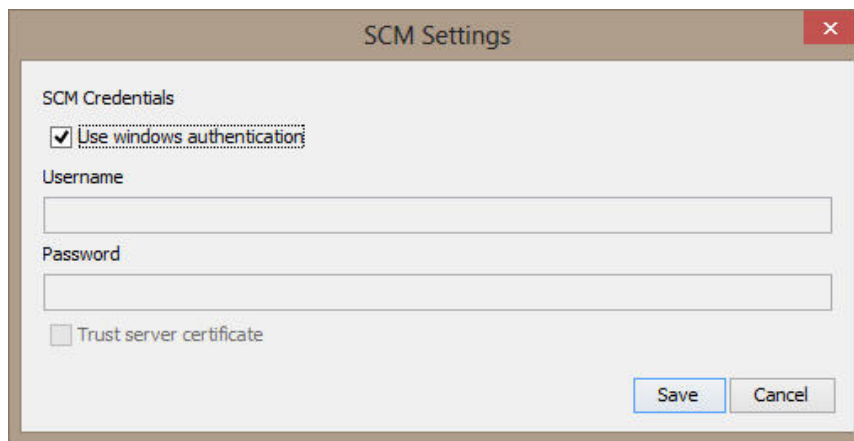
### Working with SCM in Studio

After you have installed Studio with a local repository, you are ready to start authoring in Studio and use the source control in your local environment. In the following sections, you can learn about common tasks that you perform with SCM.

#### Authentication Settings with Source Control Server

The first step in working with SCM is to set the user authentication with the source control server.

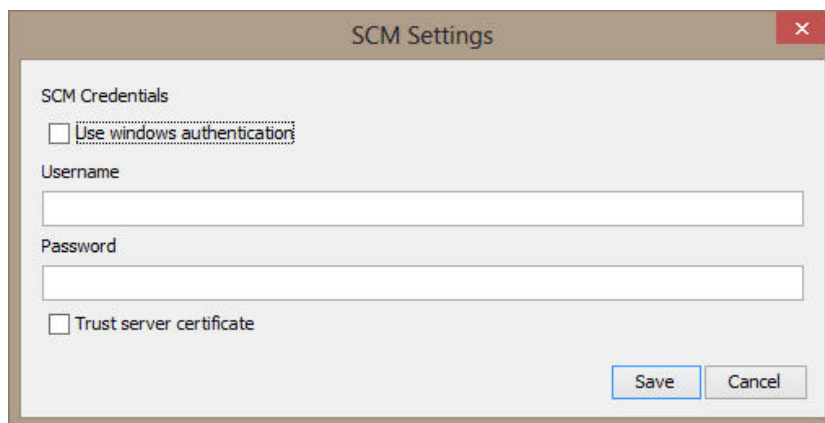
1. From the **SCM** menu, select **Settings**.



The image shows the 'SCM Settings' dialog box. Under the 'SCM Credentials' section, the checkbox 'Use windows authentication' is checked. Below it are empty text boxes for 'Username' and 'Password'. The 'Trust server certificate' checkbox is unchecked. At the bottom right are 'Save' and 'Cancel' buttons.

Select one of the following options, depending on your source control server:

- **Use Windows Authentication:** This option is the default option selected, and performs authentication using the currently logged in user. This is applicable for file-based source control repositories.
- Uncheck, the **Use Windows Authentication** option. Enter the **Username** and **Password** defined on the source control server. If you are working with a secure server (SSL/SSH), select **Trust server certificate**, otherwise you will be unable to access the server.



The image shows the 'SCM Settings' dialog box. Under the 'SCM Credentials' section, the checkbox 'Use windows authentication' is unchecked. Below it are empty text boxes for 'Username' and 'Password'. The 'Trust server certificate' checkbox is unchecked. At the bottom right are 'Save' and 'Cancel' buttons.

**Note:** The credentials are only used to access the SCM itself. If the system is based on a simple network fileshare then HP OO expects the operating system to be able to create the network connection to share. This means that HP OO does not pass on the credentials to the OS for a network connection.

## Checking out a Repository

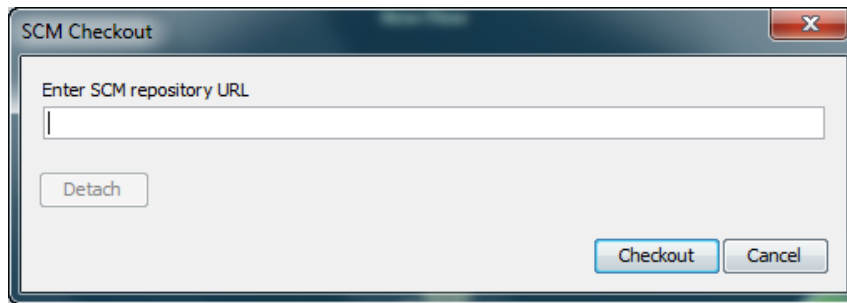
The repository contains all your projects and can be shared by multiple authors. A repository can be hosted either on a file system or on a web server. See [Creating a Local Repository](#) for creating a

file-system based repository.

The repository can be checked out directly from Studio and then projects can be added to it for sharing with other authors.

To check out all projects:

1. From the **SCM** menu, select **Check out**.



The first time you select **Checkout**, you are prompted to enter the URL of the shared folder you created when you setup the local repository. If you are working on a local repository enter the URL using the file URL scheme.

For example:

- If the repository folder is not shared, connect with the following:

```
file:///c:/temp/reop
```

- If the repository is shared and you have all the permissions

```
file://myshared/repo
```

2. Click **Checkout**. This checks out the files from the URL and into Studio's Workspace directory.

The **SCM Messages** window displays the shared folder with the location where the files are checked out to. If projects already exist in the checked-out repository they can now be imported into Studio and worked on.

3. The next step is to either create a new project or import an existing project. From the **Project** pane, either click the **+** icon, to add a new project, or click **📁**, to import an existing project.

**Note:** When you create a new project it is automatically added to SCM.

## Detaching the Workspace from SVN

After checking out a repository, the Studio workspace is detached from the source control server. This is useful if you made a mistake and checked out the wrong repository. Detaching does not delete anything from the repository.



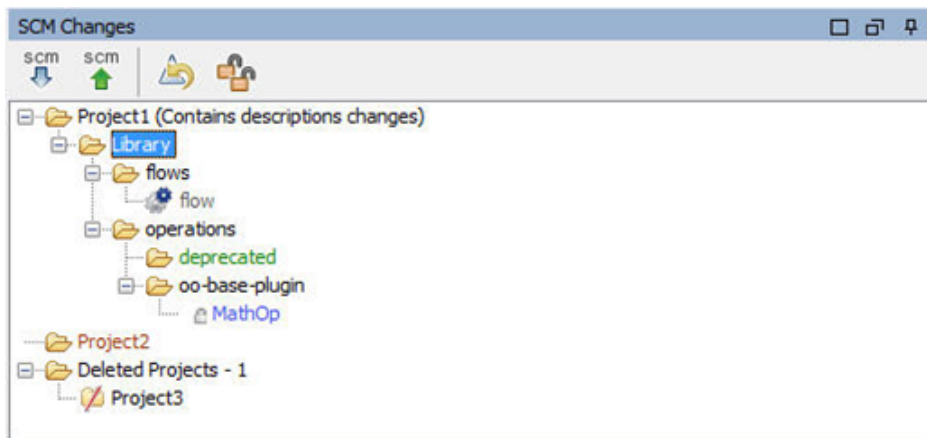
1. From the **SCM** menu, select **Check out**.
2. Click **Detach**.
3. Click **Cancel** to close the window.

## SCM Changes Panel

The SCM Changes panel displays all that was changed in the working copy, compared to the working copy's revision. For example, editing a flow will cause that flow to appear in the changes panel. This panel also shows a list of deleted projects (projects marked for deletion), if such projects exist.

### Types of Changes

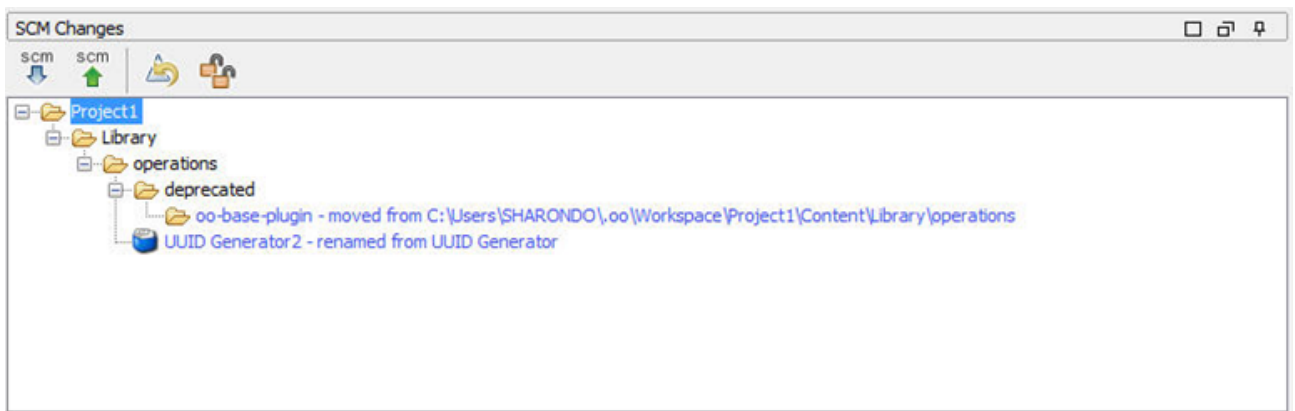
The following screenshot shows the following:



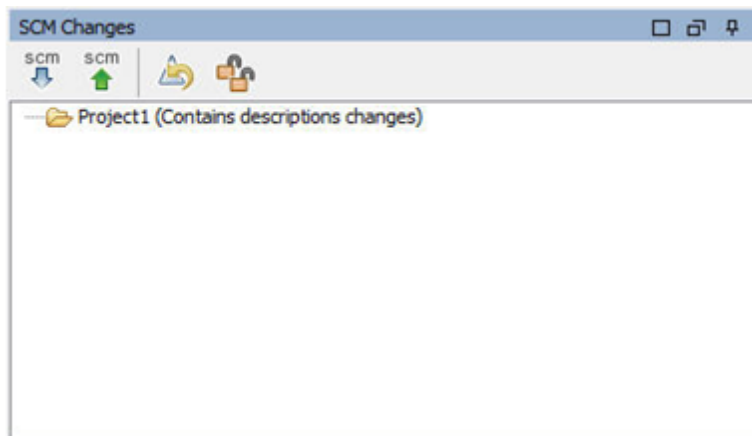
- Deleted flow, named **flow**, and colored gray.
- Added folder, named **deprecated**, colored green
- Unversioned project, named **Project2**, colored brown.
- One deleted project.
- The changed item **MathOp**, in blue, also shows a lock indicator which means it was locked for editing by the user.

### Renamed and Moved Items




Renamed and moved items have a special label with the location where the item was moved or renamed from. In the following example, the **UUID Generator** was renamed to **UUID Generator 2**, and the folder **oo-base-plugin** was moved.



- After you modify one of the following items, it is recommended that you commit the entire project. The change does not occur in the item itself but in the properties file. Therefore, such changes do not show that the item has changed but instead that the project. In this case, **Contains descriptions changes** suffix is appended to the project's name.
  - Flow, operation, or configuration item description
  - Input prompt message (if **Prompt** is selected)
  - Step prompt message (in the **Display** tab)



## The SCM Changes Toolbar

	Update all: Updates the entire Studio workspace.
	Commit all changes: Commits all the changes that show in the changes panel. Only available when there are changes.
	Revert all changes: Reverts all changes that show in the changes panel. Only available when there are changes.

	Unlock all: Unlocks any locked item.
--	--------------------------------------

## Color Codes

Studio shows the following color indications for items:



- Black: Normal item with no changes (Not available in the changes panel)
- Green: Added
- Grey: Deleted (Not available in the projects tree).
- Blue: Modified
- Brown: Unversioned
- Zig-zag underscore: Includes errors.

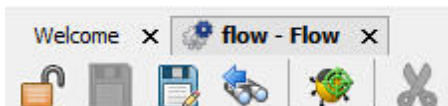
## Working with Multiple Authors

### Locking and Unlocking Items

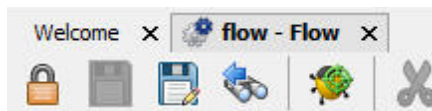
When multiple authors work on the a common project, there is a possibility that two authors will modify the same item simultaneously. Studio attempts to merge all these changes without causing conflicts. Locking an item prevents other authors from working on the item at the same time. Items that are locked by another user can still be edited, but not committed. It is highly recommended to lock any item before editing it in order to prevent conflicts during updates. If you are unable to lock an item, then it is recommended that you do not edit the item in order to avoid conflicts.

**Note:** Make sure you perform an **Update**, before locking, modifying, and committing the item. Committing changes to a locked item, automatically releases the lock.

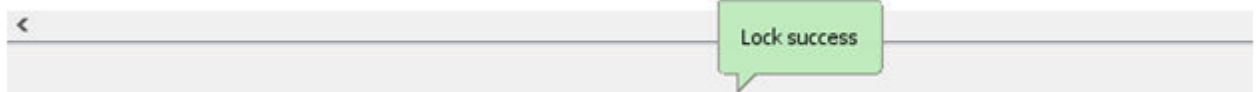
To lock an item, in the top left corner of the items editor window, click the lock item.



If the lock was successful, the icon changes to a lock and displays an SCM message displaying the full path to the locked file and with the user details.



06/09/13 17:04:33 - Locking C:\Users\SHARONDO\.oo\Workspace\Project1\Content\Library\flows\flow.xml  
'flow.xml' locked by user 'sharondo'.



Unlocking is done the same way as above, the icon will change and an SCM unlock message appears.

06/09/13 17:05:06 - Unlocking C:\Users\SHARONDO\.oo\Workspace\Project1\Content\Library\flows\flow.xml  
'flow.xml' unlocked.



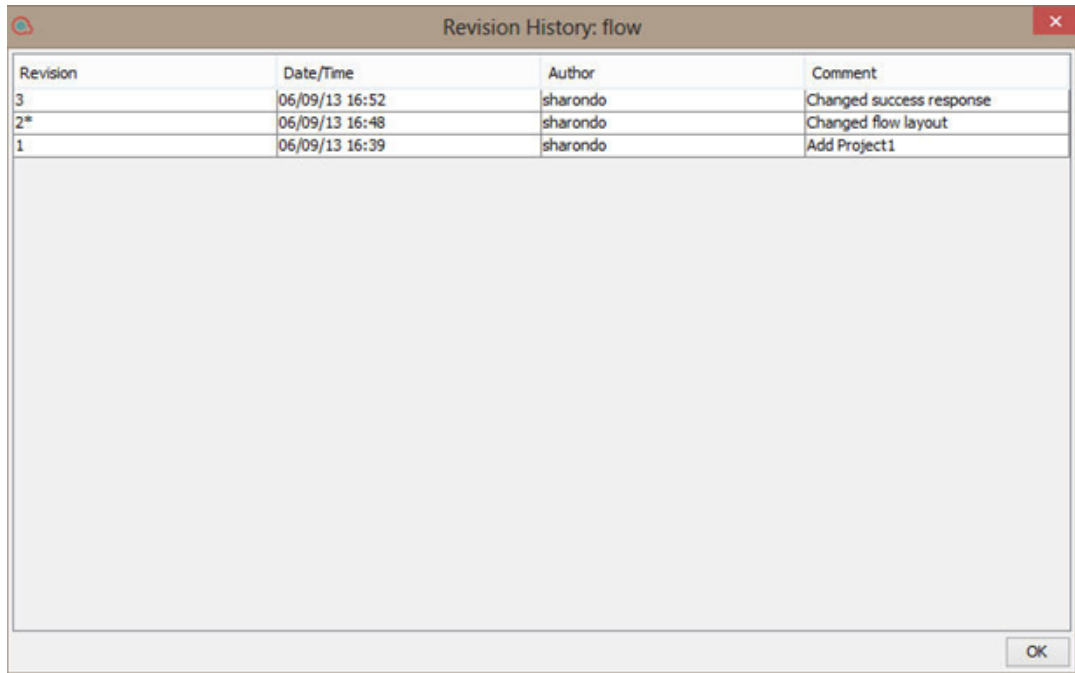
If an item is unable to lock, a warning (yellow) message appears, and the icon will not change.

Committing changes to a locked item, will automatically release its lock.

The lock button is disabled for items that are added, but not committed yet, or for items that are in projects that are not under Studio's workspace.

## Revision History

HP OO Studio also offers version history control. The **Revision History** window, displays the SCM history, it is divided into four columns and contains one line for each commit.

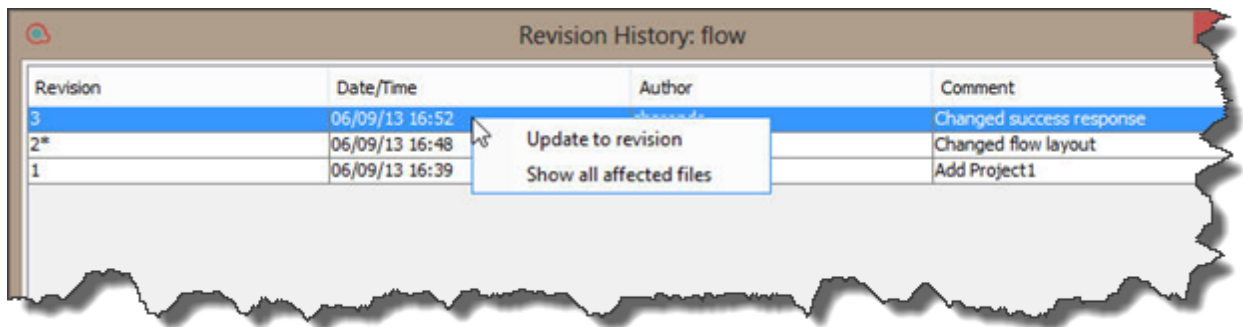


Revision	Date/Time	Author	Comment
3	06/09/13 16:52	sharondo	Changed success response
2*	06/09/13 16:48	sharondo	Changed flow layout
1	06/09/13 16:39	sharondo	Add Project1

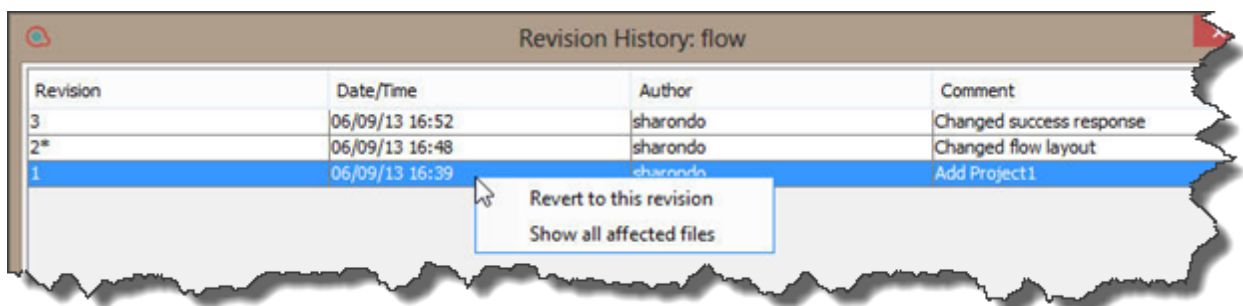
- **Revision:** The revision for the commit. The revision of the local copy is marked with an asterisk. After checkout, the asterisk always marks the latest revision, but after changes have been committed by another user, and before updating the asterisk marks a revision that is not the latest. The screen shot above shows that the local copy is at revision 2, but there was a commit to the file in revision 3.
- **Date/Time:** The date and time in which the revision was committed.
- **Author:** The author who committed the revision. This displays the user's name.
- **Comment:** The comment added during the commit by the user.

When you right-click on a history item, the context menu displays the following options:

- **Update to revision:** Updates the file to the selected revision. Using this option causes some items in the project to be in a different revision than others. In the following example, right-click on the revision above the asterisk (\*).



- **Revert to this revision:** Locally changes the object to the way it was in the selected revision. If for example a flow contains two steps in revision 333, and three steps in revision 337 then reverting it to revision 333 causes it to have two steps again. In order to make the changes visible to other authors a commit to the flow is required. In the following example, right-click on the revision below the asterisk (\*).



- **Show all affected files:** Shows all the files that were changed in the selected revision.

**Note:** Update or revert to revision will only be activated if no changes were applied to item.

## The SCM Message Panel

This displays the message results from the SCM actions. Every action results in a message, and includes what was performed and whether or not it succeeded.

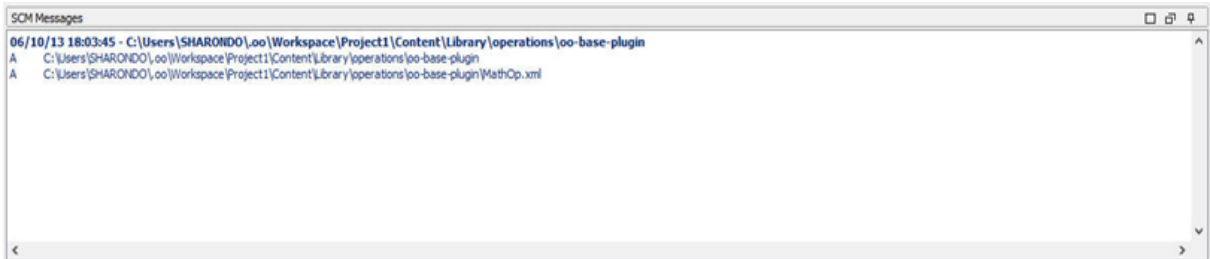
The messages panel can always be cleared by right clicking anywhere inside the panel and then clicking the **Clear All** button.

The following examples show the messages that appear in the SCM Message Panel according to the applied actions:

### Adding a new operation

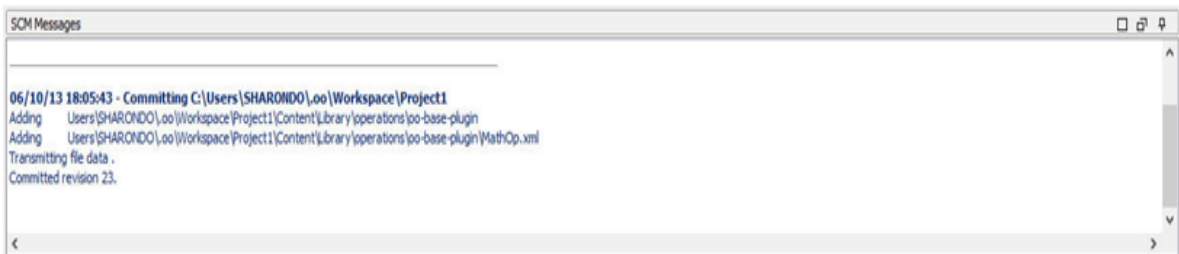
In the following example, a new operation called MathOp was added:

After each addition, the new items are automatically marked to be added. The following SCM message shows that a folder, oo-base-plugin, and the Operation's XML file, MathOp.xml were marked to be added.



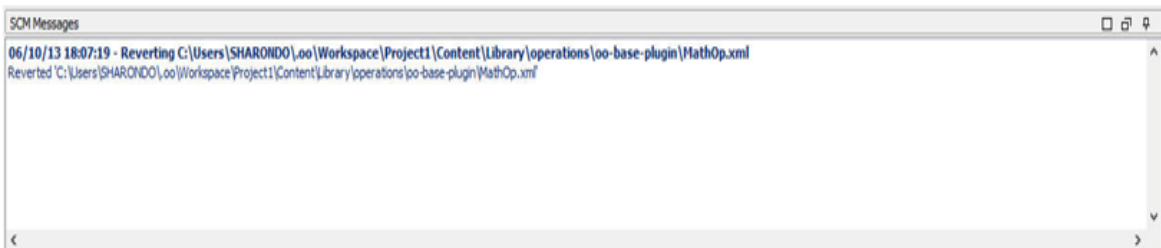
## Committing an item

After committing, the following message appears. This message shows that our previously added items were now successfully committed to the server.



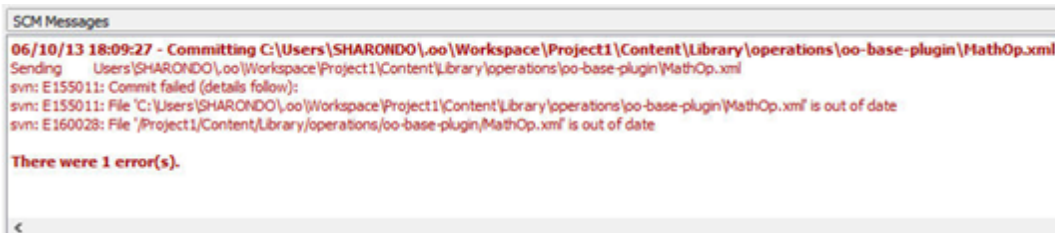
## Changing the item

After making a change to **MathOp** for example, adding a new input, revert the changes. This message shows that a change to MathOp.xml was reverted.



## Error messages

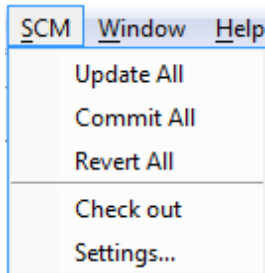
In certain cases, error appear. These are displayed in red. In this example, the changes were attempted to commit, but the file was out-of-date.



## Reference Material

### Source Control Menu

Contains operations that affect the entire workspace



Option	Description
<b>Update All</b>	Updates the workspace directory, but does not import projects into Studio. Projects that are already imported will reflect any updates that were received from the server. Since this command updates the entire workspace, new projects committed by other authors, may be retrieved during update. These projects can be imported into Studio and worked on now.
<b>Commit All</b>	Commits all changes from the active projects. When you click <b>Commit</b> , you can add a comment for the commit.
<b>Revert All</b>	Reverts all changes from the active projects.
<b>Check out</b>	Checks out the repository.
<b>Settings</b>	<p>Allows you to change the way authentication is performed with the source control server.</p> <p><b>Windows Authentication:</b> Performs authentication using the currently logged in user. This is applicable for file-based source control repositories.</p> <p><b>Username and Password Authentication:</b> Performs authentication using the supplied username and password.</p> <p><b>Trust server certificate:</b> If you are working with a secure server (SSL/SSH), select this option, otherwise you will be unable to access the server.</p>



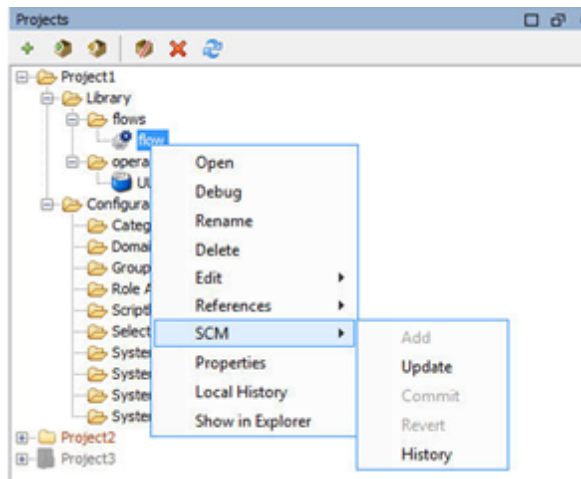
## Projects Panel

Shows the project you're working in, and displays the editable flows, operations, and other HP OO objects that you can use in the project.

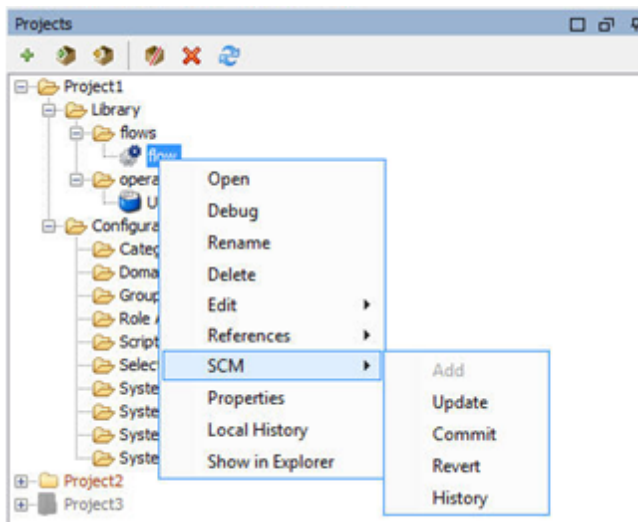
### Context Menu from the Projects panel

The context menu shows all actions that are available for the selected object, according to its state.

Following are the options available when you right-click on a committed flow. In this case only **Update** and **History** are available:



If the object was to be changed by an author, for example by adding a step, then it would have local changes. In this case, **Commit** and **Revert** are available.



Option	Description
<b>Add</b>	Marks an item (Flow, Operation, Configuration Item or Folder) to be added to source control. In Studio <b>Add</b> includes all ancestor and descendant objects. So if a folder is added, any child flows and parent folders are also added. Sibling items are not added. Items that are created from within Studio, are added automatically and committed on the next commit.
<b>Commit</b>	Commits local changes to the server. This option is available for changed items and for folders that have changed children items. A commit works recursively, so when committing a folder, all of its child items are also committed. After you commit, you can add a comment for the commit.
<b>Update</b>	Updates the selected item. This option is unavailable only for items that are locally added but not committed yet. Works recursively and updates all child items.
<b>Revert</b>	<p>Reverts all local changes in the selected items. Changes in Flows, Operations and Configuration Items are reverted. Items that were deleted are restored.</p> <p><b>Important:</b> When adding an item it is automatically marked as added, but when reverting changes on it only the addition mark is removed, but the item still exists in Studio and in the file system. This item can be re-added using <b>Add</b> in the menu, or deleted.</p>
<b>History</b>	The <b>Revision History</b> window, displays the SCM history. See <a href="#">Revision History</a> for more information.

## Deleting a Project

- Select the project that you want to delete, and click the **Delete** button. When click **Delete**, the project is deleted from Studio (un-imported). If the **Delete project from file system (cannot be undone)** option is selected, the project is marked for deletion from the repository, and the deletion occurs on the next commit.

## Troubleshooting

- **I updated my projects and deleted someone else's changes:**

In the following scenario a conflict can occur during update:

- a. User1 edits Flow1, and commits.
- b. User2 edits Flow1, and then updates.

Since User2 will receive updates for a file that has local changes, a conflict will occur.

Studio resolves such conflicts with the updating user's copy. This means that in our case, any changes User1 made to Flow1 will be deleted.

To avoid such cases we recommend always updating and locking any item before editing it.

- **I made changes directly in the file system and something went wrong**

While it is possible to create directories and rename files directly in the file system it is not recommended. It is preferable to perform all tasks from within Studio.

If changes were made in the file system and those changes results in problems, it is recommend to revert those changes using an outside SVN tool such as SlickSVN or TortoiseSVN to perform a clean-up of the workspace.

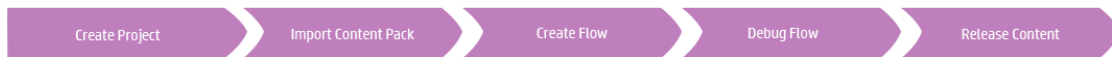
## Working with Content Packs

A content pack is a jar file containing operations, flows, actions, configuration items, and resource bundles. A content pack is the most granular entity that can be delivered for deployment.

You can download a content pack for your specific domain from HPLN, and then import it into your project.

Importing Content Packs to a Project .....	68
Managing Content Packs in a Project .....	70

## Importing Content Packs to a Project



This section describes how to import content packs to Studio.

**Note:** When you install Studio, you can select in the Installation Wizard to import existing content packs. These will load the first time you open Studio.

You need to import the base content pack first, and then import the specific content packs that you need to create your custom flows, for your specific domain. You can download these content packs from HPLN.

A content pack is a collection of operations, flows, actions (Java-based or .Net based), configuration items (such as selection lists, domain terms, and so on) and resource bundles.

Once you have imported a content pack, the files are available in the **Content Pack** pane as read only.

**Note:** You can import a content pack from any network drive.

You cannot import a content pack directly from a remote hosting site. To access remote content packs, first download a copy of the files to your local system, or map to a network drive so that the File Chooser dialog box can navigate to the location.

You can reuse a flow from a different project, by importing a content pack that was created from that flow. For information about packaging a flow into a content pack for reuse, see ["Exporting a Content Pack" on page 225](#).

It is possible to download content packs from multiple sources:


- **My content** – Projects and content packs that are already in Studio.
- **My organization** – Content packs that were developed by various authors in the organization, and which reside in the artifact repository.
- **HP content** - Content that HP releases regularly, which resides on HPLN.
- **Community content** - Content that was contributed to the community by other organizations, which also resides on HPLN.

## What do you want to do?

### Download the content packs


1. Open the HPLN Operations Orchestration Content version 10.00.site.
2. Download the base content pack entitled to a location on your network drive.
3. Download any other content packs that you need for your specific domain to a location on your network drive.

### Import the base content pack when opening Studio for the first time

1. In the **Content Packs** pane, click the **Import** button .
2. Browse to locate the content pack and click **Open**.
3. Click **OK**.

**Note:** It may take a few minutes to import the base content pack.

### Import a content pack

1. In the **Content Packs** pane, click the **Import** button .
2. Browse to locate the content pack and click **Open**.

**Note:** If required, you can select multiple content packs to import at once.

3. Click **OK**. The imported content pack is displayed in the **Content Packs** pane.

### Extract a content pack and open it as a project

If you have upgraded content from previous versions of HP OO, the upgrade tool converts your content into content packs. Follow these steps to open a content packs as an editable project.

1. In your file system (for example, Windows Explorer), unzip the content pack. The content pack is extracted into a project folder.
2. In Studio, select **File > Import Project**.
3. In the Select Project Directory dialog box, browse to locate the project that was created from the unzipped content pack.
4. Click **OK**. You can now edit this project in Studio.

**Note:** If you opened the content pack in Studio before creating the project from it, you need to close the content pack in Studio. The project and content pack will have the same UUID, so they should not be open in Studio at the same time.

## Managing Content Packs in a Project

Once a content pack is imported, you can use the operations in your flows, but note that these operations are read-only. However, you can use the flows as steps as they are read-only also in the content pack.

If you want to modify operations from a content pack, you need to copy them into the project.

**Note:** This is only recommended if you want to add responses or results, if only the inputs are used, then modify them inside the steps.

After an operation is copied to a project, it is detached from the content pack, and becomes editable. You can drag and drop this new operation to a flow and modify its properties.

A operation that is copied from a content pack to a project is a "soft copy". This means that if the operation was originally created by importing an action plug-in, the copied operation continues to reference the original operation. If the action plugin is upgraded and the original operation is updated to call the new version, the copied operation is updated automatically. For more information, see ["Creating Operations" on page 227](#).

## What do you want to do?

### Copy content pack objects to the project

1. Right-click the object to be copied in the **Content Packs** pane and select **Edit > Copy**. To select multiple objects, use the SHIFT and CTRL keys.
2. Right-click the location in the **Projects** pane where you want to paste the object and select **Edit > Paste**.


**Tip:** You can also drag and drop an object from the **Content Packs** pane to the **Projects**

pane.

## Delete a content pack

Deleting a content pack is different from closing it, in that a deleted content pack is permanently removed from the workspace.

**Note:** When you delete a content pack, it is deleted from the workspace, but is not deleted from the file system. If required, you can import it again.

1. Select the content pack and click the **Delete** button .
2. Click **Yes** in the confirmation dialog box.

## Close a content pack

If you close a content pack, it is visible in the **Content Packs** pane, but is grayed out and not available.

When to close a content pack:

- If you have two versions of a content pack in the workspace, you will need to close one before you can work with the other. The two contents packs will have the same UUID, so they should not be open in Studio at the same time.
- If you created a project from a concept pack, you should close the original content pack, before working with the project. The project and content pack will have the same UUID, so they should not be open in Studio at the same time.

1. Select the content pack that you want to close.
2. Right-click the content pack and select **Close**.

**Note:** Alternatively, you can click the **Close**  button in the **Content Packs** pane.

## Open a closed content pack

After a content pack has been closed, you can open it, to work with it again.

1. Select the closed (gray) content pack that you want to open.
2. Right-click the content pack and select **Open**.

**Note:** Alternatively, you can click the **Open**  button in the **Content Packs** pane.

## Display the properties of an imported content pack

Right-click a content pack in the **Content Packs** pane and select **Properties**. The Properties window for the content pack opens, in read-only mode.

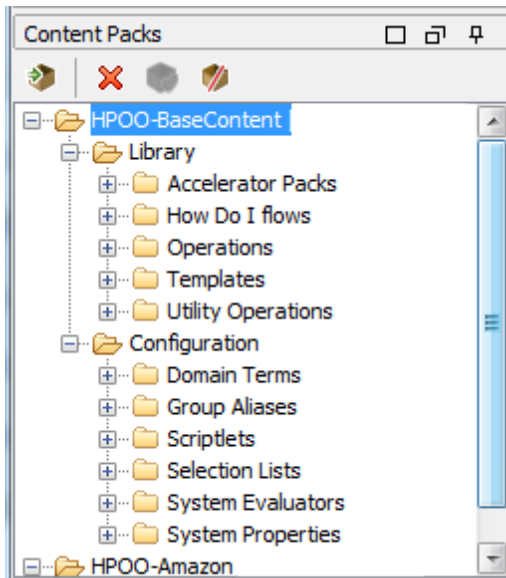
## Display the properties of an object in a content pack





Double-click a flow, operation or other object in the **Content Packs** pane. The Properties window for the object opens, in read-only mode.

# Reference Material

## Content Packs pane

The **Content Packs** pane can display multiple content packs, each with its own hierarchical tree structure.



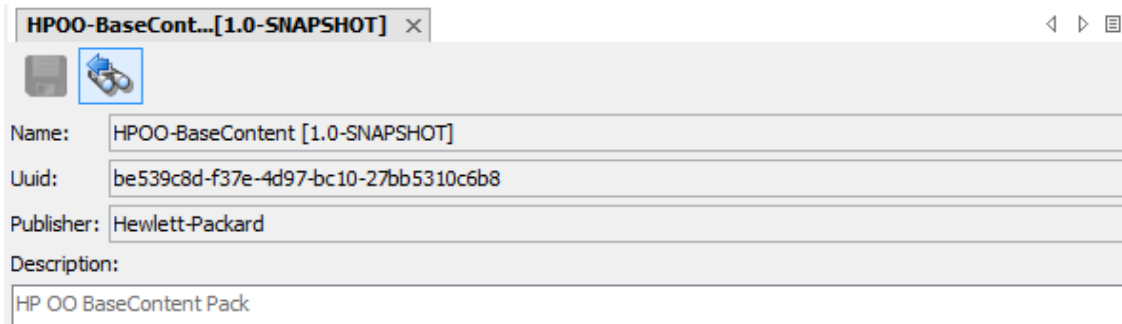
GUI item	Description
<b>Import Content Pack</b> 	Opens the Import Content Pack dialog box, letting you select a content pack to be imported.
<b>Delete</b> 	Deletes the selected content pack.
<b>Open</b> 	Opens the currently selected closed content pack.
<b>Close</b> 	Closes the currently selected content pack, so that it is grayed out.

## Content Pack Properties

When you right-click a content pack and select **Properties**, the Content Pack Properties sheet opens.



The displayed properties are read-only. The information is taken from the content pack jar file that was imported. If the content pack was created from an HP OO project, some of the information is taken from what was entered into the Create Content Pack dialog box.



HP OO BaseContent Pack [1.0-SNAPSHOT]

Name: HPOO-BaseContent [1.0-SNAPSHOT]

Uuid: be539c8d-f37e-4d97-bc10-27bb5310c6b8

Publisher: Hewlett-Packard

Description: HP OO BaseContent Pack

GUI item	Description
<b>Name</b>	The name of the content pack. If the content pack was created from an HP OO project, this name is taken from the project name.
<b>UUID</b>	Unique identifier for the content pack
<b>Publisher</b>	The publisher of the content pack. If the content pack was created from an HP OO project, this is taken from the <b>Publisher</b> field in the Create Content Pack dialog box.
<b>Description</b>	The description of the content pack. If the content pack was created from an HP OO project, this is taken from the <b>Description</b> field in the Create Content Pack dialog box.

## Managing Configuring Items

The **Configuration** folder in a project contains a number of items that you can set up: domain terms, group aliases, system accounts, system properties, and so on.

If an authoring team are working in different projects and need the same configuration items, it is recommended to create a separate, shared project containing the configuration items.

Configuring Categories .....	74
Configuring Domain Terms .....	76
Configuring Group Aliases .....	79
Configuring Scriptlets .....	82
Configuring Selection Lists .....	85
Configuring System Accounts .....	88
Configuring System Filters .....	90
Configuring System Properties .....	95

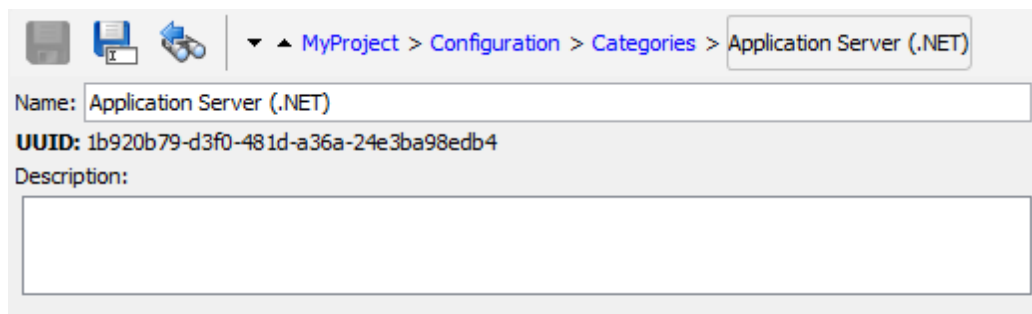
## Configuring Categories

Categories are different classifications that can be assigned to a flow. A number of categories are installed with Studio, but you can also create your own categories.

Users might use categories to create reports that indicate the health of key infrastructure components. For example, if you assign the category **Server** to all the flows that check server health, then a report that finds only flows that were assigned the **Server** category could highlight the health of the servers on your network.

You can also use categories for filtering a search. For example, you can run a search that only looks for flows that have the **Security** category.

Categories are stored in the **Configuration\Categories** folder.



▼ ▲ MyProject > Configuration > Categories > Application Server (.NET)

Name: Application Server (.NET)

UUID: 1b920b79-d3f0-481d-a36a-24e3ba98edb4

Description:

## What do you want to do?

### Create a category

1. In the **Projects** pane, expand the **Categories** folder.
2. Right-click the **Categories** folder, and then click **New**.
3. In the dialog box that appears, type a name for the new category , and then click **OK**.
4. In the **Description** box, type a description of the new category .
5. Click **Save**.

### Change a category

1. In the **Projects** pane, expand the **Configuration** and **Categories** folders, and double-click the category to open its editor.
2. Double-click the category you want to change and type the new value.

### Rename a category

1. In the **Projects** pane, expand the **Configuration** and **Categories** folders, and double-click the category to open its editor.
2. In the **Name** box, type the new name for the category .
3. Click **Save**.

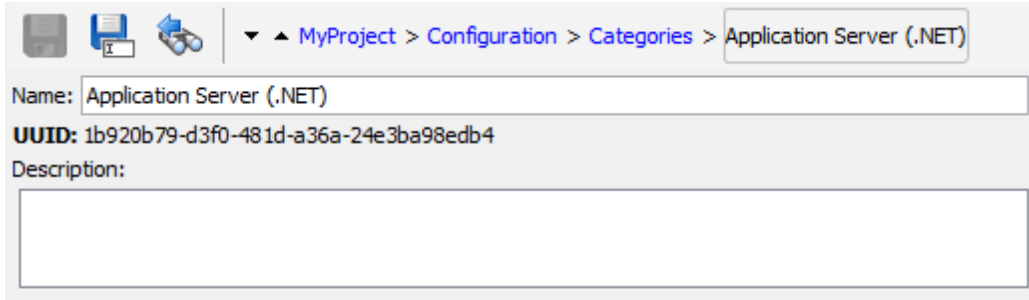
### Delete a category

Before you delete a category , it is recommended to use the **What Uses This** function to check that other items do not depend on it. For more information, see ["Finding Out How Flows and Operations are Used" on page 243](#).

1. In the **Projects** pane, expand the **Configuration** and **Categories** folders.
2. Right-click the category and select **Delete**.
3. Click **Yes** in the confirmation windows.

## Reference Material

### Categories editor



▼ ▲ MyProject > Configuration > Categories > Application Server (.NET)

Name: Application Server (.NET)

UUID: 1b920b79-d3f0-481d-a36a-24e3ba98edb4

Description:

GUI item	Description
Name	The name of the category.
Description	(Optional) Description of the category.

## Configuring Domain Terms

Domain terms are attributes that you can assign to flows and inputs. For example, you might create domain terms for the various kinds of servers in your system, and then you could get a step to run against a certain type of server only.

Domain terms can be used for specialized selection lists. For example, you can create a domain term for different kinds of actions. The values in this domain term could be **Restart**, **Reboot**, **Open**, and so on.

In another example, to specify that a flow run against certain classes of servers and not others, you can add domain terms for the various kinds of servers in your system, and provide a user prompt at the start of the flow, in which the user needs to select the classes of servers that you want to run a given flow against.

Domain terms can have values by default, obtain their values from the flow's inputs, or have the values that you specify for them.

Domain terms are stored in the **Configuration\Domain Terms** folder.

The screenshot shows a configuration editor window. At the top, there is a breadcrumb navigation path: **MyProject1 > Configuration > Domain Terms > Severity**. Below this, the **Name** field is set to **Severity**. The **UUID** is **46f698c7-f7d1-4f68-91d9-13e47a02819a**. The **Description** field contains the text **Information, Warning, Error, Critical.**. Below the description field are two buttons: **Add** and **Remove**. At the bottom, there is a table with two columns: **Name** and **Description**.

Name	Description
Information	
Warning	
Error	
Critical	

## What do you want to do?

### Create a domain term

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click the **Domain Terms** folder, and then click **New**.
3. In the dialog box that appears, type a name for the new domain term, and then click **OK**.
4. In the **Description** box, type a description of the new domain term.
5. Click **Add** to add a new domain term value.
6. In the **Name** column, enter the name for the domain term value.
7. (Optional) In the **Description** column, enter a description for the domain term value.
8. Click **Save**.

### Remove a domain term value

1. In the **Projects** pane, expand the **Configuration** and **Domain Terms** folders, and double-click the domain term to open its editor.
2. Highlight the value and click **Remove**.

### Change a domain term value

1. In the **Projects** pane, expand the **Configuration** and **Domain Terms** folders, and double-click the domain term to open its editor.

2. Double-click the value you want to change and type the new value.

## Rename a domain term

1. In the **Projects** pane, expand the **Configuration** and **Domain Terms** folders, and double-click the domain term to open its editor.
2. In the **Name** box, type the new name for the domain term.
3. Click **Save**.

## Delete a domain term

Before you delete a domain term, it is recommended to use the **What Uses This** function to check that other items do not depend on it. For more information, see ["Finding Out How Flows and Operations are Used" on page 243.](#)

1. In the **Projects** pane, expand the **Configuration** and **Domain Terms** folders.
2. Right-click the domain term and select **Delete**.
3. Click **Yes** in the confirmation windows.

# Reference Material

## Domain Term editor

▼ ▲ MyProject1 > Configuration > Domain Terms > Severity

Name: Severity

**UUID:** 46f698c7-f7d1-4f68-91d9-13e47a02819a

Description:

Information, Warning, Error, Critical.

Add Remove

Name	Description
Information	
Warning	
Error	
Critical	

GUI item	Description
<b>Name</b>	The name of the domain term.

<b>Description</b>	(Optional) Description of the domain term.
<b>Add</b>	Click <b>Add</b> to add a new item to the domain term list.
<b>Remove</b>	Click <b>Remove</b> to remove the selected item from the domain term list.
<b>Name column</b>	Enter the name of the item in the domain term list.
<b>Description column</b>	(Optional) Enter a description of the item in the domain term list.

## Configuring Group Aliases

### *RAS Groups*

A RAS group is a logical collection of RASes. Deployments can benefit from having more than a single RAS in a specific environment. For example, you are managing a remote data center in which you need two RASes to be able to withstand the action execution load, or simply for high availability of the RASes in that data center.

You can define a RAS group in the server using the RESTful API. For more information, see the *HP OO Application Program Interface (API) Guide*.

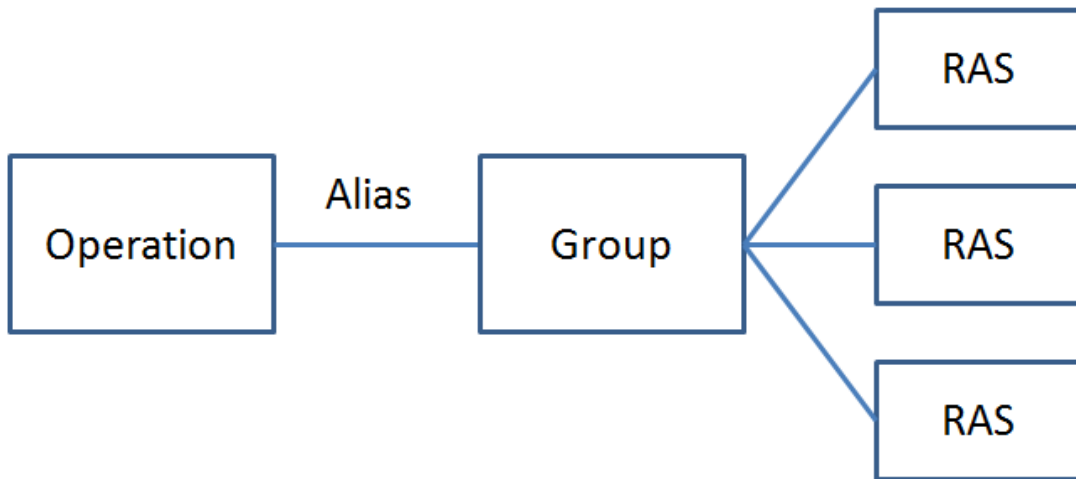
### *Group Aliases*

Group aliases let you separate between assigning an operation to a RAS during authoring time and in the runtime environment.

1. At authoring time, the author defines the operation to execute on a group alias rather than a group.
2. At runtime, the administrator maps the alias to a RAS group in the runtime environment, using the Central RESTful API. There is no need for the administrator to dive into the flows and modify the RAS assignment manually.

As a fallback, if the group alias is identical to the group name, it is mapped automatically to that group.

Optionally, at triggering time, it is possible to override the group alias and map the operation to a different RAS group.



For example, you have a group of three RASes that run on the Oracle client. You create an operation that runs a query on Oracle. By using an alias for this group, you are telling HP OO that this operation needs to run on one of the RASes in this group. The choice of which RAS to use is determined at runtime, and does not need to be configured in the operation.

Group aliases are stored in the **Configuration\Group Aliases** folder.

▼ ▲ MyProject1 > Configuration > Group Aliases > Oracle

Name: Oracle

UUID: bed86b7b-78af-49bb-8be0-b6d60eba82ed

Description:

Use the Oracle Group.

## What do you want to do?

### Create a group alias

For example, you have created a RAS group of three RASes that run on the Oracle client. You need to create a group alias that will direct operations to work with this group, and thus to run on one of these RASes.

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click the **Group Aliases** folder, and then click **New**.
3. In the dialog box that appears, type a name for the new group alias, and then click **OK**.
4. In the **Description** box, type a description of the new group alias.
5. Click **Save**.




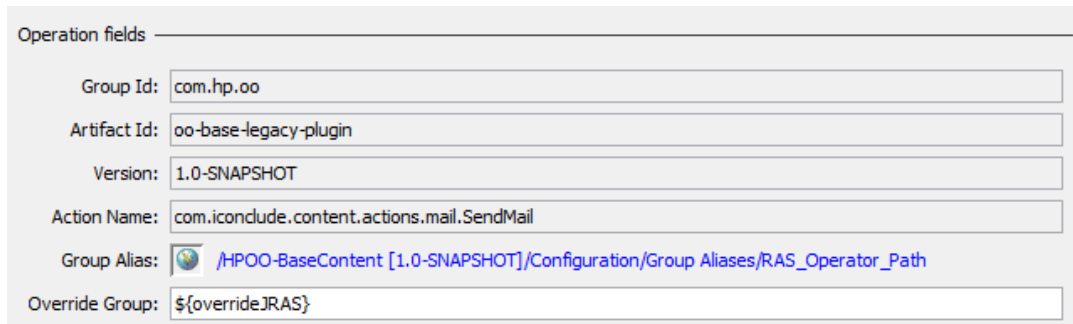
## Map the group alias to a group

Map the group alias to the group in the runtime environment using the Central RESTful API.

For more information, see the *HP OO Application Program Interface (API) Guide*.

## Use the group alias in an operation

1. Create a new operation from an action plugin, as described in ["Creating Operations" on page 227](#).
2. In the **Inputs** tab, in the **Operation Fields** section, click the **Group Alias**  button.




Operation fields

Group Id:

Artifact Id:

Version:

Action Name:

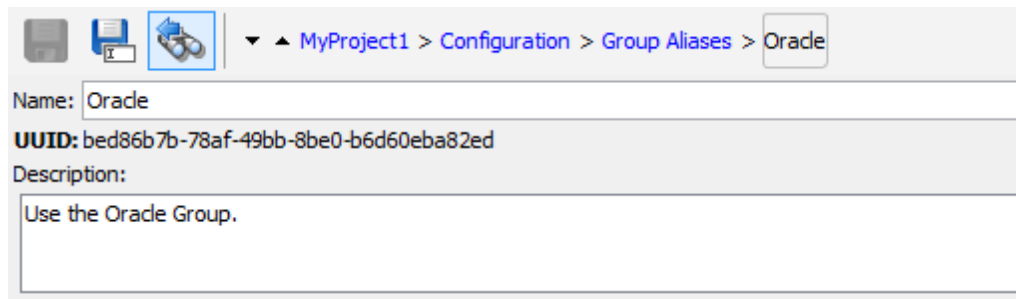
Group Alias: 




Override Group:

3. Browse to select a group alias that was set up in the **Configuration/Group Aliases** folder.
4. (Optional) In the **Override Group** box, enter a different group, if you need to override the current group with a different one.
5. Save the operation.

## Reference Material

### Group Alias editor



   ▼ ▲ MyProject1 > Configuration > Group Aliases > Orade

Name:

UUID: bed86b7b-78af-49bb-8be0-b6d60eba82ed

Description:

GUI item	Description
Name	The name of the group alias.
Description	(Optional) Description of the group alias.

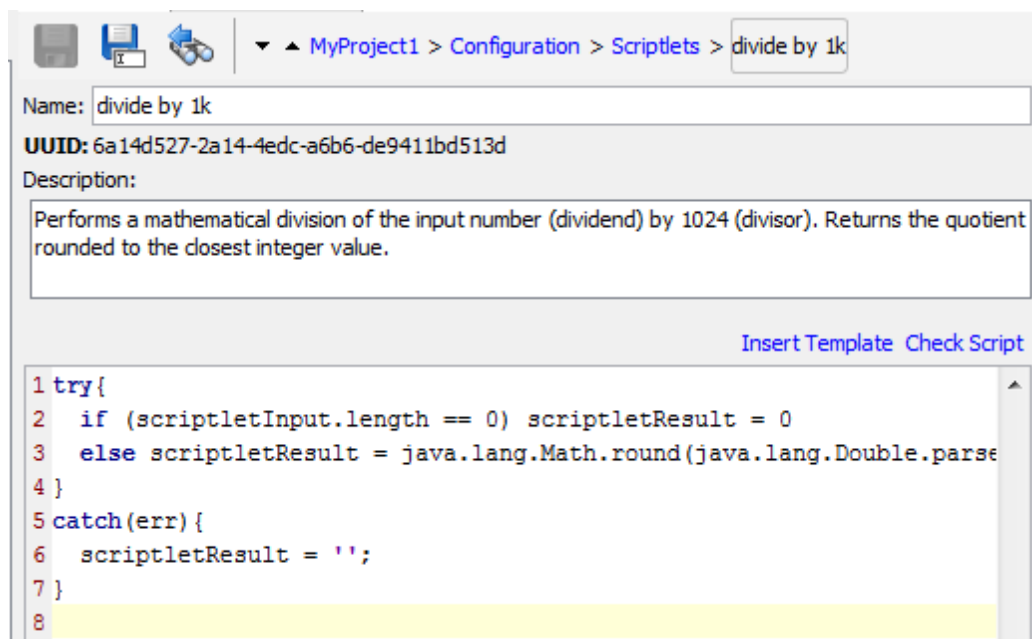
## Configuring Scriptlets

Scriptlets (written in JavaScript) are optional parts of an operation that you can use to manipulate data.

You can use scriptlets to:

- Filter the results of an operation, flow, or step
- Determine the response of an operation
- Manipulate data in a subflow before passing the data to the parent flow

For example, the **divide by 1k** scriptlet performs a mathematical division of the input number (dividend) by 1024 (divisor) and returns the quotient rounded to the closest integer value.



You can create a system scriptlet from scratch or take an existing scriptlet in an operation and save it as a shared system scriptlet. The resulting scriptlet is independent of the context for it was created and can be reused in any operation, flow, or step.

System scriptlets are stored in the **Configuration\Scriptlets** folder.


For more information about using scriptlets, see ["Using Scriptlets in a Flow" on page 200](#).

## What do you want to do?


### Save an existing scriptlet as a system scriptlet

1. Under the **Scriptlet** tab in the **Properties** sheet or Step Inspector, open the scriptlet that you

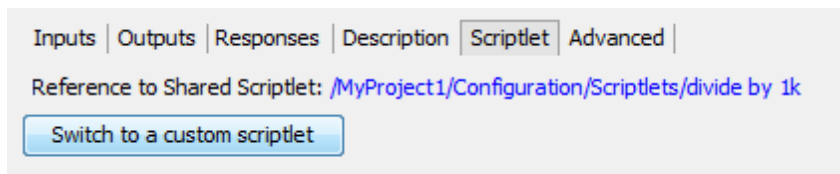
want to save as a system scriptlet.

2. In the **Projects** pane, expand the **Configuration** and **Scriptlets** folders.
3. Drag the **Scriptlet**  icon from the **Scriptlet** tab of the **Properties** sheet or Step Inspector to the **Configuration\Scriptlets** folder.
4. To rename the new system scriptlet, right-click it, click **Rename**, and change its name.

## Use a system scriptlet in an operation, flow, or step

1. Open the **Scriptlet** tab of the **Properties** sheet or Step Inspector for the operation, flow, or step on which you want to use the system scriptlet .
2. In the **Projects** pane, expand the **Configuration** and **Scriptlets** folders.
3. Drag the scriptlet from the **Scriptlets** folder to the **Scriptlet**  icon in the **Scriptlet** tab of the **Properties** sheet or Step Inspector.

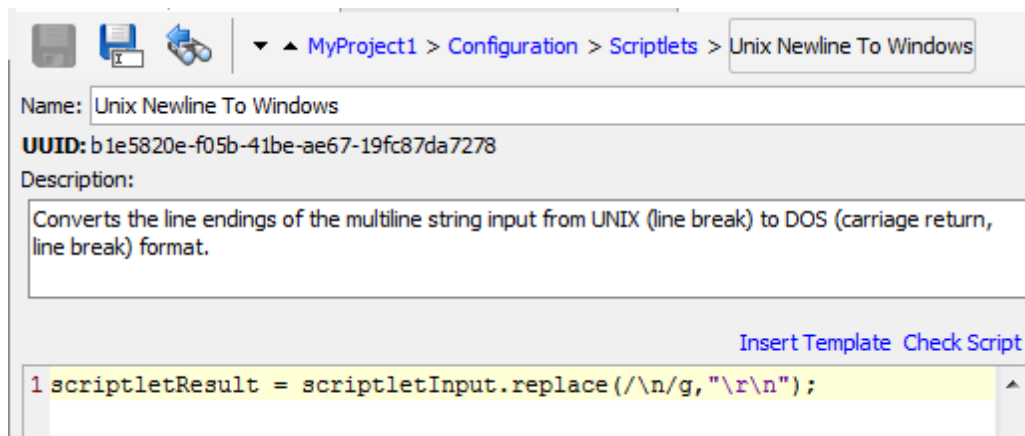
The Scriptlet tab shows that there is now a reference to a shared scriptlet.



## Create a system scriptlet

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click the **Scriptlets** folder, and then click **New**.
3. Enter a name for the scriptlet and click **OK**.

The Scriptlet Editor opens.



4. In the **Description** box, describe the purpose of the scriptlet.
5. Type the scriptlet in JavaScript.
6. (Optional) Click **Insert Template** and follow the guidelines in the template to write the scriptlet.
7. Click **Check Script** to check for errors. ["Filtering Output and Results" on page 158](#)
8. Click **Save**.

The scriptlet is saved in the **Scriptlets** folder and is now available for use in any operation, flow, or step.

### Edit a system scriptlet

1. In the **Projects** pane, expand the **Configuration** and **Scriptlets** folders.
2. Double-click the system scriptlet that you want to edit.
3. Modify the scriptlet and click **Save**.

### Delete a system scriptlet

Before you delete a system scriptlet, it is recommended to use the **What Uses This** function to check that other items do not depend on it. For more information, see ["Finding Out How Flows and Operations are Used" on page 243](#).

1. In the **Projects** pane, expand the **Configuration** and **Scriptlets** folders.
2. Right-click the system scriptlet and select **Delete**.
3. Click **Yes** in the confirmation window.

## Reference Material

### Scriptlet Editor

▼ ▲ MyProject1 > Configuration > Scriptlets > Unix Newline To Windows

Name: Unix Newline To Windows

UUID: b1e5820e-f05b-41be-ae67-19fc87da7278

Description:

Converts the line endings of the multiline string input from UNIX (line break) to DOS (carriage return, line break) format.

Insert Template Check Script

```
1 scriptletResult = scriptletInput.replace(/\n/g, "\r\n");
```

GUI item	Description
Name	The name of the scriptlet.
Description	(Optional) Description of the purpose of the scriptlet.
Insert Template	Click <b>Insert Template</b> to see guidelines to help you write your scriptlet.
Check Script	Click <b>Check Script</b> to check the scriptlet for errors.

## Configuring Selection Lists

Selection lists are lists of items that can be provided in user prompts in a flow.

**Important:** In the current version, user prompts are not supported. It is possible to set up selection lists, in preparation for later versions, where user prompts will be supported, but do not use them in flows with this version.

For example, if the flow user needs to provide a step in the flow with the service status, you can create an input whose data source is a selection list and specify the Service Status selection list (whose members are Running, Stopped, and Paused).

Selection lists are stored in the **Configuration\Selection Lists** folder.

▼ ▲ MyProject1 > Configuration > Selection Lists > Character Sets

Name: Character Sets

UUID: 87914bba-1646-4a9e-82b3-8f45ed27cd5b

Description:  
Different types of character sets

Add Remove

Value
UTF-8
UTF-16
UTF-32
EUC-JP
ISO-2022-JP
Shift_JIS
Windows-31J

## What do you want to do?

### Create a selection list

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click the **Selection Lists** folder, and then click **New**.
3. In the dialog box that appears, type a name for the new selection list, and then click **OK**.
4. In the **Description** box, type a description of the new selection list.
5. Click **Add** to add a new selection list value.
6. In the **Value** column, enter the name for the selection list value.
7. Click **Save**.

### Remove a selection list value

1. In the **Projects** pane, expand the **Configuration** and **Selection Lists** folders, and double-click the selection list to open its editor.
2. Highlight the value and click **Remove**.

### Change a domain term value

1. In the **Projects** pane, expand the **Configuration** and **Selection Lists** folders, and double-click the selection list to open its editor.

2. Double-click the value you want to change and type the new value.

## Rename a selection list

1. In the **Projects** pane, expand the **Configuration** and **Selection Lists** folders, and double-click the selection list to open its editor.
2. In the **Name** box, type the new name for the selection list.
3. Click **Save**.

## Delete a selection list

Before you delete a selection list, it is recommended to use the **What Uses This** function to check that other items do not depend on this selection list. For more information, see ["Finding Out How Flows and Operations are Used" on page 243](#).

1. In the **Projects** pane, expand the **Configuration** and **Selection Lists** folders.
2. Right-click the selection list and select **Delete**.
3. Click **Yes** in the confirmation windows.

# Reference Material

## Selection List editor

▼ ▲ MyProject1 > Configuration > Selection Lists > Character Sets

Name: Character Sets

UUID: 87914bba-1646-4a9e-82b3-8f45ed27cd5b

Description:

Different types of character sets

Add Remove

Value
UTF-8
UTF-16
UTF-32
EUC-JP
ISO-2022-JP
Shift_JIS
Windows-31J

GUI item	Description
----------	-------------

<b>Name</b>	The name of the selection list.
<b>Description</b>	(Optional) Description of the purpose of the selection list.
<b>Add</b>	Click <b>Add</b> to add a new value to the selection list.
<b>Remove</b>	Click <b>Remove</b> to remove the selected value from the selection list.
<b>Value</b>	Enter the value in the selection list.

## Configuring System Accounts

A system account is an object that contains an account's credentials (user name and password), while protecting the credentials from being viewed other than in the installation of Studio on which the system account was created.

Flow authors can use system accounts when creating a flow. For example, you can set an input source to be credentials from a system account. See ["Specifying the Input Source" on page 127](#).

**Note:** The system accounts defined here are for Studio only. System accounts also need to be set up for execution. This is done via API. For more information, see the *HP OO Application Program Interface (API) Guide*.

Users never see the system account name that provides a flow with user account credentials for access to a remote machine. Thus the credentials are protected from decryption, and the system account name is hidden from the user.

System accounts are stored in the **Configuration\System Accounts** folder.

**Note:** The following characters cannot be used in a system account name: <>\\\"/;%.

The screenshot shows the Studio configuration interface for a system account. The breadcrumb navigation at the top indicates the path: MyProject1 > Configuration > System Accounts > John Citizen. The main form contains the following fields and controls:

- Name:** A text field containing "John Citizen".
- UUID:** A text field containing the value "6fd1f200-d435-441f-9efa-925c67c7d9ea".
- Description:** A large, empty text area.
- Credentials:** A section containing:
  - User Name:** A text field containing "<domain>\John\_Citizen".
  - Password:** A text field filled with dots, representing a masked password.
  - Assign Password:** A button with a dashed border, located to the right of the password field.



## What do you want to do?

### Create a system account

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click the **System Accounts** folder, and then click **New**.
3. In the dialog box that appears, type a name for the new system account, and then click **OK**.
4. (Optional) In the **Description** box, type a description of the system account.
5. In the **User Name** box, type the user name of the account that the system account represents, using the following syntax:

<domain>\<username>

6. Click the **Assign Password** button.
7. In the **Password** box, type the password, and then in the **Confirm Password** box type it again.
8. Click **Save**.

### Edit a system account

1. In the **Projects** pane, expand the **Configuration** and **System Accounts** folders.
2. Double-click the system account that you want to edit.
3. Make your changes in the editor and click **Save**.

### Delete a system account

1. In the **Projects** pane, expand the **Configuration** and **System Accounts** folders.
2. Right-click the system account and select **Delete**.
3. Click **Yes** in the confirmation windows.

**Note:** If you delete a system account from a content pack and then redeploy the content pack, the system account is not removed from the database. If this occurs, you will need to remove the system account via rest API:

Do DELETE on: /oo/rest/system-accounts/<sa\_name>.

For more information about working with rest APIs, see the *HP OO Application Program Interface (API) Guide*.

## Reference Material

### System Accounts editor

GUI item	Description
<b>Name</b>	The name of the system account.  <b>Note:</b> The following characters cannot be used in a system account name: <>\"/;%.
<b>Description</b>	(Optional) Description of the purpose of the system account.
<b>User Name</b>	The user name of the account that the system account represents, using the syntax <domain>\<username>.
<b>Assign Password</b>	Click to open the Enter Password dialog box, where you will need to type the password twice.

## Configuring System Filters

Filters are used to extract and modify parts of an operation's output or a step's result. A system filter is available system-wide, to be used in multiple steps and operations.

For example, the filters used in a Ping operation might be useful for other ping operations.

You can create a system filter from scratch or take an existing filter in an operation and save it as a system filter. The resulting system filter is independent of the operation for it was created and can be reused in any output or result.

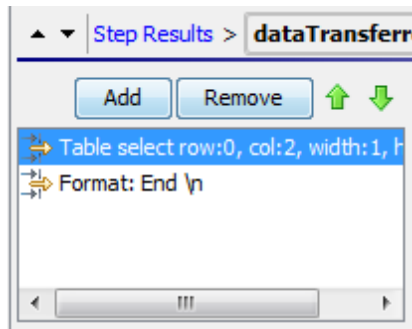
System filters are stored in the **Configuration\System Filter** folder.

The screenshot shows the Studio Authoring Guide interface for configuring a system filter. The breadcrumb navigation at the top indicates the path: **MyProject1 > Configuration > System Filters > Table**. The filter is named **TableFilter** with a UUID of **ba2eacc5-f1f3-4c0b-ad25-4d1e97d9a5a3**. The description field is empty. Below the description, a text label states: "Parses the input as a table and sorts it on a specified column". The configuration options include:   
- **Column Delimiter:** A dropdown menu set to **Whitespace**.   
- **Row Delimiter:** A dropdown menu set to **NewLine**.   
- **First Row is Header:** A checkbox that is currently unchecked.   
- **Strip First Row of Result:** A checkbox that is currently unchecked.   
- **Sort On Column:** A dropdown menu set to **1**.   
- **Select Row:** A text input field containing **0**.   
- **Select Col:** A text input field containing **3**.   
- **Select Width:** A text input field containing **1**.   
- **Select Height:** A text input field containing **1**.   
A tooltip is visible over the **First Row is Header** checkbox, displaying the text: "If checked, interpret the first row as column headers". At the bottom right, there is a **Test Filter** button. Below this button is a **Test Filter Input** text area. At the bottom of the interface, there are icons for file operations (document, folder, scissors) and buttons for **Clear** and **Quick Command**.

## What do you want to do?

### Save an existing filter as a system filter

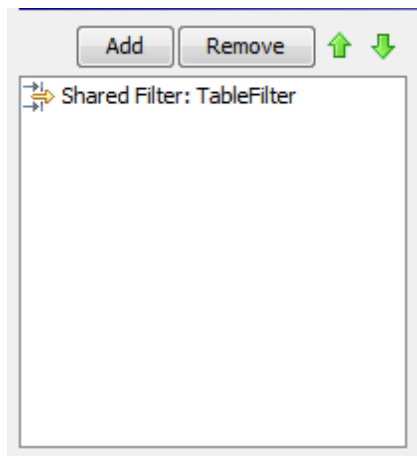
1. Open the operation and, in the Filter Editor, select the filter you want to save as a system filter.
2. In the **Projects** pane, expand the **Configuration** and **System Filters** folders.
3. In the operation's Filter Editor, drag the filter from the **Filter** list to the **System Filters** folder.



4. To rename the new system filter, right-click it, click **Rename**, and change its name.

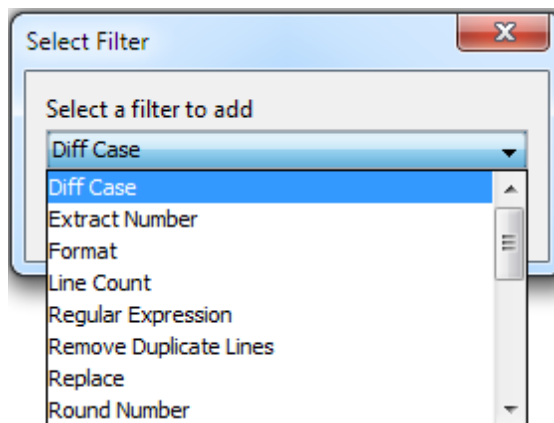
### Use a system filter in an output or result

1. Open the Filter Editor of the output or result on which you want to use the system filter.
2. In the **Projects** pane, expand the **Configuration** and **System Filters** folders.
3. Drag the filter that you want to use from the **System Filters** folder to the **Filter** list in the Filter Editor.



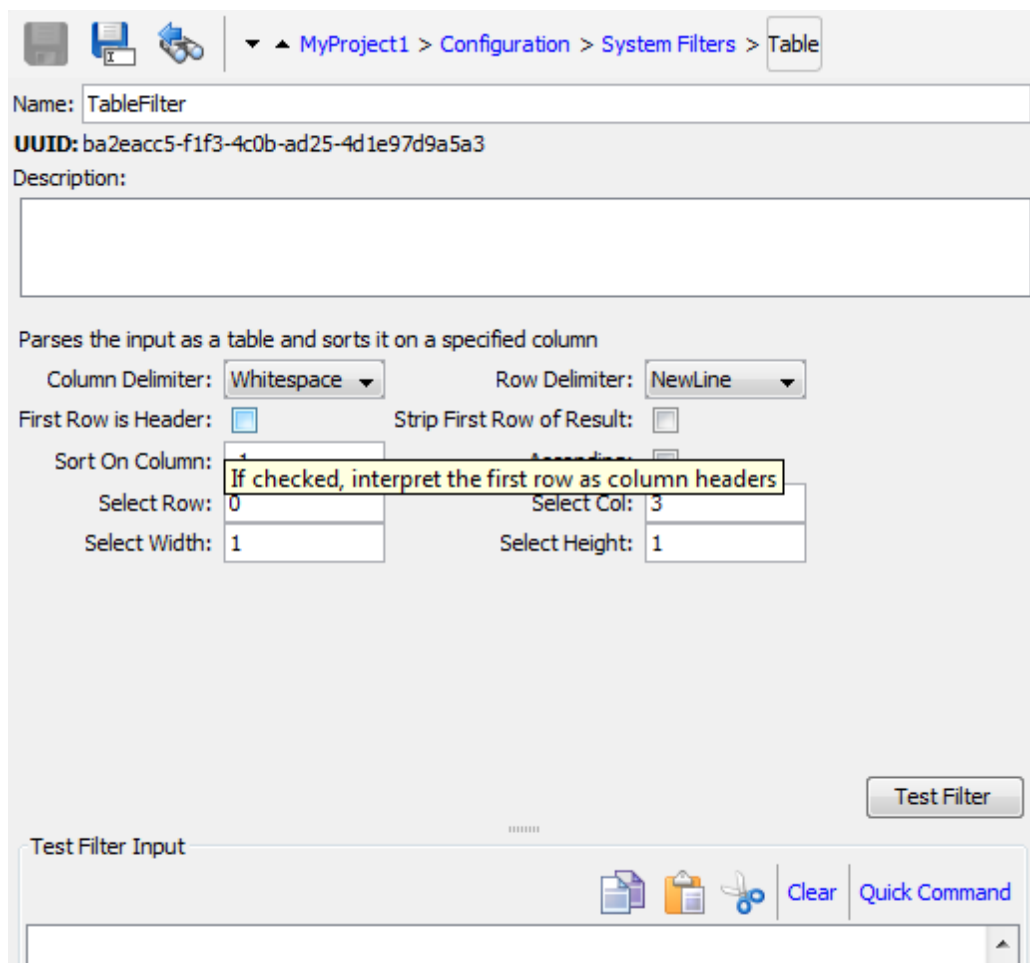
### Create a system filter

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click the **System Filters** folder, and then click **New**.
3. From the **Select Filter** list, select the type of filter.



4. Enter a name for the filter and click **OK**.

The Filter Editor opens. The appearance of the Filter Editor varies, depending on the type of filter you selected.



5. In the **Description** box, describe the purpose of the filter.
6. Enter the text, string, expression value, or scriptlet with which to filter the output or result. For information about the different filter options, see ["Filtering Output and Results" on page 158](#).
7. Test the filter:
  - a. Click **Clear** to clear the **Test Filter Input** box.
  - b. Click **Quick Command**.
  - c. Type a command that generates the desired data.
  - d. Click **OK**. The output of the command appears in the **Test Filter Input** box.

For more information about testing filters, see ["Filtering Output and Results" on page 158](#).

8. Click **Save**.

The filter is saved in the **System Filters** folder and is now available in the **Validation Format** list in the Input Editor.

## Edit a system filter

1. In the **Projects** pane, expand the **Configuration** and **System Filters** folders.
2. Double-click the system filter that you want to edit.
3. Modify the filter and click **Save**.

## Delete a system filter

Before you delete a system filter, it is recommended to use the **What Uses This** function to check that other items do not depend on it. For more information, see ["Finding Out How Flows and Operations are Used" on page 243](#).

1. In the **Projects** pane, expand the **Configuration** and **System Filters** folders.
2. Right-click the system filter and select **Delete**.
3. Click **Yes** in the confirmation window.

# Reference Material

## Filter Editor

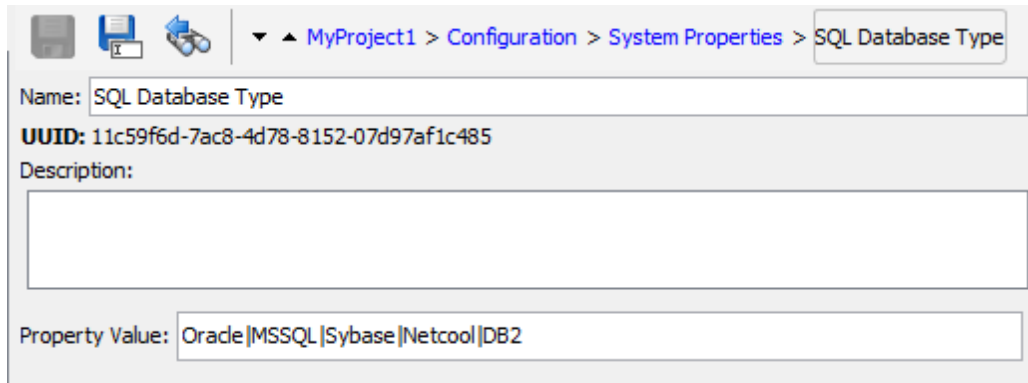
The appearance of the Filter Editor varies, depending on the type of filter you selected. For information about the different options, see *Filter Options* in ["Filtering Output and Results" on page 158](#).

GUI item	Description
Name	Displays the name of the system filter
Description	Enter a description of the system filter

## Configuring System Properties

System properties are global flow variables with values that never change and so can be used in any flow, saving you the time of recreating a flow variable each time you need to use it. Any reference to a system property obtains the system property's value.

For example, the **SQL Database Type** system property lists the different types of SQL databases.



The screenshot shows the Studio configuration interface. At the top, there is a breadcrumb navigation path: **MyProject1 > Configuration > System Properties > SQL Database Type**. Below this, the 'Name' field is set to 'SQL Database Type'. The 'UUID' is displayed as '11c59f6d-7ac8-4d78-8152-07d97af1c485'. The 'Description' field is empty. At the bottom, the 'Property Value' field contains the text 'Oracle|MSSQL|Sybase|Netcool|DB2'.

System properties are stored in the **Configuration\System Properties** folder.

## Best Practices

Be cautious about creating system properties or changing their values, because they:

- Have global scope, becoming part of any flow run's context when the run is begun. As a result, changing a system property's value can break existing operations and flows.
- Become part of the context of a flow run when the run begins.
- Are not readily visible. System properties are visible in the Studio Debugger (in the **Context Inspector** under **System Properties**) and in the **Configuration\System Properties** folder.

In addition, creating an input when you create a flow automatically creates a flow variable of the same name as the input when the flow is run. Thus you can unwittingly create an empty flow variable of the same name as a system property, thus obtaining unexpected behavior.

- Are superseded by flow variables of the same name. If an input can get its value from a flow variable—that is, if the flow variable exists and has a value assigned to it—then the flow variable has priority over the system property as the source of the input's value.

On the other hand, the value of a system property cannot be changed by the assignment of an input value or result to the system property. When you assign a value from either of those two sources to a system property, you are really creating a flow variable with the same name as the system property and assigning the value to that flow variable.

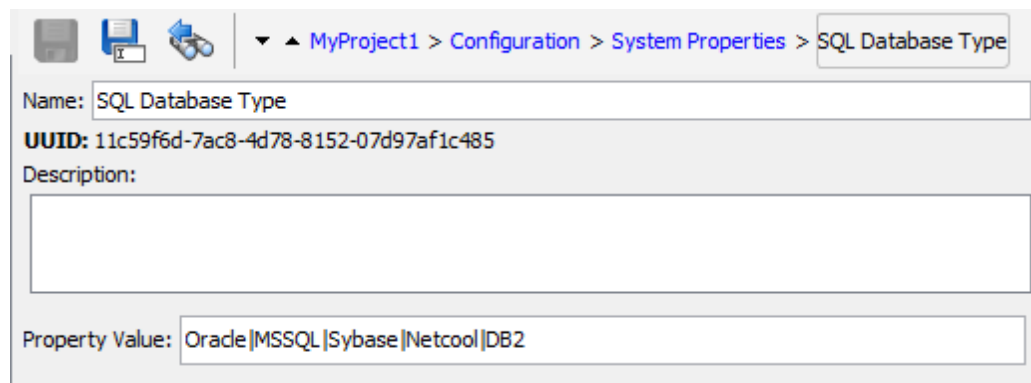
## What do you want to do?

### Create a system property

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click the **System Properties** folder, and then click **New**.
3. Enter a name for the system property and click **OK**.



The System Properties Editor opens.



The screenshot shows the 'System Properties Editor' window. The breadcrumb path is 'MyProject1 > Configuration > System Properties > SQL Database Type'. The 'Name' field is 'SQL Database Type'. The 'UUID' is '11c59f6d-7ac8-4d78-8152-07d97af1c485'. The 'Description' field is empty. The 'Property Value' field contains 'Oracle|MSSQL|Sybase|Netcool|DB2'.

4. (Optional) In the **Description** box, type a description of the system property.
5. In the **Property Value** box, type the values for the system property, using | as a separator.
6. Click **Save**.

The system property is saved in the **System Properties** folder and is now available for use in any flow.

## Use the value of a system property in a flow

1. Specify the system property as the data source for a flow or step input.
2. In a scriptlet, use the appropriate command to obtain the system property's value.

**Note:** For information on the required command and its syntax, on the **Scriptlet** tab of an operation, click **Insert Template**. The template provides the necessary commands for working with the global context. For more information, see ["Using Scriptlets in a Flow" on page 200](#).

## Change the value of a system property

There are several ways to change the value of a system property:

- Change the system property in a scriptlet. This changes the value from the point at which the script is run. For information on the required command and its syntax, on the **Scriptlet** tab of an operation, click **Insert Template**.
- Create an operation that sets the system property's value.
- Open the system property in the **Configurations\System Properties** folder, and change the value.

## Delete a system property

Before you delete a system property, it is recommended to use the **What Uses This** function to check that other items do not depend on it. For more information, see ["Finding Out How Flows and Operations are Used" on page 243](#).

1. In the **Projects** pane, expand the **Configuration** and **System Properties** folders.
2. Right-click the system property and select **Delete**.
3. Click **Yes** in the confirmation window.

## Reference Material

### System Properties Editor

▼ ▲ MyProject1 > Configuration > System Properties > SQL Database Type

Name: SQL Database Type

UUID: 11c59f6d-7ac8-4d78-8152-07d97af1c485

Description:

Property Value: Oracle|MSSQL|Sybase|Netcool|DB2

GUI item	Description
<b>Name</b>	The name of the system property.
<b>Description</b>	(Optional) Description of the system property.
<b>Property Value</b>	Type the values for the system property, using   as a separator.

## Authoring a Flow – Basics

A flow is a set of actions that are linked by decision-making logic, to automate tasks.

For example, you want to verify that a page on your Web site contains the correct, current data, such as a certain piece of text. If the desired data is not on the Web page, you want to push new content to the site. You can create a flow to do these tasks automatically.

This chapter covers all the basic steps that need to be done to create a simple flow. For information about creating more complex flows, see ["Advanced Authoring" on page 185](#).

Creating a Flow – Step-by-Step .....	99
Creating a New Flow .....	103
Creating Steps in a Flow .....	106
Adjusting the Appearance of a Flow .....	112
Modifying a Flow .....	115
Creating Input .....	120
Specifying the Input Source .....	127
Evaluating Input Data .....	134
Creating Transitions .....	135
Setting Responses .....	139
Creating Outputs and Results .....	149
Setting Operation Outputs .....	150
Setting Step Results .....	153
Filtering Output and Results .....	158
Working with Variables .....	175
Creating Return Steps .....	181

## Creating a Flow – Step-by-Step



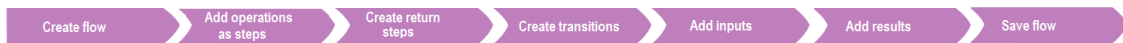
This topic takes you step-by-step through the major steps involved in creating a flow. It demonstrates how to create a simple flow that checks whether a page on a Web site contains a certain piece of text and, if that text is not found, posts a page to the Web site.

Note that this is just a high level look at the Studio workflow, and there are many options that are not described here. For more detailed information about any of the steps, use the links to learn about the flow authoring options in-depth.

This topic presents the steps in a suggested order, but the steps do not have to be completed in the order shown.

For best practices for creating a flow, see ["Best Practices" on page 23](#).

## Step 1: Create a flow



1. Open the project in which you want to create the flow.
2. Open the folder in which you want to create the flow.
3. Select **File > New > Flow**.
4. Type a meaningful name for the flow and click **OK**.

**Note:** Names can be a maximum of 128 characters long, and are not case-sensitive.

5. Open the flow in the authoring pane and click **Properties** (at the bottom of the pane), then click the **Description** tab.
6. Enter a description of the flow.

For more information and options, see ["Creating a New Flow" on page 103](#).

## Step 2: Add operations as steps

Drag an operation to the authoring pane, to use it as a step in your flow.

- If you want to modify the operation, copy and paste it into the **Projects** pane before dragging it onto the authoring pane.


**Note:** This is only recommended if you want to add responses or results, if only the inputs are used, then modify them inside the steps.

In our example, there are two steps. Step 1 uses the **Check Web Site** operation, which checks a Web page to see whether it contains specific text. Step 2 uses the **Post Page** operation to post a page to the Web site.

For more information and options, see ["Creating Steps in a Flow" on page 106](#).

## Step 3: Create return steps to end the flow

Create one or more return steps to end the flow. Return steps indicate four primary possible end states to the flow: **Success**, **Diagnosed**, **No action**, and **Failure**.

1. On the authoring pane toolbar, click the **Step Palette** button .
2. From the **Step** palette, drag the appropriate return step icons to the authoring canvas.
3. If required, change the flow responses that are assigned to the steps.

In our example flow, there are two end states: **Success** and **Failure**. There is no need to change the default flow responses.

For more information and options, see ["Creating Return Steps" on page 181](#).

## Step 4: Create transitions

Create connections between steps so that each response for a step takes the flow to another step or to an end step.

**Note:** Every response icon on the step must be connected to another step; otherwise, the flow will show errors.

1. On the step that you want to connect to the next step, click the icon that represents one of the



responses, and drag a line to the destination step for that response.

2. If you need to modify the transition, double-click the line to open the Transition Inspector.
3. Repeat the process for the other response icons on the step.

In our example:


- If the Web page cannot be found, the flow ends in failure. Drag from the red **Fail** response icon to the **Fail** end step.
- If the page is there but the text isn't present, go to the second step, of posting another page to Web site. Drag from the yellow response icon to the second step.
- If the page is there and the desired text is present, the flow ends in success. In both steps, drag from the green **Success** response icon to the **Success** end step.

For more information about transitions, see ["Creating Transitions" on page 135](#).

For more information about setting the responses for an operation, see ["Setting Responses" on page 139](#).

## Step 5: Add inputs to the flow

1. Double-click a step to display its details in the Step Inspector.

2. In the Step Inspector, click the **Inputs** tab.
3. Click **Add Input** and enter a name for the input.
4. Click in the row's **Type** column and select one of the value assignment types from the list:
  - Single Value
  - List of Values
5. To define the data source for the input, open the Input Editor by clicking the arrow  at the end of the input row.

In our example, the **Check Web Site** step needs to know which page to check (mysite.com/mypage.htm) and what text to look for ("needed text"). In this case, you would create two single constant value inputs.

For more information and options, see ["Creating Input" on page 120](#).

### Step 6: Add results to the flow

By adding a result, you can capture a step's output and save it as a flow variable (to be used in other steps in the flow) or a flow output field (to be passed to a parent flow).

1. In the Step Inspector, click the **Results** tab.
2. Click **Add Result** and enter a name for the result.
3. In the **Name** column, enter a name for the result, and press the Return key on your keyboard. This name will be used as the name of the flow variable or flow output field.
4. From the **From** list, select the source for the result. For example, select the primary output.
5. From the **Assign To** list, decide where the value will be stored: as a flow variable or output field.
6. From the **Assignment Action** list, select the appropriate action: overwrite, append, prepend, or one of the arithmetic assignment actions.

In our example, the **Check Web Site** step can be set to store the text that it found as a flow variable. In our simple two-step flow, the successful completion of the first step goes straight to the **Resolved: Success** end step. But it is possible to add another step, for example, a **Send Email** step, which includes the flow variable data in the body of the email.

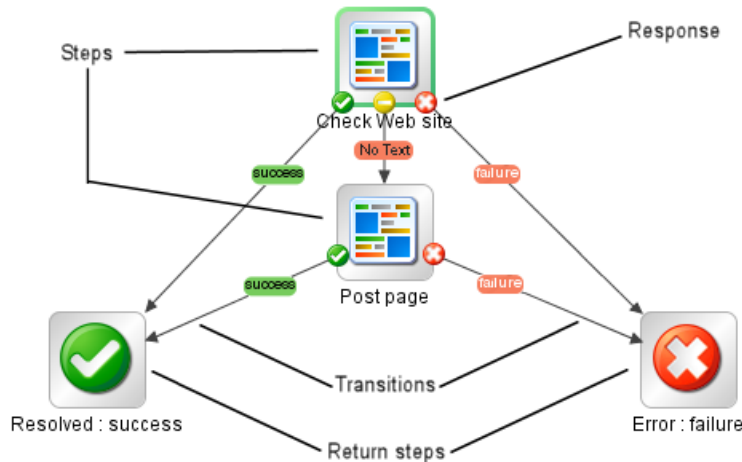
For more information and options, see ["Setting Step Results" on page 153](#).

### Step 7: Save the flow

Click the **Save** button.

Studio validates the flow. If a flow is not valid, it is saved and an error message appears.

The different flow elements are shown below:



## Creating a New Flow

There are two options for creating a flow:

- Create the flow from scratch
- Use a predefined template as the basis of the flow

## Best Practices

### Naming a Flow

- Make sure you plan the structure of the flow you want to create, before starting to build it. Think about whether you can break it down into multiple small flows that you can reuse.
- Use a self-explanatory name for the flow, which clearly describes the purpose of the flow.
- Use naming conventions for different flow types. For example, add prefixes to flow names, based on the flow type. We recommend using a document that describes the naming conventions and other guidelines for flow authors.
- Be consistent about case. For example, use title case for all flow names.
- For flows that run a single task, use a "<Verb> <Noun>" name format. For example, **Send Mail**, **Create Snapshot**.
- For sample flows, use the word "Sample" in the name. For example, **Send Mail Sample**, **Create Snapshot Sample**.
- For flows that check whether something is the case, use the question being answered as the name. For example, **Is Computer Account Enabled**.

- For health check flows that collect information about a system or environment, include "Health check" in the flow name (except for cases where you have a special Health Check folder). For example, **Solaris Health Check**.
- If you are working with an integration, keep the original flow names from the API being used.

## Flow Description

- When you create a flow, it is recommended to add a description of what the flow does, in the **Description** tab. The description should include search words to help you or others find the flow. The description should also tell users about the flow inputs, giving them hints about the kind of values to provide.
- The order in which inputs and outputs are listed in the description must be the same as the order in which they appear in the **Inputs / Outputs** tab.
- For more details about best practices for the description, see ["Best Practices" on page 23](#).

**Note:** You can use the Generate Documentation feature to make the information in the description available to authors and users. For more information, see ["Generating Documentation about Flow and Operations" on page 245](#).

## Create a flow from scratch

1. Open the project in which you want to create the flow.
2. Open the folder in which you want to create the flow.
3. Select **File > New > Flow**.
4. Type a meaningful name for the flow and click **OK**.

**Note:** Names can be a maximum of 128 characters long, and are not case-sensitive.

## Create a flow from a template

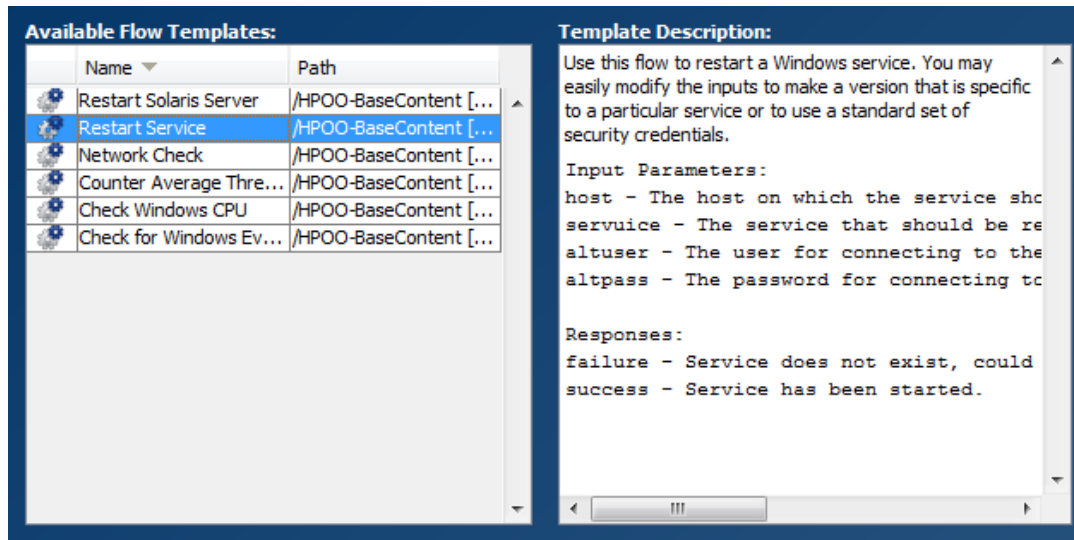
The templates provided with Studio provide flows that perform certain frequently used tasks. For example, there is a **Restart Service** template for creating a flow to restart a service.


1. Open the project in which you want to create the flow.



2. On the Welcome screen, click the **New Flow** button.
3. In the list of templates that appears, highlight a flow template to display its description. If required, drag the scroll bar to display the text.

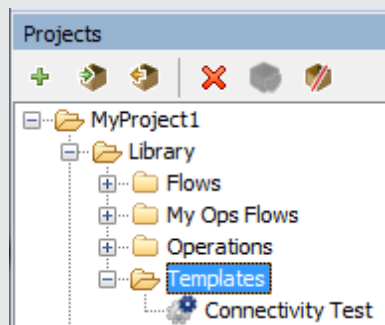




4. Select the flow template that meets your needs, and then click the **Create**  button.
5. The new flow is created in the **My Ops Flows** in the **Library** folder of the selected project. If you want to store the flow in a different folder, drag it to the desired folder, or use the **Edit > Cut** and **Edit > Paste** commands.

**Tip:** You can also get templates from the **Templates** folder in **OO-Base Content** content pack, in the **Content Packs** pane. Double-click the template to open it in the authoring pane. If you want to modify the flow, copy it into your project and modify the copy.

**Note:** You can create templates from other flows, by creating a **Templates** folder under **Library**, in your project, and storing flows in that folder. These flows will be displayed in the list of flow templates.



## Add a description of the flow

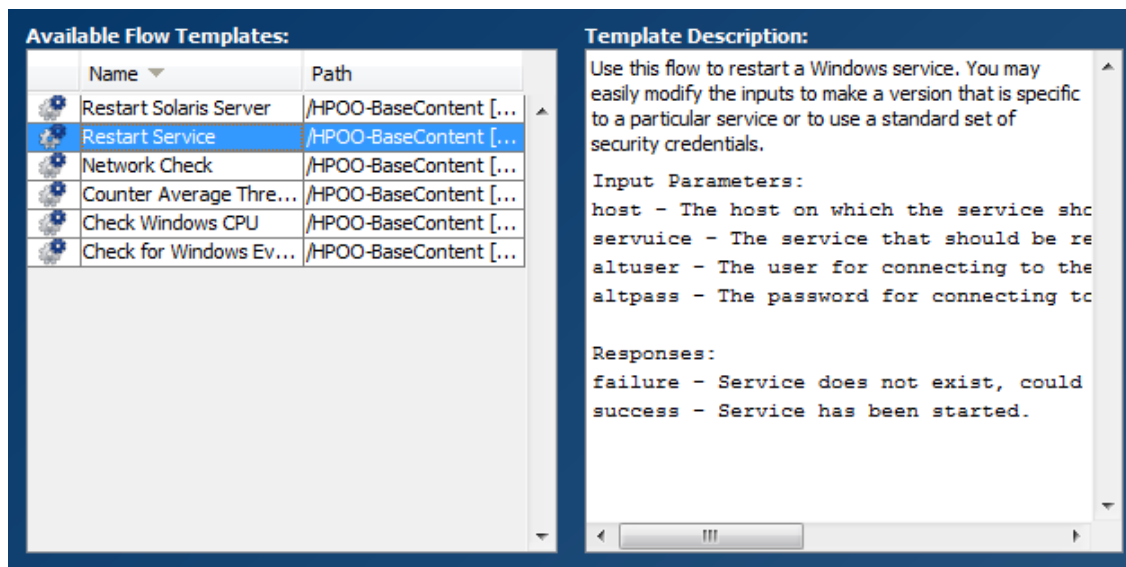
1. On the authoring pane, right-click the flow, and click **Properties**.

2. Click the **Description** tab.
3. Enter a description of the flow and click **OK**.

## Reference Material

### Available Flow Templates

When you click the **New Flow** button on the HP OO Welcome screen, the **Available Flow Templates** list is displayed.



GUI item	Description
<b>Name</b>	Displays the names of the available templates. Highlight a name to display its description in the <b>Template Description</b> box.
<b>Path</b>	Displays the path to the folder where each template is stored.
<b>Template Description</b>	Displays a description of the selected template.

## Creating Steps in a Flow

When you create a step from an operation, the step is an instance of the operation and so inherits the operation's inputs, outputs, references, and other characteristics.

To create a step from an operation, drag the operation to the authoring pane.

- If you drag an operation from the **Content Packs** pane, you will be able to modify the step, but note that the operation in the **Content Packs** pane is read-only.

- If you want to modify an operation before creating the step, copy it from the **Content Packs** pane and paste it into the **Projects** pane, before dragging it onto the authoring pane.

## Best Practices

- When you create a step, it is recommended to add a description of the operation or flow from which the step was made, in the **Description** tab. The description should include search words to help you or others find your step, and should tell users about the step inputs, responses, and results. For more details about best practices for the description, see ["Best Practices" on page 23](#).
- The start step should be located in the upper-left corner of the flow, with the following exceptions:
  - The start step has many responses, each of which leads to another step.
  - Placing the start step in the upper-left corner would cause excessive visual complexity, such as the crossing of transitions.
- If you are renaming a step, make sure that the name clearly describes the purpose of the step.
- Consider using callouts to provide information about a step.
- If you don't need to customize the properties of an operation, use the original read-only version for the step rather than a copy.
- If you are working with an integration, keep the original operation names from the API being used.

## What do you want to do?

### Create a step from an operation

1. In the **Projects** pane or **Content Packs** pane, select the operation that you want to add to the flow.

**Note:** The operations in the **Content Packs** pane are read-only.

2. Drag the operation from the project tree to the authoring pane.
3. If required, rename the step to reflect its function within the flow (operation names may be too generic):
  - a. Right-click the step that you want to rename and select **Rename**.
  - b. Type the new name in the highlighted field.

4. If required, edit the step. For more information, see ["Modifying a Flow" on page 115](#).

## Copy a read-only operation into the project to make it editable

1. In the **Content Packs** pane, select the operation that you want to copy.
2. Select **Edit > Copy**.
3. Select the location in the project tree where you want to paste the copy, and select **Edit > Paste**. This operation is treated as a new object and is detached from the content pack it arrived with.
4. If required, edit the operation.



**Note:** If you edit the operation, any steps that were created from this operation inherit the changes to the properties. If you edit the step, this does not affect the original operation.

5. Drag the operation from the project tree to the authoring pane.

**Note:** A operation that is copied from a content pack to a project is a "soft copy". This means that if the operation was originally created by importing an action plugin, the copied operation continues to reference the original operation. If the action plugin is upgraded and the original operation is updated to call the new version, the copied operation is updated automatically. For more information, see ["Creating Operations" on page 227](#).

## Copy a step from within the flow

To copy and paste a step, use any of the following tools:

- The **Copy**  and **Paste**  buttons on the authoring pane toolbar
- The **Edit > Copy** and **Edit > Paste** menu commands
- The right-click menu
- Keyboard combinations (CTRL+C, CTRL+V)

## Give a step a description

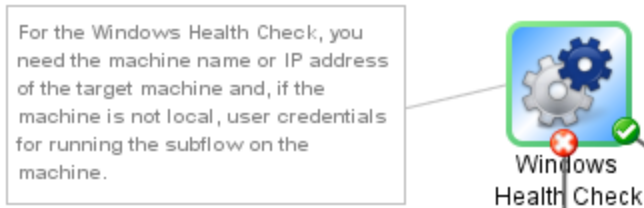
1. On the authoring pane, right-click a step, and click **Properties**.
2. Click the **Description** tab.
3. Enter a description of the step and click **OK**.




For best practices for writing a description, see ["Best Practices" on page 23](#).

## Give a step a callout

Callouts contain information about a step. They can greatly enhance the usability of a flow, by providing information such as:

- Data movement: how information is passed from one step to another
- Names of flow variables that store data
- Formats required for the input data



1. Display the **Step** palette by clicking the **Step Palette** button  from the authoring pane toolbar.
2. Click the **Callout**  button and drag the callout onto the authoring pane.
3. Type the text for the callout.
4. To connect the callout to a step, drag from the gray circle  to the step.
5. Drag the corners of the callout text area to resize it.

## Create a step from a flow (subflow)

A subflow is a flow within a flow. For more information, see ["Creating a Subflow Within a Flow" on page 185](#).

1. Open the parent flow in the authoring pane.
2. In the Library, select the flow that you want to use as a step (or subflow).
3. Drag the flow from the Library to the parent flow in the authoring pane. The flow that you dragged becomes a step in the parent flow.

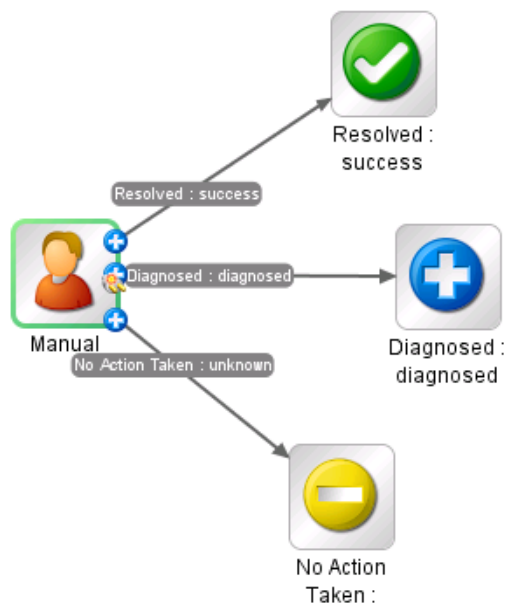
**Note:** The **Accelerator Packs** folder contains flows that can run as parent flows.

## Create a manual step

A manual step is one that offers a choice of actions. The user will need to select an action at runtime.

To create a manual step, you copy the manual operation template from the base content and define the actions that will be made available to the user.

1. In the **Content Packs** pane, select the manual operation template .
2. Select **Edit > Copy**.
3. Select the location in the project tree where you want to paste the copy, and select **Edit > Paste**. This operation is treated as a new object and is detached from the content pack it arrived with.
4. Drag the operation from the project tree to the authoring pane.
5. In the step, add the actions that will be available to the user.



**Note:** It is also possible to add the actions in the operation properties, rather than in the step. If you do this, you will be able to use the operation in other flows.

## Create a display step

A display step is one that displays information in a pop-up prompt message, but does not perform any other action. The user will just need to click **Continue** at runtime.

To create a display step, you copy the display operation template from the base content and define the information that will be displayed to the user.

The prompt message can include variables. For example, to tell the user what time the preceding step concluded, you could include a date/time variable (`${dateTime}`) in the message.

1. In the **Content Packs** pane, select the display operation template .
2. Select **Edit > Copy**.
3. Select the location in the project tree where you want to paste the copy, and select **Edit > Paste**. This operation is treated as a new object and is detached from the content pack it arrived with.
4. Drag the operation from the project tree to the authoring pane.
5. Open the Step Inspector for the step and click the **Display** tab.
6. Select the **Always prompt user before executing this step** check box.
7. In the **Prompt Title** box, type the prompt's label.
8. In the **Prompt Width** box, type the width of the prompt in pixels.
9. In the **Height** box, type the height of the prompt in pixels.
10. In the **Prompt Text** box, type a message to the user.
11. Click **OK**, and save your changes.

**Note:** It is also possible to add the display information in the operation properties, rather than in the step. If you do this, you will be able to use the operation in other flows.

## Reference Material

### Step Inspector > Display tab

In the **Display** tab of the Step Inspector, you can create a user prompt that is displayed to the user.

Step Name:

☐ Always prompt user before executing this step

Prompt Title:


Prompt Width:  Height:

Prompt Text:

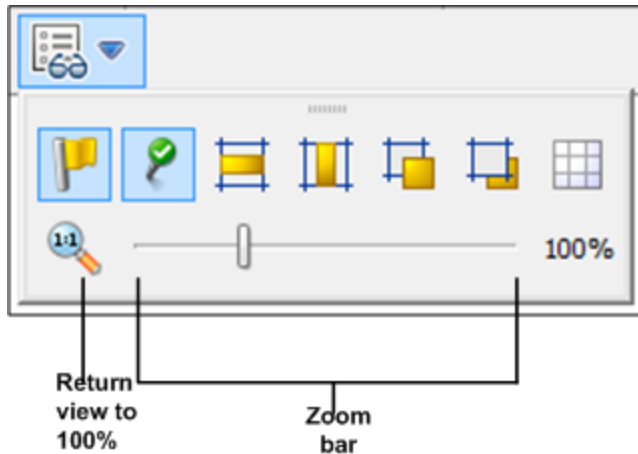
GUI item	Description
<b>Always prompt user before executing this step</b>	Select the checkbox if you want the prompt window to appear every time this step is run.
<b>Prompt Title</b>	Type the label that will appear in the title bar of the prompt window.
<b>Prompt Width</b>	Type the width of the prompt window in pixels.
<b>Height</b>	Type the height of the prompt window in pixels.
<b>Prompt Text</b>	Type the message that will appear in the body of the prompt window. You can include variables in the message. For example, <code>\${dateTime}</code> .

## Adjusting the Appearance of a Flow

When creating a flow, you can use the **View Options** palette to neaten up the flow and adjust its appearance on the authoring pane.

Display the **View Options** palette by clicking the **View Options** button  from the authoring pane toolbar.








## What do you want to do?

### Snap steps to the grid

Snapping objects to the grid is a quick way to keep them aligned and neat.



1. If the grid is not visible in the background of the authoring pane, click the **Show/Hide Grid** button  in the **View Options** palette. When you drag an operation to the authoring pane, it snaps to the nearest line in the grid.
2. To move a step from one line in the grid to another, move the step slightly and release the mouse.

### Align steps

1. To align selected steps horizontally, select one or more steps and then select **Align Selection Horizontally**  in the **View Options** palette.
2. To align selected steps vertically, select one or more steps and then select **Align Selection Vertically**  in the **View Options** palette.



### Show or hide response labels and icons

If your flow is looking overcrowded because of response labels and icons on operations, you can choose to hide these.

1. To show or hide response labels, click the **Show/Hide Labels**  button to toggle between hiding and showing the response labels.
2. To show or hide the response icons, click the **Show/Hide Connected Response Icons**  button to toggle between hiding and showing the response icons.

## Move objects to the front or back

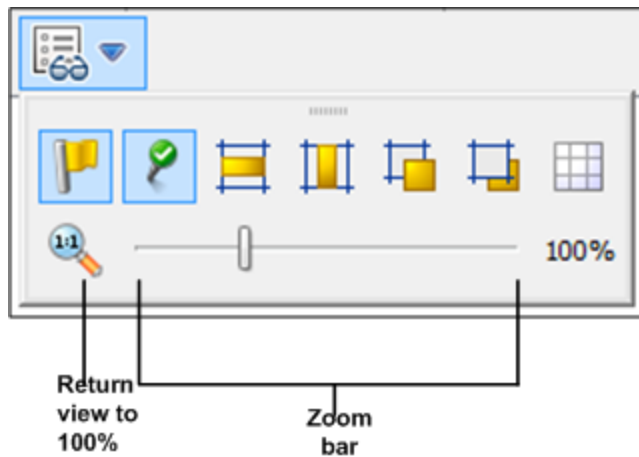
Flows may include objects that are stacked together in the authoring pane. This may occur in long flows where there are many items in the flow. In such cases, you need to bring the most important objects to the front of the stack.






1. To move an object to the front of the stack, select it and click **Bring to Front** .
2. To move an object to the back of the stack, select it and click **Send to Back** .



## Reference Material

### View Options palette

The **View Options** palette contains buttons for changing the appearance of the flow on the authoring pane.



Button	Description
<b>Show/Hide Labels</b> 	Shows or hides response labels on objects
<b>Show/Hide Connected Response Icons</b> 	Shows or hides response icons on objects
<b>Align Selection Horizontally</b> 	Aligns selected steps horizontally
<b>Align Selection Vertically</b> 	Aligns selected steps vertically
<b>Bring to Front</b> 	Moves the selected object to the front of the stack

Button	Description
<b>Send to Back</b> 	Moves the selected object to the back of the stack
<b>Show/Hide Grid</b> 	Reveals the authoring pane grid, which you can use for arranging steps. When you stop dragging a step, it snaps to the nearest position on the grid.

## Modifying a Flow

After a flow has been created, you can modify the flow. For example, you might copy a flow that you created earlier and adapt it for a slightly different use. Or you might take one of the out-of-the-box flows provided in HP OO, such as those in the **Accelerator Packs** folder, and adjust it for your needs.

## Best Practices

Always make a copy of a flow before modifying it.

Before making changes to a flow, use **References > What uses this?** to check whether other flows use it.

If you are copying a flow and you think that you may need to modify the properties of the operations, it is best to use the **Copy Deep** command. This makes a copy of the operations along with the flow, so that you can modify them without affecting the originals. See ["Copying Flows and Operations" on page 240](#).

If you copy a flow using the **Copy Deep** command, create a new folder for the flow and its operations.

**Caution:** Make sure that you understand the difference between modifying a step and modifying an operation.

- When you modify the properties of a step (in the Step Inspector), this only affects the individual step.
- When you modify the properties of an operation (in the **Properties** sheet), this affects all the flows that use this operation as a step. You need to be extremely careful about modifying an operation's properties—doing so can break other flows that use it.

## What do you want to do?

### Change the start step

If you add a new step at the start of a flow, it will appear with a warning icon, because the start step has not been defined.

Right-click the step that you want to use to start the flow and select **Set Start Step**.

### Rename a step

1. Right-click the step that you want to rename and select **Rename**.
2. Type the new name in the highlighted field, press the ENTER key, and save your work.

**Best practice:** Make sure that the name clearly describes the purpose of the step.

### Rename a flow or operation

If you are renaming an operation, check whether this operation is used in other flows. If so, it is better to make a copy of the operation and rename the copy.

1. In the **Project** pane, right-click the flow or operation that you want to rename and select **Rename**.
2. Type the new name in the highlighted field, press the ENTER key, and save your work.

**Best practice:** Make sure that the name clearly describes the purpose of the flow or operation.

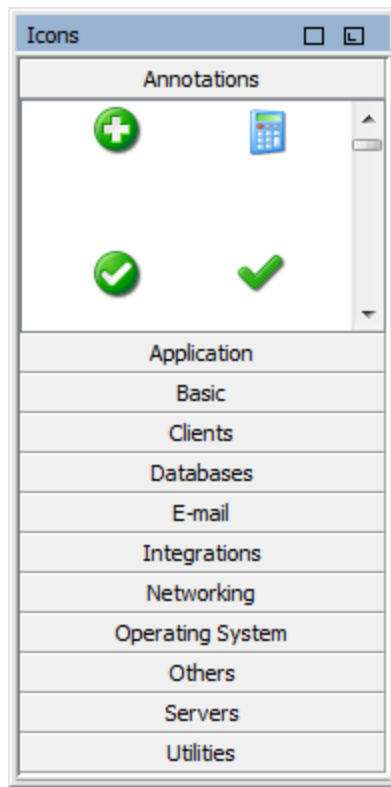
### Move a group of steps in the flow diagram

1. Hold down the SHIFT or CTRL key to select a group of steps.
2. Click and drag the steps as a group.

### Change an icon

You can change the icon on a step, operation, or flow to one that gives a clearer visual cue to what the element does.

1. To open the **Icons** pane, click the **Icons** tab.



2. Select the icon group name that describes the icon you need.
3. Select the icon, and drag it onto the step, operation, or flow.


**Note:** By holding down the CTRL key, and then dragging an icon, you can place the icon on top of an existing icon, in layers.

**Best practice:** If you have classified flows according to type, use specific icons for each flow type.

## Modify a step in the Step Inspector

1. Right-click a step and select **Properties**. The Step Inspector opens.
2. In the Step Inspector, you can modify the step:
  - Add or edit inputs in the step. For details, see ["Creating Input" on page 120](#).
  - Add or edit results in the step. For details, see ["Setting Operation Outputs" on page 150](#).

- Add or edit a description of the step. For details, see ["Creating Steps in a Flow" on page 106](#).
- Add or edit a user prompt for the step. For details, see *Display a user prompt for the step* below
- Add or edit a scriptlet in the step. For details, see ["Using Scriptlets in a Flow" on page 200](#).

**Tip:** To keep the Inspector open so that you can shift its focus from step to step without having to close and reopen the Inspector, click the **Pin** button  at the right end of the Inspector's title bar.

## Display a user prompt for the step

You can create a user prompt that is displayed before a step is run. The prompt message can include variables. For example, to tell the user what time the preceding step concluded, you could include a date/time variable (`${dateTime}`) in the message.

1. Right-click a step and select **Properties**.
2. Click the **Display** tab in the Step Inspector.
3. Select the **Always prompt user before executing this step** check box.
4. In the **Prompt Title** box, type the prompt's label.
5. In the **Prompt Width** box, type the width of the prompt in pixels.
6. In the **Height** box, type the height of the prompt in pixels.
7. In the **Prompt Text** box, type a message to the user.
8. Click **OK**, and save your changes. The step acquires a blue arrowhead, indicating the display prompt.



## Change which operation a step is based on

For example, you need an existing flow step to be associated with a different operation, but you want to keep the existing transitions to and from that step.

1. Right-click a step and select **Properties**.
2. In the Step Inspector, click the **Advanced** tab.

3. Under **Source Operation**, click the **Select** button.
4. In the Select Source Operation dialog box, navigate to and select the operation that you want to base the step on, and then click **OK**.
5. Rename the step to reflect the change in the operation.
6. Review and make any changes necessary to the value assignments for inputs, to reflect any differences between the old operation's inputs and those of the new one.

## Reference Material

### Step Inspector > Display tab

In the **Display** tab of the Step Inspector, you can create a user prompt that appears before a step is run.

The screenshot shows the 'Step Inspector' window with the 'Display' tab selected. The 'Step Name' is 'SQL Query'. Below the tabs (Inputs, Results, Display, Description, Advanced, Scriptlet), there is a checkbox labeled 'Always prompt user before executing this step'. Below this, there are input fields for 'Prompt Title', 'Prompt Width' (set to 0), and 'Height' (set to 0). At the bottom, there is a large text area labeled 'Prompt Text'.

GUI item	Description
<b>Always prompt user before executing this step</b>	Select the checkbox if you want the prompt window to appear every time this step is run.
<b>Prompt Title</b>	Type the label that will appear in the title bar of the prompt window.
<b>Prompt Width</b>	Type the width of the prompt window in pixels.
<b>Height</b>	Type the height of the prompt window in pixels.

<b>Prompt Text</b>	Type the message that will appear in the body of the prompt window. You can include variables in the message. For example, <code>\${dateTime}</code> .
--------------------	--

## Creating Input

Inputs specify how and when the steps in a flow obtain the data that they need. For example, in a **Network Check** flow, the first step pings a server, so it needs the IP address of the server to ping. The IP address is provided via an input.

Each input is mapped to a variable, whose value can be set in the following ways:

- Create a user prompt, so that the value will be entered by the person running the flow, at the start of the flow.
- Set the input's value to a specific, unchanging value.
- Set the value to be obtained from another step.
- Assign a flow variable to the input. A flow variable is of a collection of variables and data values that are available to the entire flow.

You can create an input for a flow, operation, or step.

**Caution:** Make sure that you understand the difference between modifying a step and modifying an operation.

- When you modify the properties of a step (in the Step Inspector), this only affects the individual step.
- When you modify the properties of an operation (in the **Properties** sheet), this affects all the flows that use this operation as a step. You need to be extremely careful about modifying an operation's properties—doing so can break other flows that use it.

## Best Practices

- Define all possible inputs in the flow description, while differentiating between optional and required inputs.
- Delete optional inputs in steps, if these are not required.
- Create user selection lists for input wherever possible. These help prevent errors based on typing mistakes.
- Be consistent about case. For example, use camel case for all input names.
- If you are working with an integration, keep the original input names from the API being used.



- Do not disable a required input in an operation, because it will disable it in all instances of that operation; you should modify the individual step.
- Add inputs in a consistent order, according to ordering rules. For example:
  - By intuitive or logical grouping
  - By importance (required inputs first)
  - In alphabetic order

## What do you want to do?

### Create an input

1. Open the **Properties** sheet or Step Inspector:
  - To add an input to an operation, right-click the operation in the **Project** pane and select **Properties**.
  - To add an input to a flow, right-click the flow in the **Project** pane and select **Properties**.
  - To add an input to a step, double-click the step in the authoring pane.
2. Select the **Inputs** tab and click the **Add Input** button.


Input	Required	Type	From
keyName	<input checked="" type="checkbox"/>	Single Value	Value: finalResult
myValue	<input checked="" type="checkbox"/>	Single Value	Value: A number of \${ports} \${portType} ports were scanned on ...
Field_1	<input type="checkbox"/>	Single Value	Value: \${keyName}
Field_2	<input type="checkbox"/>	Single Value	Value: \${myValue}

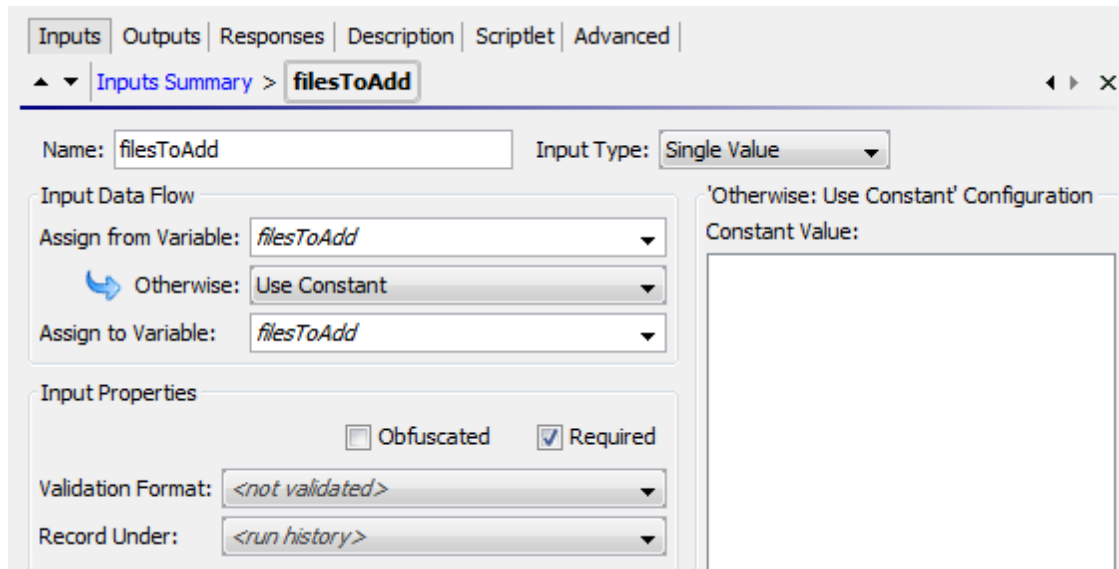
3. Enter the name of the new input and click **OK**. The input appears in a new row.

**Note:** Do not name the input “service” or “sp”. Doing so can create errors in flow runs in certain situations.

4. (Optional) To make the input mandatory for the step to function, select the **Required** check box on the new row.
5. From the **Type** list, specify how the input gets its value:

- **Single Value**
- **List of Values** - This enables you to run an operation against multiple targets.

6. Click the right-pointing arrow  at the end of the row to open the Input Editor.



The screenshot shows the 'Input Editor' for a variable named 'filesToAdd'. The 'Input Type' is set to 'Single Value'. Under 'Input Data Flow', 'Assign from Variable' is set to 'filesToAdd', and 'Assign to Variable' is also set to 'filesToAdd'. The 'Otherwise' dropdown is set to 'Use Constant'. Under 'Input Properties', 'Obfuscated' is unchecked and 'Required' is checked. 'Validation Format' is set to '<not validated>' and 'Record Under' is set to '<run history>'. On the right, there is a section for 'Otherwise: Use Constant' Configuration with a 'Constant Value' text area.

7. Specify the input source in the Input Editor:

- To assign the value from a variable with the same name as the input, accept the default name that appears in the **Assign from Variable** box.
- To assign the value from a different flow variable, enter the variable name in the **Assign from Variable** box.

For example, you might have a first step that looks for a piece of information, and which saves this information as a flow variable. Then, you might have a second step that displays that information. The second step uses the flow variable that was created in the first step.

8. From the **Otherwise** list, select the action to occur if the flow variable that you specified in the **Assign from Variable** box does not exist or has no value stored in it. The options are:

- **Prompt User** - Set up a prompt for the user to supply the information, either by entering information or making a selection from a list, at the start of the flow.

**Note:** Do not select the **Prompt User** option for a step in the middle of a flow. User prompts in the middle of a flow are not supported as an input source in this version.

- **Use Constant:** Enter the constant value that will be used for the input. For example, an IP address that is always used.
- **Use Previous Step Result:** Select a previous step result to be used in the case that this input has no value.
- **System Account:** Enter the system account name and credential type.
- **Logged-in user credentials:** Enter for the logged in user, user name or password.

For more details about different types of input source, select the relevant task in "[Specifying the Input Source](#)" on page 127.

9. By default, Studio creates a flow variable with the same name as the input. This variable can be used in later steps in the flow. It is possible to change this name in the **Assign to Variable** box.

For example, if you have a step in which the user needs to enter a password, you might want to name the variable `password` to make it easy to identify.

10. (Optional) To obfuscate the input's value, select the **Obfuscated** check box. The input will appear as a row of asterisks when the flow is run.

## Remove an input

1. Open the **Properties** sheet (for a flow or operation) or the Step Inspector (for a step).
2. On the **Inputs** tab, select the input that you want to remove, and then click **Remove Input**. A default input that was removed appears in gray italics.

## Restore a default input that was removed

If you removed a default input from the **Inputs** tab, you can restore it. Default inputs are those that were created as part of the operation on which the step is based. Default inputs that have been removed appear in the list of inputs grayed out and italicized.

1. On the **Inputs** tab, click **Add Input**.
2. Type the exact name of the input that you want to restore.
3. Click **OK**.

## Disable a required input

Some operations have required inputs, and it is not possible to clear the **Required** check box. However, if you don't need an input in a step, you can disable it.

**Caution:** Do not disable a required input in an *operation*, because it will disable it in all instances of that operation; you should modify the individual *step*.

1. In the step, open the Input Editor for the required input.
2. From the **Otherwise** list, select **Use Constant**, but leave the '**Otherwise: Use Constant**' **Configuration** section empty. The input is now disabled.




## Reference Material

### Step Inspector > Inputs tab

The **Inputs** tab in the Step Inspector is where you specify how and when a step in a flow obtains the data that it needs.

The screenshot shows the 'Inspector' window with the 'Inputs' tab selected. The 'Step Name' is 'Set Inactive Users By Group'. Below the tabs, there's an 'Inputs Summary' section with buttons for 'Add Input', 'Remove Input', and arrows for moving inputs up and down. A table lists the inputs with columns: 'Assign To Input', 'Required', 'Type', and 'From'.


Assign To Input	Required	Type	From
host	<input checked="" type="checkbox"/>	Single Value	Value: \${dnsServer}
baseDN	<input checked="" type="checkbox"/>	Single Value	Value: \${domainDN}
DN	<input checked="" type="checkbox"/>	Single Value	Value: CN=Users,\${domainDN}
username	<input type="checkbox"/>	Single Value	Value: \${altuser}
password	<input type="checkbox"/>	Single Value	Value: *****
daysOld	<input checked="" type="checkbox"/>	Single Value	Value: 30
groupDN	<input checked="" type="checkbox"/>	Single Value	Value: CN=Domain Admins,CN=Users,\${domainDN}



GUI item	Description
<b>Name</b>	Displays the name of the step (read only).
<b>UUID</b>	Displays the unique identifier of the flow or operation (read only).
<b>Add Input</b>	Adds a new input row.
<b>Remove Input</b>	Deletes the selected input row.
<b>Move Up</b> 	Moves the selected input row higher in the list, so that it is processed earlier in the run.
<b>Move Down</b> 	Moves the selected input row lower in the list, so that it is processed later in the run.
<b>Input Editor</b> 	Displays the Input Editor for the input in the row.
<b>Input column</b>	Displays the name of the input.
<b>Required column</b>	Makes this input mandatory.

<b>Type column</b>	<p>From the <b>Type</b> list, specify how the input gets its value. The choices are:</p> <ul style="list-style-type: none"> <li>• <b>Single Value</b></li> <li>• <b>List of Values</b> - to run an operation against multiple targets</li> <li>• <b>Credentials</b> - to get the input data from system or user credentials</li> </ul>
<b>From column</b>	Specify where the input gets its value from.

## Properties sheet > Inputs tab

The **Inputs** tab in the **Properties** sheet is where you specify how and when a flow or operation obtains the data that it needs.

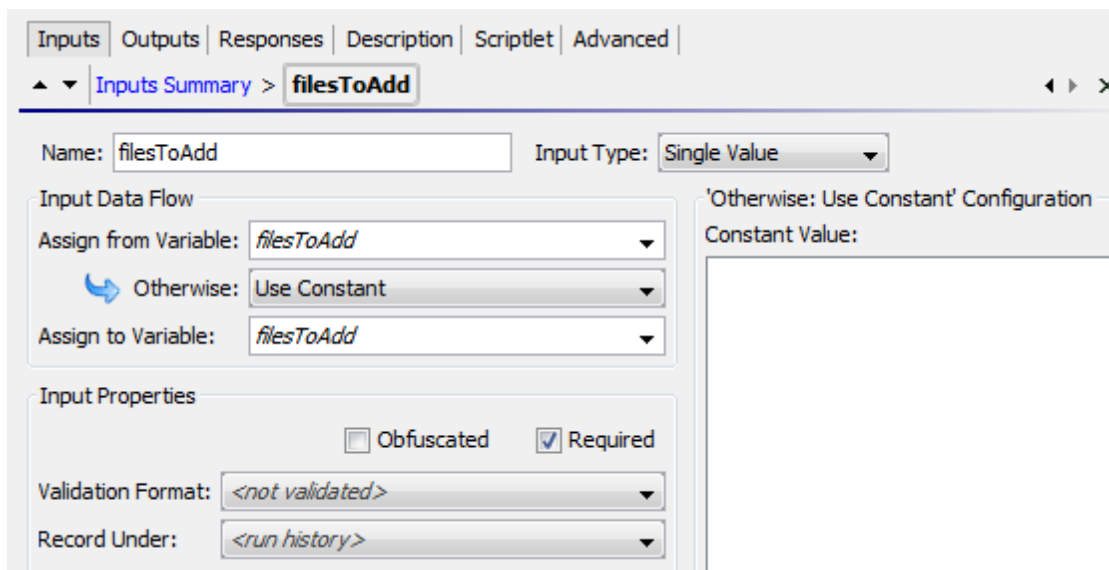
GUI item	Description
<b>Name</b>	Displays the name of the flow or operation (read only).
<b>UUID</b>	Displays the unique identifier of the flow or operation (read only).
<b>Assign Categories</b>	Opens the Assign Categories dialog box, enabling you to assign a category to a flow.
<b>Add Input</b>	Adds a new input row
<b>Remove Input</b>	Removes the selected input row
<b>Move Up</b> 	Moves the selected input row higher in the list, so that it is processed earlier in the run.

<b>Move Down</b> 	Moves the selected input row lower in the list, so that it is processed later in the run.
	Displays the Input Editor for the input in the row.
<b>Input column</b>	Displays the name of the input.
<b>Required column</b>	Makes this input mandatory.
<b>Type column</b>	From the <b>Type</b> list, specify how the input gets its value. The choices are: <ul style="list-style-type: none"> <li>• <b>Single Value</b></li> <li>• <b>List of Values</b> - to run an operation against multiple targets</li> </ul>
<b>Template column</b>	Specify where the input gets its value from.

## Input Editor



The Input Editor is where you specify the details of the input source, after you have set up the basic parameters of the input in the **Inputs** tab of the Step Inspector (for a step) or the **Properties** sheet (for a flow or operation).

To display the Input Editor, click the right-pointing arrow  at the end of an input row.



The screenshot shows the 'Input Editor' for an input named 'filesToAdd'. The interface includes tabs for 'Inputs', 'Outputs', 'Responses', 'Description', 'Scriptlet', and 'Advanced'. The 'Inputs' tab is active, showing the 'filesToAdd' input. The 'Name' field is 'filesToAdd' and the 'Input Type' is 'Single Value'. Under 'Input Data Flow', 'Assign from Variable' is 'filesToAdd', 'Otherwise' is 'Use Constant', and 'Assign to Variable' is 'filesToAdd'. Under 'Input Properties', 'Obfuscated' is unchecked and 'Required' is checked. 'Validation Format' is '<not validated>' and 'Record Under' is '<run history>'. On the right, there is a section for 'Otherwise: Use Constant' Configuration with a 'Constant Value' field.

GUI item	Description
<b>Name</b>	Displays the name of the input. May be modified here.
<b>Input Type</b>	Displays the input type. May be modified here.

<b>Assign from Variable</b>	Enter or select the name of the flow variable that will be the source of the input.
<b>Otherwise</b>	Select what will occur if the flow variable specified in the <b>Assign from Variable</b> box does not exist or has no value stored in it.
<b>'Otherwise: &lt;action&gt;' Configuration</b>	Configure the details of what happens if the flow variable specified in the <b>Assign from Variable</b> box does not exist or has no value stored in it. This section varies, depending on which action is selected in the <b>Otherwise</b> list.
<b>Assign to Variable</b>	Select the flow variable that you want to assign the input's value to.
<b>Obfuscated</b>	Obfuscate the input's value, so that it appears as a row of asterisks when the flow is run.
<b>Required</b>	Makes this input mandatory.
<b>Validation Format</b>	Validates the input's value with a system evaluator. This functionality is not currently supported.
<b>Record Under</b>	Makes the value available for diagnostics or auditing. This functionality is not currently supported.
<b>Arrow buttons</b> 	If there are multiple inputs open in the Input Editor, click the vertical arrow buttons to navigate between them.
<b>Arrow buttons</b> 	Click the horizontal arrow buttons to navigate between the Input Editor and the Inputs Summary.

## Specifying the Input Source

While you are setting up an input in a flow, operation, or step, there are a number of options available for how to specify the input source. The Input Editor looks different, depending on your selections.


The tasks in this section are optional sub-tasks within the main task of setting up an input. See ["Creating Input" on page 120](#).

**Note:** User prompts in the middle of a flow are not available as an input source in this version.

### *What do you want to do?*


#### **Specify the input source as a single constant value**

Specify the input value source as a static value. For example, a single constant value could be an IP address that is always used in a step.

1. Create an input and set the type to **Single Value**.
2. Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.
3. By default, the name that you gave to the input appears in the **Assign from Variable** box, because Studio assumes that there is a variable with the same name, which will be the source of the input value.
  - To assign the value from a variable with the same name as the input, accept the default name that appears in the **Assign from Variable** box.
  - To assign the value from a variable with a different name from the input, enter the variable name in the **Assign from Variable** box.
4. From the **Otherwise** list, select **Use Constant**. This defines what will occur if the flow variable that you specified in the **Assign from Variable** box does not exist or has no value stored in it.
5. In the '**Otherwise: Use Constant**' Configuration section, type the value for the input (for example, `false`). You can also use a combination of text and variable reference, using the following format: `${variablename}`. For example, `Ping of ${targethost} succeeded`.

You can also use credentials as an input source lets you enable a flow to perform tasks that require system account credentials.

Create an input.

Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.  
From the **Otherwise** list, do one of the following:

1. Select **Logged-in User Credentials**. The logged-in user is considered to be the user account under which the flow is started.
2. Select **System Account**, and then from the **Account Named** list, select the system account to use for the operation's credentials, and choose the property you want to extract from the system account (for example, password/username). This allows the flow to perform tasks that require these account credentials, while protecting the credentials from exposure by keeping them hidden behind the system account name. For more information about system accounts, see ["Configuring System Accounts" on page 88](#).

**Note:** Do not select the **Prompt User** option. User prompts are not supported in this version.


## Specify the input source as a single user-entered text

Specify the input value source as user-entered text when the user must supply the information necessary for the flow to do its work. For example, you might want the user to specify the IP address of their own server, at the start of the flow.

**Note:** Do not specify the input source as user-entered text for a step in the middle of a flow.




User inputs in the middle of a flow are not supported in this version.

1. Create an input and set the type to **Single Value**.
2. Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.
3. From the **Otherwise** list, select **Prompt User**.
4. In the '**Otherwise: Prompt User**' Configuration section, select **Text**.
5. In the **User Message** box, type a prompt message to let the user know what kind of data is needed.

### Specify the input source as a single user selection

Another way to get the user to supply the input is to present a list from which the user must make a selection. For example, you might want the user to select from a choice of locations, at the start of the flow.

**Note:** Do not specify the input source as user selection for a step in the middle of a flow. User inputs in the middle of a flow are not supported in this version.

1. Create an input and set the type to **Single Value**.
2. Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.
3. From the **Otherwise** list, select **Prompt User**.
4. In the '**Otherwise: Prompt User**' Configuration section, select **Selection**.
5. From the **List Source** list, select one of the following:

- **Selection List** - select from a set of predefined lists.

From the **Named** list, select the list you want to present to the user.

**Tip:** You can add to the set of predefined lists by creating a list. For information on creating a list, see *Creating selection lists for user prompts*.

- **Domain Term** - Domain terms are specialized selection lists. For example, to specify that a flow runs against certain classes of servers, you can add domain terms for the various kinds of servers in your system and create a user prompt in which the user selects the classes of servers that you want to run the flow against.

From the **Named** list, select the domain term list you want to present to the user.

- **Flow Variable** - create a list that is populated by the contents of a flow variable.


From the **Named** list, type or select the flow variable that contains the list.

In the **Source Delimiter** box, type the character that separates the elements in the list.

6. In the **User Message** box, type a prompt message to let the user know what kind of data is needed.

## Specify the input source as a constant list of values

A list of values for a static input enables you to run a step on multiple targets. For example, to run an operating system health check or install a software update on multiple machines.

1. Create an input and set the type to **List of Values**.
2. Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.
3. Under **Input Properties**, in the **Input Delimiter** box, type the character that separates the elements in the list.
4. From the **Otherwise** list, select **Use Constant**.
5. In the '**Otherwise: Use Constant**' **Configuration** section, in the **Constant Value** box, do one of the following:
  - Type the values for the input, separating between the values with the character that you entered in the **Input Delimiter** box.
  - Type one or more flow variable references, using the following format:

```
${flowvariablename1}<delimiter>${flowvariablename2}
```


**Note:** It is possible to include both typed values and variables in the same list. For example, `${flowvariableA}|${flowvariableB}|10.2.0.200|18.35.100.7`

In this example, **flowvariableA** contains 220.220.3.9 and **flowvariableB** contains 10.51.110.12 and the delimiter is set to the character "|". If you type the two variable names and type the IP addresses of two others manually into the **Constant Value** box, the operation will run on all four machines: 220.220.3.9, 10.51.110, 1210.2.0.200, and 18.35.100.7.

## Specify the input source as list of input values obtained from user-typed text

In this type of input, the user needs to type a list of values, separated by a delimiter. For example, you want the user to enter multiple host addresses for a flow to target, at the start of the flow.

**Note:** Do not specify the input source as user entered text for a step in the middle of a flow. User inputs in the middle of a flow are not supported in this version.


1. Create an input and set the type to **List of Values**.
2. Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.
3. Under **Input Properties**, in the **Input Delimiter** box, type the character or character sequence that separates the elements in the list.
4. From the **Otherwise** list, select **Prompt User**.
5. In the '**Otherwise: Prompt User**' **Configuration** section, beside **Prompt For**, select **Text**.
6. In the **User Message** box, type a prompt text that lets the flow user know what kind of data the operation needs.

**Note:** Make sure that the prompt text explains to the user how to type a successful list, particularly with attention to the delimiter character (or character sequence) that is required. Note that including a space between list elements when the delimiter character sequence doesn't specify one would cause the operation to fail.

## Specify the input source as list of input values obtained from user selections

The user prompt presents the user with a list from which they can select multiple items. For example, the user needs to select a list of machines for the flow to target, at the start of the flow.

**Note:** Do not specify the input source as user selection for a step in the middle of a flow. User inputs in the middle of a flow are not supported in this version.

1. Create an input and set the type to **List of Values**.
2. Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.
3. Under **Input Properties**, in the **Input Delimiter** box, type the character that separates the elements in the list.
4. From the **Otherwise** list, select **Prompt User**.
5. In the '**Otherwise: Prompt User**' **Configuration** section, beside **Prompt For**, select **Selection**.
6. From the **List Source** list, select one of the following:
  - **Selection List** - select from a set of predefined lists.

From the **Named** list, select the list you want to present to the user.

**Tip:** You can add to the set of predefined lists by creating a list. For information on creating a list, see *Creating selection lists for user prompts*.

- **Domain Term** - Domain terms are specialized selection lists. For example, to specify that a flow runs against certain classes of servers, you can add domain terms for the various kinds of servers in your system and create a user prompt in which the user selects the classes of servers that you want to run the flow against.

From the **Named** list, select the domain term list you want to present to the user.

- **Flow Variable** - create a list that is populated by the contents of a flow variable.

From the **Named** list, type or select the flow variable that contains the list.

In the **Source Delimiter** box, type the character that separates the elements in the list.

7. In the **User Message** box, type a prompt message to let the user know what kind of data is needed.

## Specify the input source as the previous step's result

For example, the previous step might have been to test whether a process works, and the input in the current step is to display the results of that test.

1. Create an input and set the type to either **Single Value** or **List of Values**, depending on whether you want multiple values for the input.
2. From the **Otherwise** list, select **Use Previous Step Result**.
3. If the input value comprises more than one value, then under **Input Properties**, in the **Input Delimiter** box, type the character that separates the elements in the list.

**Note:** If the previous step's result contained multiple items, the input delimiter that you specify must match the delimiter in that result.

## Reference Material

### Input Editor

The Input Editor is where you specify the details of the input source, after you have set up the basic parameters of the input in the **Inputs** tab of the Step Inspector (for a step) or the **Properties** sheet (for a flow or operation).

To display the Input Editor, click the right-pointing arrow  at the end of an input row.

The screenshot shows the 'Inputs Summary' tab for an input named 'filesToAdd'. The 'Input Type' is set to 'Single Value'. Under 'Input Data Flow', 'Assign from Variable' is set to 'filesToAdd', 'Otherwise' is set to 'Use Constant', and 'Assign to Variable' is set to 'filesToAdd'. Under 'Input Properties', 'Obfuscated' is unchecked and 'Required' is checked. 'Validation Format' is set to '<not validated>' and 'Record Under' is set to '<run history>'. On the right, the 'Otherwise: Use Constant' Configuration section is visible with a 'Constant Value' field.

GUI item	Description
<b>Name</b>	Displays the name of the input. May be modified here.
<b>Input Type</b>	Displays the input type. May be modified here.
<b>Assign from Variable</b>	Enter or select the name of the flow variable that will be the source of the input.
<b>Otherwise</b>	Select what will occur if the flow variable specified in the <b>Assign from Variable</b> box does not exist or has no value stored in it.
<b>'Otherwise: &lt;action&gt;' Configuration</b>	Configure the details of what happens if the flow variable specified in the <b>Assign from Variable</b> box does not exist or has no value stored in it. This section varies, depending on which action is selected in the <b>Otherwise</b> list.
<b>Assign to Variable</b>	Select the flow variable that you want to assign the input's value to.
<b>Encrypted</b>	Encrypt the input's value, so that it will appear as a row of asterisks when the flow is run.
<b>Required</b>	Makes this input mandatory.
<b>Validation Format</b>	Validates the input's value with a system evaluator. This functionality is not currently supported.
<b>Record Under</b>	Makes the value available for diagnostics or auditing. This functionality is not currently supported.
<b>Arrow buttons</b> ▲ ▼	If there are multiple inputs in the Inputs Summary, click the arrow buttons to navigate between them in the Input Editor.

## Evaluating Input Data

Evaluators are used to validate inputs. For example:

- If the input is an email address, you can use an evaluator to check that the input is in the correct email format.
- If the input must be a numeric value greater than or equal to 1, you can use an evaluator to check that this is the case.

Studio has standard system evaluators for validating the following:

- Alphanumeric values
- Email
- File name
- IP address
- No white space
- Numeric values
- UUID
- Phone number

**Note:** The default data evaluator for phone numbers supports only the North American telephone number format (1-nnn-nnn-nnnn) for calling from within North America. To validate other regional phone-number formats, you will need to create a system evaluator for this.

Evaluators use the following:

- Simple operators such as =, !=, Begins with, Contains, Match All Words, and Match At Least One Word and so on
- Regular expressions – for more information, see ["Using Regular Expressions in a Flow" on page 204](#)
- Scriptlets – for more information, see ["Using Scriptlets in a Flow" on page 200](#)

## *What do you want to do?*

### Use an evaluator to validate an input

While creating an input in the Input Editor, you can validate the input's value, by selecting an evaluator from the **Validation Format** list.

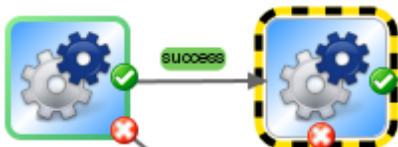
The screenshot shows the 'Inputs Summary' configuration window for an input named 'excelFileName'. The window has tabs for 'Inputs', 'Outputs', 'Responses', 'Description', and 'Scriptlet'. The 'Inputs' tab is active, showing the 'Inputs Summary' for 'excelFileName'. The 'Name' is 'excelFileName' and the 'Input Type' is 'Single Value'. Under 'Input Data Flow', 'Assign from Variable' is 'excelFileName', 'Otherwise' is 'Prompt User', and 'Assign to Variable' is 'excelFileName'. Under 'Input Properties', 'Encrypted' is unchecked and 'Required' is checked. The 'Validation Format' is 'MyEvaluator', which is highlighted with a red rectangle. The 'Record Under' is '<run history>'. On the right, the 'Otherwise: Prompt User' Configuration shows 'Prompt For' set to 'Text' and 'User Message' set to '4552e495-4595-4916-b58b-ce521bdb1e9a.excelFileName.prompt'.

**Record Under:** Makes the value available for diagnostics or auditing. This functionality is not currently supported.

For more information about creating an input, see ["Creating Input" on page 120](#).

## Creating Transitions

You connect any two steps in a flow with a transition. A transition starts from one of a step's responses (represented by a response icon, such as **Success** or **Fail**) and goes to another step. Every response in a flow must have a transition either to a subsequent step or to a return step that returns an outcome for the entire flow and ends the flow.



More than one response can be connected to a given step. For example, several failure responses often are connected to a single failure return step.

For information about setting the responses for an operation, see ["Setting Responses" on page 139](#).

## Best Practices

- As much as possible, transition lines should not cross.
- Use straight transitions, if possible. You should only use curved transitions when this is necessary for the flow layout.
- When possible, position steps so that transitions are horizontal, vertical, or at a 45-degree diagonal.

- Collapse multiple transitions from one step to another so that a single line represents all of the transitions.
- Position transition labels so that they do not overlap the step labels or each other.
- Rename transition labels if this will make the flow clearer to another user.
- Place transition labels toward the outside of the flow when possible. For example, if two steps are at the top of the flow canvas, the transition labels should be above the transition lines. If the steps are at the bottom of the canvas the labels should be below the transition lines.

## What do you want to do?

### Add a transition between steps

1. Open the flow on the authoring pane in Studio.
2. On the step that you want to connect to the next step, click either the response name or the icon that represents one of the responses, and drag to the destination step for that response.
3. Double-click the transition. The Transition Inspector opens.
4. (Optional) To change the transition's name, in the **Name** box, type the new name.
5. In the **Description** box, type a description that explains what happened in the preceding step that caused this transition to be followed. This description will appear in the **Results Summary** area in HP OO Central, when the flow is run.

**Note:** A transition's description relates to the step from which the transition originated. For example, the message "localhost successfully pinged" describes what happened in the step "Ping Target System", even though it was written for the transition that follows the step.

### Create a description that includes a flow variable

You can use flow variables in the description to store changeable information. For example, to identify a server whose name is stored in the `servername` flow variable, you could type: "Server `${servername}` is available for connection".

1. Create a transition between two steps.
2. Double-click the transition to open the Transition Inspector.
3. In the **Description** box, type a description that includes a flow variable that contains data that came from the step's operation or elsewhere in the flow run. The reference must use the format `${flow variable name}`.



For example, a step that carries out a ping command can save the host machine's name into a flow variable called `host`. To use this value in the transition description you can reference it with the syntax `${host}`. The description in the transition from the success response might read "Successfully pinged `${host}`." When this is run in HP OO Central against a host named "server1", the summary description will read "Successfully pinged server1."

### Limit who can run the step beyond the transition (gated transition)

Gated transitions let you control who can continue with the flow beyond the transition, by restricting access to the next step to users who belong to a particular role. If someone who is not a member of this role group tries to run the flow, the flow will stop, and the user will have the choice of handing off the flow to another user or canceling the flow.

Gated transitions are colored red.

1. Create a transition between two steps.
2. Double-click the transition to open the Transition Inspector.
3. Select the **Check user's groups before proceeding** check box.
4. From the **Required Role Alias** list, select the role that the user must be assigned in order to continue running the flow.

### Require that the run is handed off following the transition

You can set a transition to hand off the flow to another person. This might be necessary if the next step requires information from a different user.

During the flow run, a hand-off transition opens a new email message with the URL of the flow included in the body of the message. The person running the flow can address the email message to the person taking over the flow and then send the message. After the recipient receives the message, they can continue running the flow.

1. Create a transition between two steps.
2. Double-click the transition to open the Transition Inspector.
3. Select the **Hand-off flow after this transition** check box.

### Count the completion of the transition in the flow's ROI value

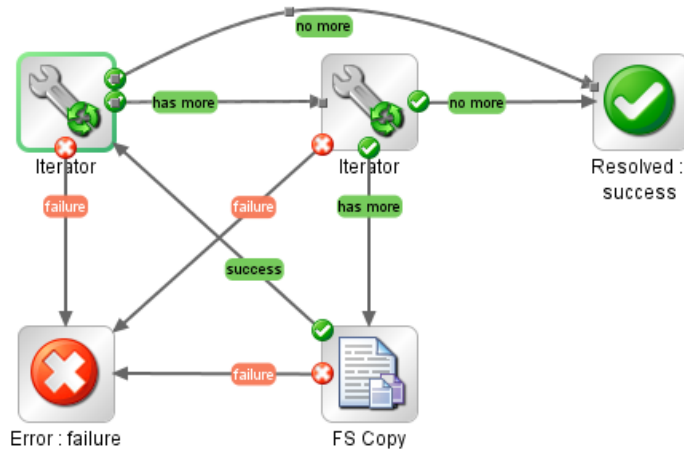
You can associate a value with a transition in the flow. These values represent the return on investment (ROI) value of each transition. When the flow is executed, these values are recorded, based on the transitions actually made. Administrators will be able to view reports, in Central, which display the ROI values of the flow, giving it valuable business statistics.

1. Create a transition between two steps.
2. Double-click the transition to open the Transition Inspector.

3. In the **Transition ROI Value** box, enter a numerical value for the transition.

### Add a curve-defining point to create a curved transition

You can add a curve-defining point to reshape a transition from a straight line to a curve. This helps to tidy up your flow or to separate transitions that are stacked.



1. Position your mouse over the transition where you want to place the curve-defining point.
2. To create the point, hold down SHIFT and click the mouse.
3. Drag the point until the transition curves the way you want it to.

### Remove a curve-defining point

Position the cursor over the curve-defining point, hold down SHIFT, and click the mouse.

### Move a transition name

Click the transition name and drag it to another location.

### Break a transition between two steps

To break an existing transition between two steps, select the transition and press the Delete key on the keyboard.

## Reference Material

### Transition Inspector

The Transition Inspector is where you specify the details of a transition.

The screenshot shows the 'Inspector' window in the Studio Authoring Guide. The 'Transition Name' is set to 'success'. Under the 'Gated Transition' section, the checkbox 'Check user's groups before proceeding' is selected, and the 'Required Group' is set to 'AUDITOR'. The checkbox 'Hand-off flow run after this transition' is not selected. The 'Transition ROI Value' is set to '0.0'. The 'Description' section is empty.

GUI item	Description
<b>Transition Name</b>	By default, the transition name is the same as the name of the response where it originated (success, failure, and so on), but you can change the transition name.
<b>Check user's group before proceeding</b>	Select this check box to create a gated transition, which will only let a user proceed with the next step if they are assigned to the required role alias.
<b>Required Role Alias</b>	Select the role alias that the user must be assigned to in order to continue running the flow.
<b>Hand-off flow after this transition</b>	Hand-off transitions are not supported in the current version. Select this check box to hand off the flow to another person after the transition.
<b>Transition ROI Value</b>	Enter a value for the transition, so that if the transition is followed during a flow run, its value is added to the value of the flow for that run.
<b>Description</b>	Type a description that explains what happened in the preceding step that caused this transition to be followed. The description appears in the <b>Results Summary</b> area in HP OO Central.

## Setting Responses

A response is one of a number of possible outcomes of an operation or flow.



The four types of response are:

- **Resolved** ✅ – This is the standard response for an operation or flow that runs correctly.
- **Diagnosed** ➕ – This response indicates that an operation or flow has determined what a problem is and has opted not to take action on it other than notification.
- **No Action Taken** 🟡 – This response is used when an operation or flow gathers data but cannot determine any diagnosis or remediation.

**Note:** An operation that is intended solely to gather data should return **Resolved** ✅ when it is complete, rather than **No Action Taken** 🟡.

- **Error** ❌ – This response is used if the step or flow fails to run. For example, because of bad input or failure to reach a system.

In some cases, an operation or flow may have multiple responses of the same type. For example, an SQL query operation may have the following outcomes:

- **More items** 🟡
- **No more items** 🟡
- **Failure** ❌

You can add, delete, and modify the responses in an operation or flow. You cannot modify the responses on a step, with the exception of return steps. For more information about return steps, see ["Creating Return Steps" on page 181](#).

## Response Rules

A rule enables you to limit the response so that it only occurs when a particular condition of the operation's result is true. The rule compares a value that you specify with a value in a field of an operation's raw results.

For example, you can create a rule that will only give a **Success** response if the results include a value that is greater than 1.

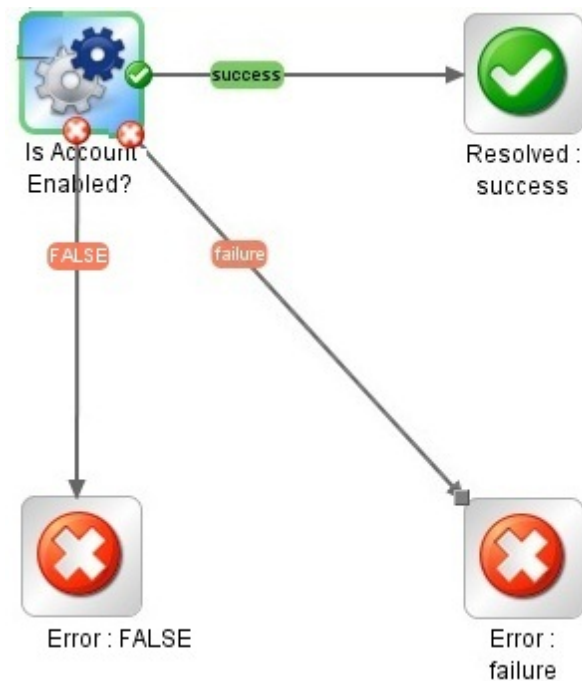
If you create more than one rule for a response, then all the rules for that response must evaluate to true for the response to be chosen.

Responses are evaluated in the order in which they are listed on the operation's **Responses** tab. The first response whose rule or rules evaluate to true is the response chosen. So if the port open response's rule evaluates to true, then that is the response chosen, even if the rule for port listening

would also evaluate to true. The order of responses can be very important for obtaining the most helpful outcome for your flow.

## Best Practices

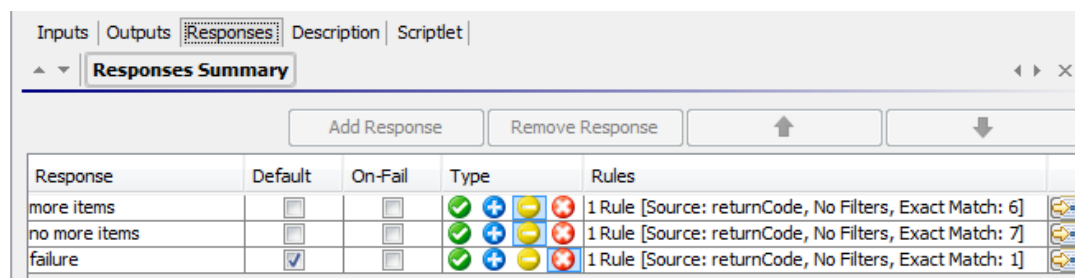
- Avoid confusing a failed operation with a negative result. For example, if an operation asks a question for which the answer may be TRUE or FALSE, a FALSE answer is not the same as a failure. In such a case, you need two **Error** return results: one for a FALSE result and one for a failure of the operation.








## What do you want to do?

### Add a response to an operation

1. On the **Responses** tab of the operation, click **Add Response** and then type a name for the new response.



2. To make a response the one chosen if an operation fails to execute, select the check box in the **On-Fail** column for that response.
3. To identify a response as the default response, select the check box in the **Default** column. The default response is the one chosen if none of the responses' rules evaluate to true.
4. In the **Type** column, select the type of response:
  - **Resolved:** 
  - **Diagnosed:** 
  - **No action:** 
  - **Failure:** 

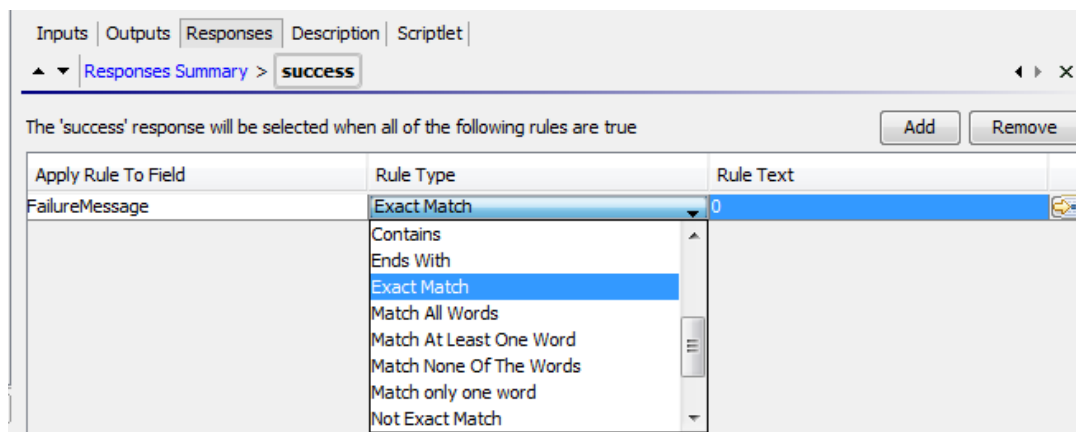
This determines which response icon will be shown on the operation, when it is used to create a step.
5. To create a rule for the response, at the right end of the response's row, click the right-pointing arrow . For more information, see *Create a rule for the response*, below.

## Create a rule for the response

A rule enables you to limit the response so that it only occurs when a particular condition of the result is true.

1. Create a new response in an operation.
2. In the response rule editor, click **Add**.

The new rule appears.



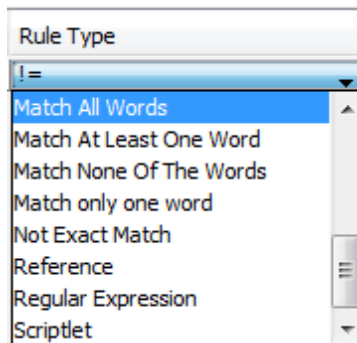
3. In the **Apply Rule to Field** column, select the results field whose value you want to test a rule against.

The result fields you can test include the result's exit code, output string, error string, failure message, and timed-out true or false.

**Note:** To display more information on these result fields, click the **Description** tab.

To see what the values for these fields are, test the operation in a flow, using the Studio Debugger. In the Debugger, when you execute a flow, you will find the results for any step in the Step Result Inspector. For more on the Debugger, see ["Testing and Debugging a Flow" on page 210](#).

4. In the **Rule Type** column, select the comparison or match that you want to test the field value with.




- Select a simple operator such as =, !=, Begins with, Contains, Match All Words, and Match At Least One Word and so on
- Select **Regular Expression** to create a regular expression.
- Select **Scriptlet** to create a scriptlet.
- Select **Reference** to create a reference to a shared rule.

5. In the **Rule Text** column, type the text to use in the test.

## Filter and test a response rule

In the Rule Details Editor, you can:

- Specify the rule in greater detail, including the use of rules, filters, regular expressions, or scriptlets.
  - Test the rule as you develop it
1. To open the Rule Details Editor, click the right-pointing arrow  at the right end of the rule's row.

Inputs | Outputs | Responses | Description | Scriptlet

▲ ▼ Responses Summary > failure > rule (1 of 1)

Rule Details

Rule Type: Exact Match Filter Result Before Applying Rule

Test to see if the input contains a value

Ignore Case: ☒

Text: 1

Input (Matches Highlighted)

Pattern Does Not Match Clear Quick Command

Enter the input characters you'd like to match against in this area. All (non-overlapping) matches will be highlighted. The first character of each match will also be underlined, so you can better tell when two matches are adjacent. (eg: highlightedhighlighted).

Editing either text area will cause an immediate update in the highlighted matches. If the regular expression entered has syntax problems -- even temporarily while typing -- the insertion caret will turn red until editing restores a valid expression.

**Note:** If you choose **Scriptlet** as the rule type, the Rule Details Editor includes a scriptlet editor. For more information on creating scriptlets, see ["Using Scriptlets in a Flow" on page 200](#).

If you choose **Regular expression** as the rule type, the Rule Details Editor includes a regular expression editor. For more information on creating regular expressions, see ["Using Regular Expressions in a Flow" on page 204](#).

2. To use a different rule type, select it from the **Rule Type** list.
3. To filter the result before applying the rule, click **Filter Result Before Applying Rule** and, in the Filter Editor, create the filter.

Creating a filter for a response rule is the same as creating a filter for an output or result. See ["Filtering Output and Results" on page 158](#).

4. For most of the rule types, in the **Text** box, type the text that you want to test comparison with and, if you want to ignore case, select the **Ignore Case** check box.



For **Regular Expression** rules, specify the regular expression and its application as you do when creating a **Regular Expression** filter for operation results. For information, "[Using Regular Expressions in a Flow](#)" on page 204.

The results box displays the results of the test: it displays either **Pattern Matches** or **Pattern Does Not Match** and highlights the matching text.

5. To work on another rule for the operation's response, click the up or down arrow next to **Responses Summary**.

**Note:** In a rule, when you use a mathematical comparator (such as =, !=, <, or >) in an evaluation of a string that starts with a number, the comparator compares only the numerical portion of the string. For example, if you compare "123" with "123Test" using != (does not equal), the evaluation would be "false", although "123" is clearly not the same as "123Test". You can work around this issue, however, by comparing the strings with the **Not Exact Match** evaluator.

## Add a response to a flow

When you create responses for a flow, you make these responses available for return steps in the flow.

For example, if the outcome leading to an **Error** return step is not a failure in an operation but a result that did not meet a required threshold, then you might want to create a new response for the **Error** return step, which reflects this outcome, so that it will appear as **Error: threshold not met**.

1. Open the **Properties** sheet for the flow.
2. Click the **Responses** tab.
3. Click **Add Response** and then, in the text box that appears, type the name of the response. For example, `threshold not met`.
4. Click **OK**.

When you create an **Error** return step for the flow, you will be able to select **threshold not met** as a response. For more information about return steps, see "[Creating Return Steps](#)" on page 181.

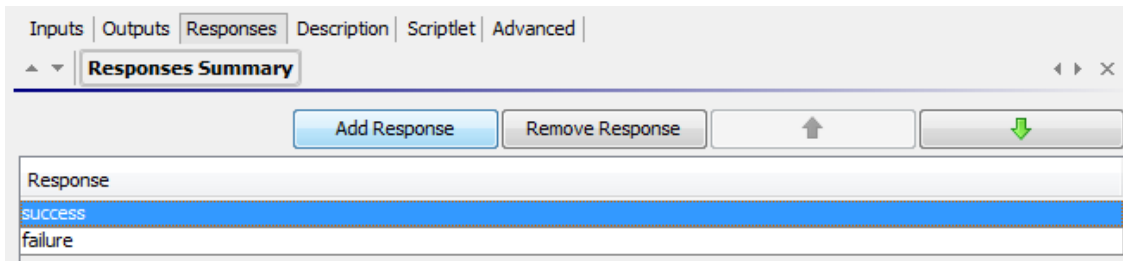
## Delete a response from an operation or flow


1. Open the **Properties** sheet for the operation or flow.
2. Click the **Responses** tab.
3. Select the response, and then click **Remove Response**.

## Reference Material

### Flow Properties sheet > Responses tab

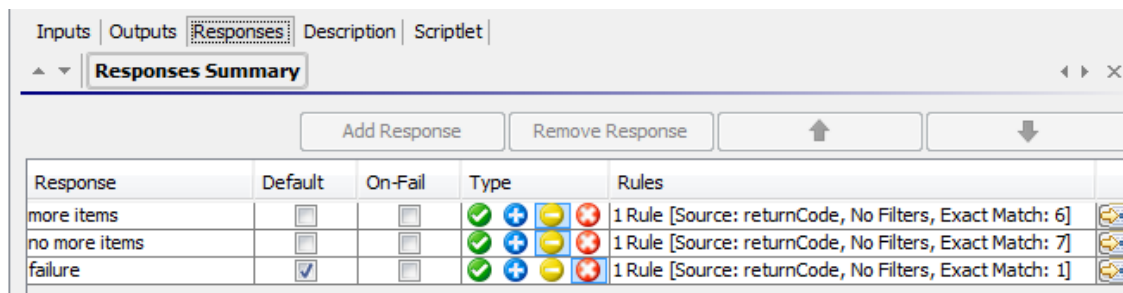
The **Responses** tab in an flow's **Properties** sheet is where you specify the possible responses to be available for return steps in a flow. For example, **Error: threshold not met**.









GUI item	Description
<b>Add Response</b>	Adds a new response row.
<b>Remove Response</b>	Removes the selected response row.
	Click to move the selected response up or down in the list.

### Operation Properties sheet > Responses tab

The **Responses** tab in an operation's **Properties** sheet is where you specify the possible responses for an operation.



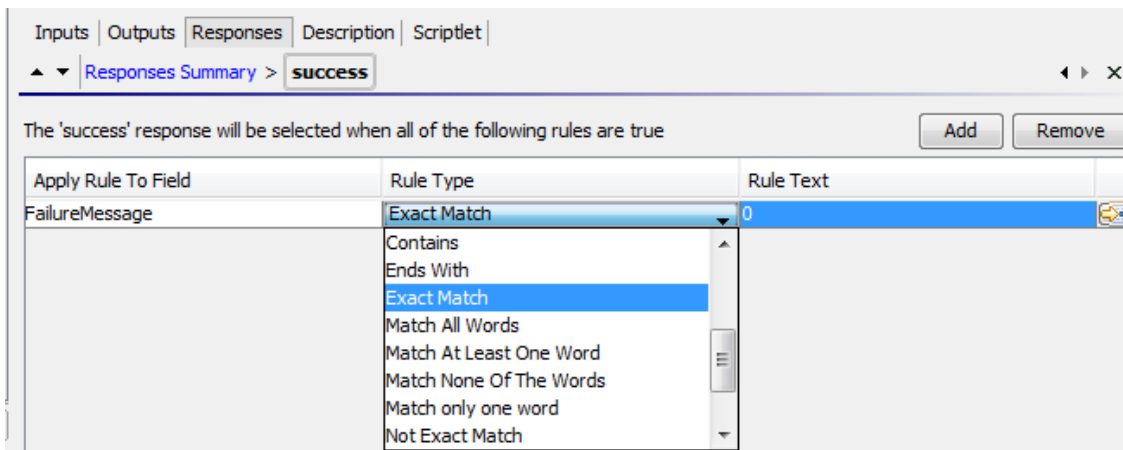
GUI item	Description
<b>Add Response</b>	Adds a new response row.
<b>Remove Response</b>	Removes the selected response row.
	Click to move the selected response up or down in the list.

<b>Default</b>	Select to identify a response as the default response. The default response is the one chosen if none of the responses' rules evaluate to true.
<b>On-Fail</b>	Select to make a response the one chosen if an operation fails to execute.
<b>Type</b>	Select the type of response: <ul style="list-style-type: none"> <li>• Success / resolved: </li> <li>• Diagnosed: </li> <li>• No action: </li> <li>• Failure: </li> </ul>
<b>Rules</b>	Displays any rules that have been created for the response.
	Click to display the Rule Editor, to create a rule for the response.



## Operation Properties sheet > Responses tab > Rule Editor

The Rule Editor is where you limit the response so that it only occurs when a particular condition of the result is true.

For example, you can create a rule that will only give a **Success** response if the results include a value that is greater than 1.



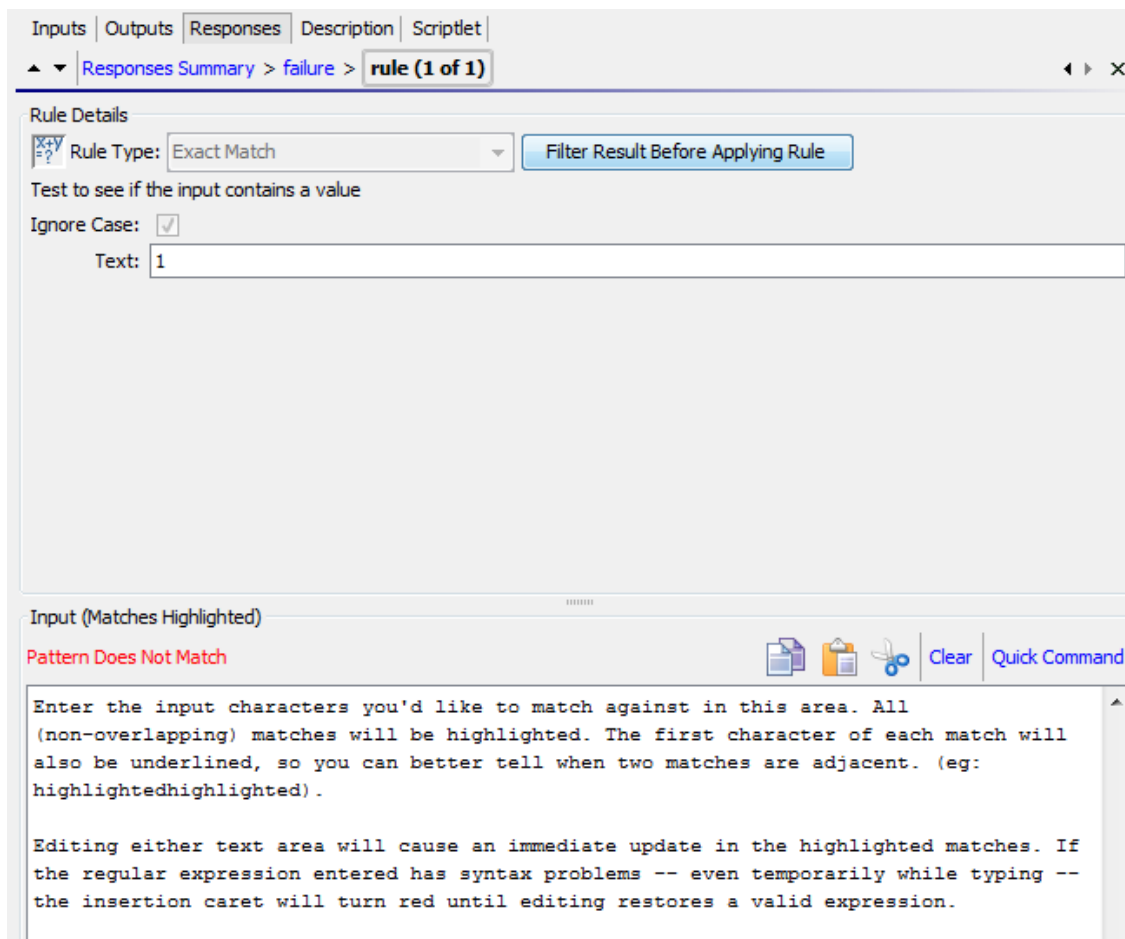
GUI item	Description
<b>Add Response</b>	Adds a new response row.
<b>Remove Response</b>	Removes the selected response row.

	Click to move the selected response up or down in the list.
<b>Apply Rule to Field</b>	Select the results field whose value you want to test a rule against. The result fields you can test include the result's exit code, output string, error string, failure message, and timed-out true or false.
<b>Rule Type</b>	Select the comparison or match that you want to test the field value with.
<b>Rule Text</b>	Type the text to use in the test.
	Click to open the Rule Details Editor, to test and filter the rule.

## Operation Properties sheet > Responses tab > Rule Editor > Rule Details Editor

The Rule Details Editor is where you can test and apply filters to a rule.



Rules enable you to limit a response so that it only occurs when a particular condition of the result is true.



Inputs | Outputs | Responses | Description | Scriptlet

▲ ▼ Responses Summary > failure > rule (1 of 1) ◀ ▶ ✕

**Rule Details**

 Rule Type: Exact Match  Filter Result Before Applying Rule




Test to see if the input contains a value

Ignore Case: ☒

Text: 1

---

Input (Matches Highlighted)

Pattern Does Not Match    Clear Quick Command

Enter the input characters you'd like to match against in this area. All (non-overlapping) matches will be highlighted. The first character of each match will also be underlined, so you can better tell when two matches are adjacent. (eg: highlightedhighlighted).

Editing either text area will cause an immediate update in the highlighted matches. If the regular expression entered has syntax problems -- even temporarily while typing -- the insertion caret will turn red until editing restores a valid expression.

GUI item	Description
<b>Rule Type</b>	Displays the rule type that was selected in the Rule Editor and enables you to select a different rule type.
<b>Filter Result Before Applying Rule</b>	Click to display the Filter Editor, in order to filter the result before applying the rule.
<b>Text</b>	Type the text that you want to test comparison with.
<b>Ignore Case</b>	Select to ignore whether the text is in upper or lower case.
<b>Results box</b>	Displays the results of the test: Displays either <b>Pattern Matches</b> or <b>Pattern Does Not Match</b> and highlights the text that matches.
<b>Copy</b>	Copy data within the results box.
<b>Paste</b>	Paste data into the results box.
<b>Cut</b>	Cut data within the results box.
<b>Clear</b>	Clear data within the results box.
<b>Quick Command</b>	Type a command that generates the data on which you want to test the filter. The output of the command appears in the results box.

## Creating Outputs and Results

One of the ways to capture data for use in a flow is via a step result. There are two ways to assign this data:

- When the output in a result is assigned to a **flow variable**, you can pass it as data to other steps in the flow.
- When the output in a result is assigned to a **flow output field**, you can pass it as data to a parent flow.

There are a number of steps in this process:

1. Set up the outputs for an operation, including the primary output.

See ["Setting Operation Outputs" on the next page](#).

2. When you use the operation for a step in a flow, you decide which of the operation outputs you want to use as step results—which ones you want to assign to flow variables or flow output fields.

See ["Setting Step Results" on page 153](#).

3. (Optional) It is possible to narrow an output or result to a more highly focused selection by

creating filters.

See ["Filtering Output and Results" on page 158](#).

## Setting Operation Outputs

The first stage in setting up flow outputs is to set up the outputs in the operation. After this is done, when you (and other flow authors) use this operation in a flow, you will be able to assign outputs to flow variables.

### *Types of Operation Output*

The different kinds of operation output include:

- **Raw result** is *all* of the operation's return code, data output, and error string.

The raw output is not directly visible in Studio, except as the raw result of a step that was created from the operation.

- The primary and other outputs are portions of the raw output—for example, success code, output string, error string, or failure message—that you specify as an output.
  - The **primary output** is the output used to populate the step's primary result. The primary output supplies a value to an input whose assignment is **Previous Step's Result**.
  - A **secondary output** in an operation is another output, in addition to the primary output.

**Tip:** You can narrow an output to a more highly focused selection by creating one or more filters for the output. See ["Filtering Output and Results" on page 158](#).

### *Examples of Operation Output*

Most operations have outputs that are specific to the operation. However, you will frequently encounter the following outputs when working with operations in the Library's **Accelerator Packs**, **Integrations**, and **Operations** folders:

- **returnResult**

When you see "returns:" with no field named, this is usually the primary output. The primary output is also accessible via **Result** with a capital R (which is universal).

- **response** (or **returnCode**)

A code or string used to determine the response the operation will take.

- **failureMessage**

An internal output provided by the infrastructure. If an operation returns a failure, this output provides the exception. Note that many operations do not use this output.

## ***Best Practices***

- Be consistent about case. For example, use camel case for all output names.
- If you are working with an integration, keep the original output names from the API being used.

## ***What do you want to do?***

### **Specify the primary output for an operation**


When you set up an operation, you can specify its primary output. Once you have created a primary output, you can change its source, but you cannot return to having no primary output.

1. Right-click the operation in the **Project** pane and select **Properties**.
2. Select the **Output** tab.
3. From the **Extract Primary Output from Field** list, select a source field. For example, **FailureMessage**.

**Tip:** For information on the data provided in each output field, click the operation's **Description** tab.

### **Add a secondary output for an operation**

A secondary output in an operation is another output, in addition to the primary output.

1. Right-click the operation in the **Project** pane and select **Properties**.
2. Select the **Output** tab.
3. Click **Add Output**.
4. Type the name of the output.
5. From the **Output Field** list, select the field from which an output gets its data.
6. To create filters for the output data in the secondary output, click the right-pointing arrow  at the end of the row.

For information on creating filters, see ["Filtering Output and Results" on page 158](#).

### **Delete an output from an operation**

1. Right-click the operation in the **Project** pane and select **Properties**.

2. Select the **Output** tab.
3. Select the output you want to delete and click **Remove Output**.


## Change the field from which an output gets its data

1. Open the **Properties** sheet for the operation and select the **Output** tab.
2. To change the field for the primary output, click the downward-pointing arrow to the right of the **Extract Primary Output From Field** box, and then select the desired field from the list.
3. To change the field for a secondary output, click in the **Output Field** column of the output's row, and then select the desired field from the list.

## Reference Material

### Properties sheet > Outputs tab

The **Outputs** tab in the **Properties** sheet is where you specify the primary and secondary outputs in an operation.

GUI item	Description
<b>Extract Primary Output From Field</b>	Select the field from which the primary output will get its data
<b>Edit Filters</b>	Displays the Filter Editor for the primary output
<b>Add Output</b>	Adds a new output row
<b>Remove Output</b>	Removes the selected output row
<b>Output Field</b>	Select the field from which this secondary output will get its data
	Displays the Filter Editor for the output in the row



## Setting Step Results

Operations produce a variety of outputs, but outputs are not automatically retained in the flow. If they were, this could affect performance, by slowing down the flow with unnecessary data.

In the **Results** tab of the Step Inspector, you specify the results that you need. Results contain output from the operation. You can store results in two ways:

- Create **flow variables**, which are accessible to operations, transitions, and prompts *in the same flow*. For more information, see ["Working with Variables" on page 175](#).

**Example:** A step called **LocalPing** determines whether a target host is available and stores the output of the ping operation in a result called **PingOutput**. This creates a flow variable called **PingOutput**, which can be used in later steps.

The next step, called **Display**, displays the **PingOutput** variable to the user. The prompt text in this step is set up as Ping Results: {PingOutput}.

- Create **output fields**, which are accessible to operations, transitions, and prompts *in the parent flow*, if the flow is used as subflow (a step in another flow). For more information, see ["Creating a Subflow Within a Flow" on page 185](#).

**Example:** A parent flow includes a step that contains the **Windows Health Check** flow as a subflow. The results of the **Windows Health Check** flow are stored as value in an output field called **HealthCheckOutput** and are available for the main flow.

The main flow includes a **Send Mail** operation, which displays the value of the **HealthCheckOutput** output field in the body of the email.

There are two kinds of step result:

- **Raw result** is *all* the raw data that was returned from an operation executed in the context of a flow. The step's raw and primary results come from the underlying operation's raw output and primary output.
- Other results, which you create on the **Results** tab of the Step Editor. In the Step Inspector, you can create and specify secondary results.

Before setting up the flow results in a step, make sure that the primary output has been set up for the relevant operation. See ["Setting Operation Outputs" on page 150](#).

**Tip:** You can narrow a result to a more highly focused selection by creating one or more filters for the output. See ["Filtering Output and Results" on page 158](#).

## ***Best Practices***

- Be consistent about case. For example, use camel case for all result names.
- The operation or flow that a step is based on may provide several outputs. But when adding step results, make sure to only to use the outputs that you need in the flow.

## ***What do you want to do?***

### **Create a primary result in a step**

The primary output is set in the operation. The primary output supplies a value to an input whose assignment is **Previous Step's Result**.

In a step, you can decide to capture this primary output in a flow variable (to be used in other steps in the flow) or a flow output field (to be passed to a parent flow).

1. Double-click a step in the authoring pane.
2. Select the **Results** tab and click **Add Result**.
3. In the **Name** column, enter a name for the result. Press the Return key on your keyboard. This name will be used as the name of the flow variable or flow output field.

**Note:** Do not use "Result" for the result name.

4. From the list in the **From** column, select the primary output as the source for the result.

For example, you might select **Result Field: returnResult**, which is the primary output for that operation.


For information about setting the primary output, see ["Setting Operation Outputs" on page 150](#).

### **Create a secondary result in a step**

1. Double-click a step in the authoring pane.
2. Select the **Results** tab and click **Add Result**.
3. In the **Name** column, enter a name for the result. Press the Return key on your keyboard. This name will be used as the name of the flow variable or flow output field.
4. From the **From** list, select the source for the result.
5. From the **Assign To** list, decide where the value will be stored:

- To store the value in a flow variable, select **Flow Variable**
  - To make the value available to a parent flow, select **Flow Output Field**
6. From the **Assignment Action** list, select the appropriate action:
- **OVERWRITE** – Replace the current value of the flow variable or flow output field with this value.
  - **APPEND** – Place this value at the end of the current value of the flow variable or flow output field.
  - **PREPEND** – Place this value in front of the current value of the flow variable or flow output field.
  - Use one of the four arithmetic assignment actions, **ADD**, **SUB**, **MULTIPLY**, and **DIVIDE**, to arithmetically modify the current value of the flow variable or flow output field.

For example, if the step result is 3.14, and you select **MULTIPLY**, the effect is to multiply the current value of the flow variable or flow output field by 3.14.

7. To create filters for the output data in the secondary result, click the right-pointing arrow  at the end of the row.

For information on creating filters, see ["Filtering Output and Results" on page 158](#).

### Delete a result from a step

1. Double-click a step in the authoring pane.
2. Select the **Results** tab and click **Remove Result**.
3. Save the step.

### Change the field from which a result gets its data

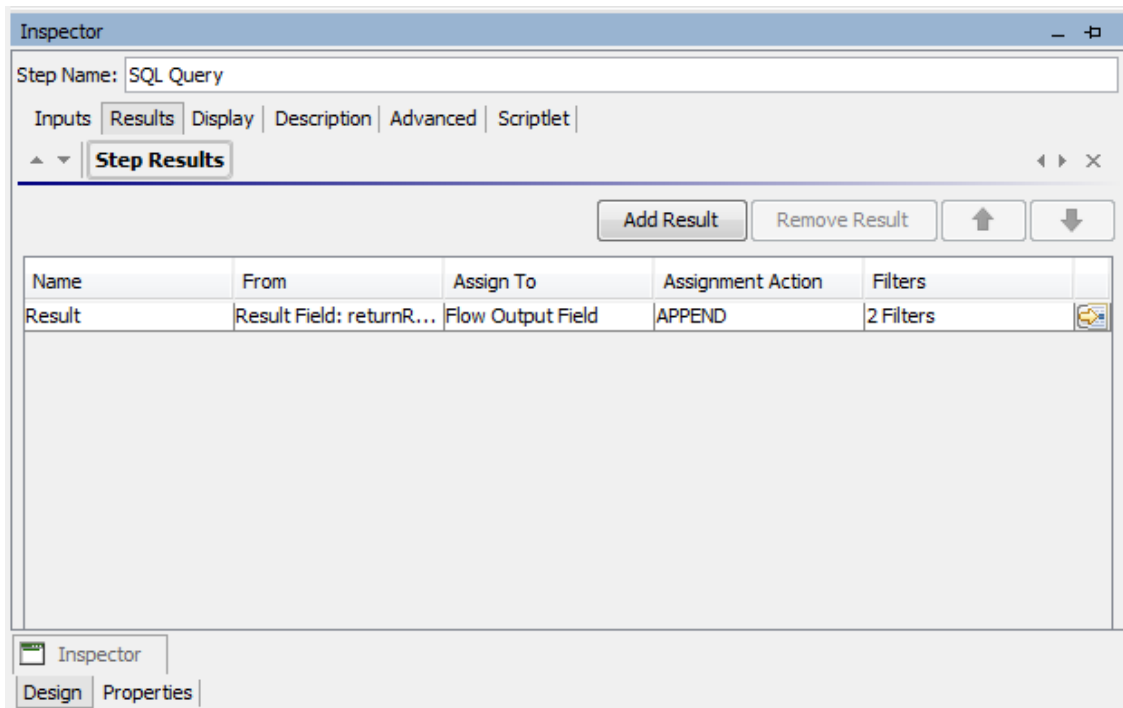
1. Double-click a step in the authoring pane.
2. Select the **Results** tab.
3. Click in the **From** column of the output's row, and then select the desired field from the list.
4. Save the step.

## Reference Material


### Step Inspector > Results tab

The **Results** tab in the Step Inspector is where you specify which outputs will be saved into flow

variables or made available for parent flows.

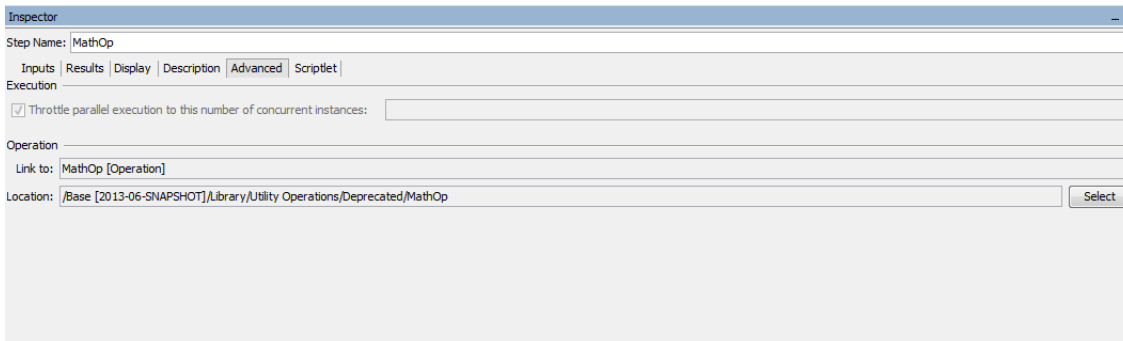


GUI item	Description
<b>Add Result</b>	Adds a new result row.
<b>Remove Result</b>	Removes the selected result row.
<b>Name</b>	Enter a name for the result. This name will be used as the name of the flow variable or flow output field.
<b>From</b>	Select the source for the result.
<b>Assign To</b>	Select where the result value will be stored: <ul style="list-style-type: none"> <li>To store the value in a flow variable, select <b>Flow Variable</b></li> <li>To make the value available to a parent flow, select <b>Flow Output Field</b></li> </ul>

<b>Assignment Action</b>	<p>From the <b>Assignment Action</b> list, select the appropriate action:</p> <ul style="list-style-type: none"> <li>• <b>OVERWRITE</b> – Replace the current value of the flow variable or flow output field with this value.</li> <li>• <b>APPEND</b> – Place this value at the end of the current value of the flow variable or flow output field.</li> <li>• <b>PREPEND</b> – Place this value in front of the current value of the flow variable or flow output field.</li> <li>• Use the arithmetic assignment actions <b>ADD</b>, <b>SUB</b>, <b>MULTIPLY</b>, and <b>DIVIDE</b> to arithmetically modify the current value of the flow variable or flow output field.</li> </ul> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>For example, if the step result is 3.14, and you select <b>MULTIPLY</b>, the effect is to multiply the current value of the flow variable or flow output field by 3.14.</p> </div>
	<p>Displays the Filter Editor for the result in the row.</p>

## Step Inspector > Advanced tab

The **Advanced** tab in the Step Inspector is where you specify where you can change the source operation that the step is based on.



GUI item	Description
<b>Link to</b>	Displays the source operation that the step is based on.
<b>Location</b>	Displays the location of the source operation that the step is based on.
<b>Select</b>	Opens the Select Source Operation dialog box, where you can navigate to and select the operation that you want to base the step on.



## Filtering Output and Results

You can create filters to extract and modify parts of an operation's output or a step's result.

For example, if you only want the maximum, minimum, and average round-trip times for a ping operation to a server, you can isolate and extract all three pieces of information from the operation's raw output by filtering the raw output into three outputs.

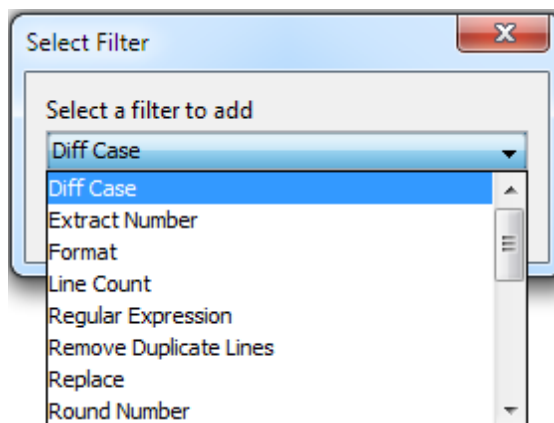
### *What do you want to do?*

#### Create a filter

1. Open the Filter Editor. This step varies, depending on what you are filtering:
  - To filter an operation's primary output, open the operation's **Properties** sheet, click the **Outputs** tab, and then click the **Edit Filters** button.
  - To create a filter for an operation's secondary output, open the operation's **Properties** sheet, click the **Outputs** tab, and then click the right-pointing arrow  at the end of the output row.
  - To create a filter for a step's secondary result, double-click the step in the authoring pane, click the **Results** tab, and then click the right-pointing arrow  at the end of the result row.
2. In the Filter Editor, click the **Add** button.

**Note:** You can add multiple filters to an output or result.

3. From the **Select Filter** list, select the type of filter.



4. In the **Details** area in the upper-right of the Filter Editor, configure the filter. See [Filter Options](#) to see the options for different filters.

## Test a filter with data from a command line

To test, a filter, paste some data into the **Test Filter Input** box. If this data can be generated by a local command-line command, do the following:

1. Open the Filter Editor for an output or result.
2. Click **Clear** to clear the **Test Filter Input** box.
3. Click **Quick Command**.
4. Type a command that generates the desired data.
5. Click **OK**. The output of the command appears in the **Test Filter Input** box.
6. Do one of the following:
  - Click **Test All Filters**
  - Select the filters you want to test and click **Test Selected Filters**

The filters are applied to the data in the **Test Filter Input** box (in top-to-bottom order), and the filtered results appear in the **Test Output** box.

## Test a filter with data from the debugger

If the data you need is produced by means that you cannot reproduce with a simple command-line command, copy the data from the debugger and paste it into the **Test Filter Input** box:

1. Open the Filter Editor for an output or result.
2. Click **Clear** to clear the **Test Filter Input** box.
3. Run the flow in the debugger.
4. Highlight the relevant step.
5. In the **Step Result Inspector** pane, copy the contents of the **Raw Result** tab.
6. In the Filter Editor, paste the contents into the **Test Filter Input** box.
7. Do one of the following:
  - Click **Test All Filters**
  - Select the filters you want to test and click **Test Selected Filters**

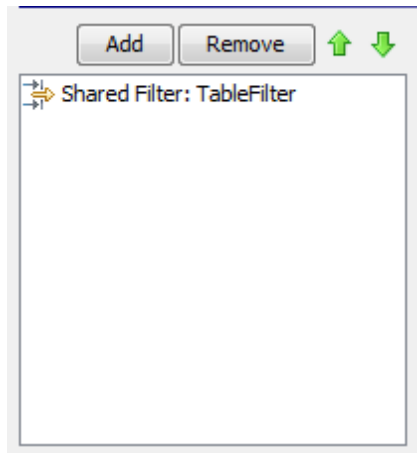
The filters are applied to the data in the **Test Filter Input** box (in top-to-bottom order), and the filtered results appear in the **Test Output** box.

## Filter a different output or result

While you have the Filter Editor open, you can click the upward- or downward-pointing arrows beside **Outputs Summary** to create a filter for a different output or result.

## Use a system filter in an output or result

1. Open the Filter Editor of the output or result on which you want to use the system filter.
2. In the **Projects** pane, expand the **Configuration** and **System Filters** folders.
3. Drag the filter that you want to use from the **System Filters** folder to the **Filter** list in the Filter Editor.



## Save a filter for reuse as a system filter

You can take an existing filter in an operation and save it as a system filter. The resulting system filter is independent of the operation for it was created and can be reused in any output or result.

For more information, see ["Configuring System Filters" on page 90](#).

## Filter Options

### Diff Case

A **Diff Case** filter changes all the characters in the string either to upper case or to lower case. If you leave the **To Upper Case** check box cleared, the filter changes all the characters to lower case.

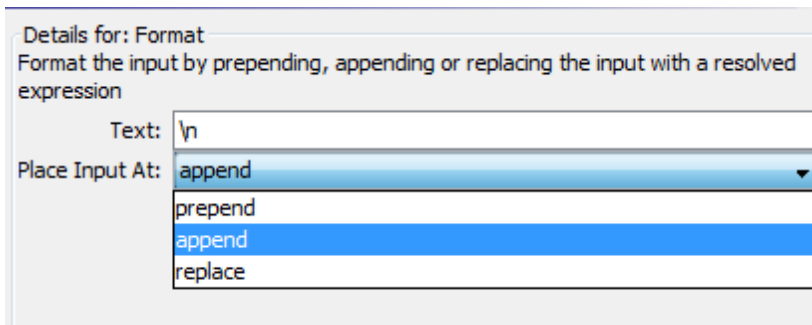
### Extract Number

An **Extract Number** filter extracts the first number found in the result. The filter treats an unbroken series of integers as a single number. For example, the **Extract Number** filter would extract the number "123" from the strings "123Test" or "Test123".

### Format

A **Format** filter attaches text to a result or output or replaces the original content of the result or output with the text that you specify.





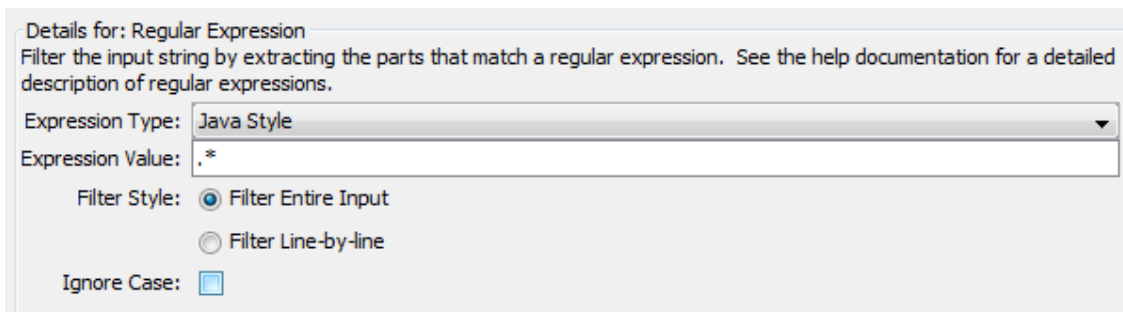
1. In the **Text** box, type the text you want to attach to the result or use to replace the result.
2. From the **Place Input At** list:
  - To attach the text to the front of the existing text, select **prepend**
  - To add the text to the back of the existing text, select **append**
  - To replace the output with the text, select **replace**

## Line Count

A **Line Count** filter outputs the total number of lines of the result.

## Regular Expression

A **Regular Expression** filter filters the raw results using a regular expression (regex).



1. From the **Expression Type** list, select **Java Style**. Do not use the other styles; they have been deprecated.
2. In the **Expression Value** box, type the regular expression.
3. For **Filter Style**, select **Filter Entire Input** or **Filter Line-by-line**, according to how you want the filter applied to the raw results.
4. To make the regular expression not case-sensitive, select **Ignore Case**.

For more information about working with regular expressions, see ["Using Regular Expressions in a Flow" on page 204](#).

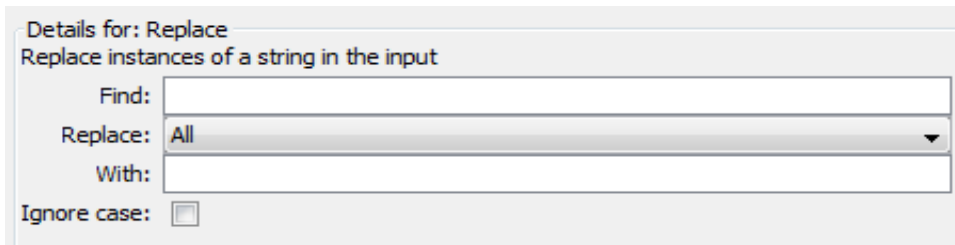
## Remove Duplicate Lines

This filter finds lines that are identical and removes all but one of them.

To apply the filter only to duplicate lines that follow each other directly, select **Consecutive**.

## Replace

This filter replaces the first or last instance or all instances of one string with another string.



Details for: Replace  
Replace instances of a string in the input

Find:

Replace: **All**

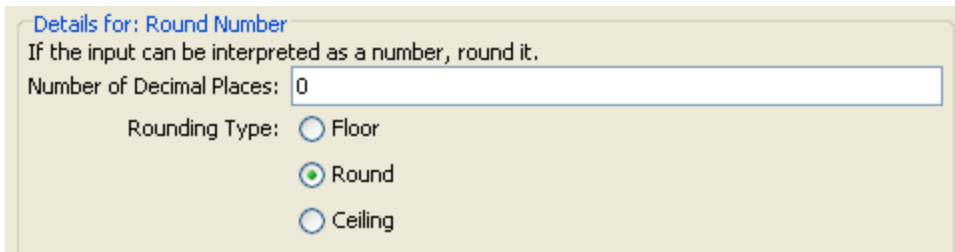
With:

Ignore case: ☐

1. In the **Find** box, type the string to search for and replace.
2. From the **Replace**, select **First**, **All**, or **Last**, depending on which instances of the target string you want to replace.
3. In the **With** box, type the string to replace the target string with.
4. To make the search not case-sensitive, select the **Ignore case** check box.

## Round Number

This filter rounds numbers up or down.



Details for: Round Number  
If the input can be interpreted as a number, round it.

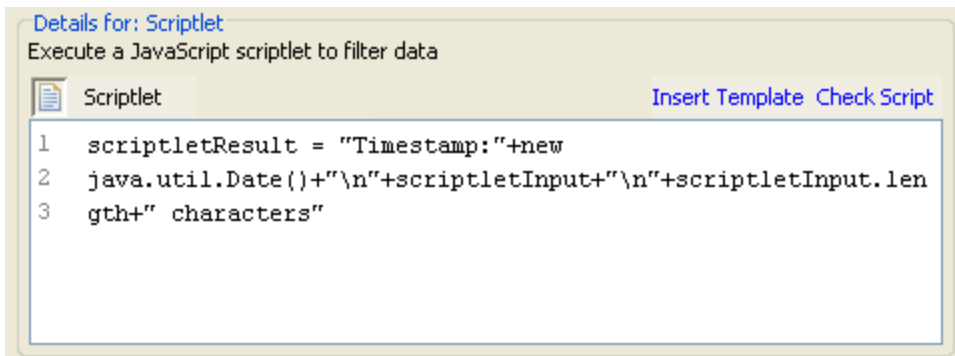
Number of Decimal Places:

Rounding Type: ☐ Floor ☒ Round ☐ Ceiling

1. To specify the accuracy of the rounding, in the **Number of Decimal Places** box, type the number of decimal places the number should be rounded to.
2. For **Rounding Type**, specify in which direction the number should be rounded:
  - **Floor** always rounds the number down
  - **Ceiling** always rounds the number up
  - **Round** rounds the number up if the last meaningful place is 5 or more, and down otherwise

## Scriptlet

This filters data with a scriptlet that you create.



1. To obtain scriptlet lines that you will need for the scriptlet to work as a filter, click **Insert Template**.

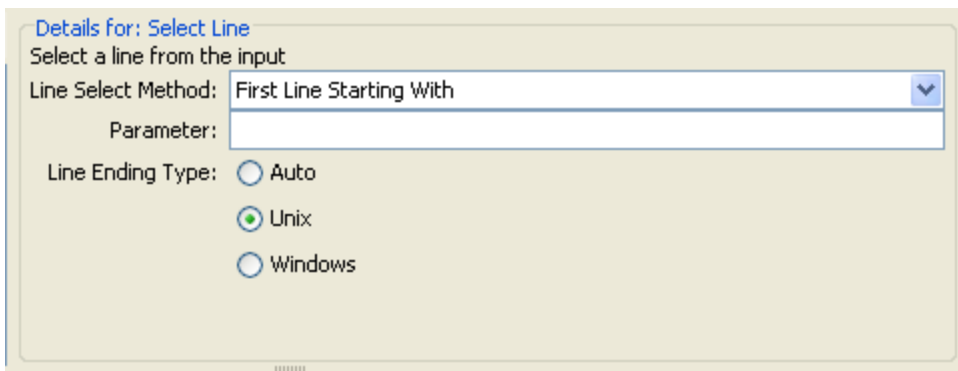
The template that is inserted is specific to the language that you chose and includes the most commonly used commands for accessing flow variables, values of global variables, operation results, and inputs, and for setting and manipulating flow variable values and results.

2. To debug the scriptlet, click **Check Script**.

For more information about scriptlets, see ["Using Scriptlets in a Flow" on page 200](#).

## Select Line

This filter defines a line that you want to extract from the raw results.



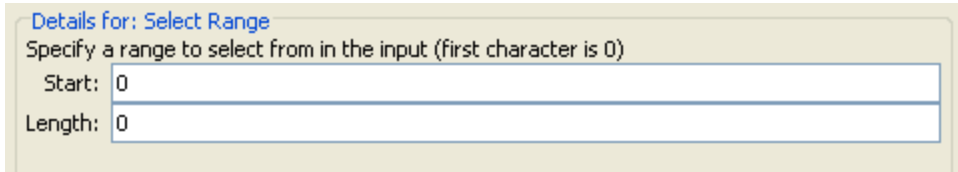
1. From the **Line Select Method** list, select a criterion for the line that you're interested in.
2. In the **Parameter** box, type a string that the line contains.
3. From the **Line Ending Type** group, select one of the following:
  - If the text that you're filtering was generated on a Unix operating system (which ends lines with LF), select **Unix**.
  - If the text that you're filtering was generated on a Windows operating system (which ends lines with CR/LF), select **Windows**.

- To enable the filter to accept either type of line ending, select **Auto**.

**Auto** is the default selection.

## Select Range

This filter defines a string that you want to extract from the input data. The two criteria for defining the string are its length in characters and the position of its first character from the start of the input data.



Details for: Select Range  
Specify a range to select from in the input (first character is 0)  
Start: 0  
Length: 0

1. In the **Start** box, type the zero-based start position of the string.
2. In the **Length** box, type the number of characters in the string.

Keep in mind that a new line may count as one or two characters, depending on the operating system from which you obtain the data you're filtering.

## Sort

This filter orders the lines of input data by the first character in each line.



Details for: Sort  
Sort the input based on newlines.  
Ascending: ☒  
Treat as Numbers: ☐

1. To specify the direction of the sort:
  - For ascending order, leave the **Ascending** check box selected.
  - For descending order, clear the **Ascending** check box.
2. To order the data by ASCII order, select the **Treat as Numbers** check box.

Note that ascending ASCII order is roughly as follows, for English characters:

- White space
- Symbols
- Numbers
- Alphabetic characters

## Strip

This filter strips characters from the beginning or end of the raw results.

**Note:** If this filter follows other filters, the characters are stripped from the beginning or end of the subset of the raw results that was obtained by the processing of preceding filters.

**Details for: Strip**  
Remove a matching string from the head or tail of the input

Strip Method: All Characters Up To And Including

Characters to Strip:

- From the **Strip Method** list, select how you want the filter to strip the raw results. You can specify the following options for stripping the string that you specify in the **Characters to Strip** text box:
  - **All Characters Up To** the string
  - **All Characters Up To And Including** the string
  - **All Characters After** the string
  - **All Characters After And Including** the string
- In the **Characters to Strip** text box, type the string to find.

## Strip Whitespace

This filter removes all the whitespace characters from the front and the end of the raw results.

## Table

A table filter does not convert the raw results into a table, but enables you to manipulate the raw results as if they were a table, including sorting columns and selecting columns, rows, and blocks.

**Details for: Table**  
Parses the input as a table and sorts it on a specified column

Column Delimiter: Whitespace Row Delimiter: NewLine

First Row is Header: ☐ Strip First Row of Result: ☐

Sort On Column: -1 Ascending: ☐

Select Row: 0 Select Col: 0

Select Width: 1 Select Height: 1

**Note:** Row numbering is 0-based (starts with 0 [0]), and column numbering is 1-based.

1. In the **Column Delimiter** list, choose the character that will serve to divide the data into columns in a meaningful way.
2. In the **Row Delimiter** list, choose the character that will serve to divide the data into rows in a meaningful way.

**Note:** Two or more consecutive whitespaces count as a single whitespace, so a column may be occupied by data that you expected to find in a column to the right. For example, this behavior will appear if you apply this filter to the output of a “dir” command-line command with whitespace specified as the column delimiter.

3. To treat the members of the first row as column headers, select **First Row is Header**.
4. To remove the first row, select **Strip First Row of Result**.
5. To sort on a column, type the (1-based) column number in the **Sort On Column** box.

**Tip:** The value **-1** means do not sort on any column.

6. To specify ascending order, select the **Ascending** box.

By default, the sort order is descending.

7. To select a row you want the filter to extract:

- In the **Select Row** box, type the (0-based) row number.

**Tip:** **-1** selects all the rows in the data.

- In the **Select Width** box, type the number of columns in that row that you want extracted.

**Tip:** **-1** selects all the remaining columns in the data to the right of the column specified in **Select Col**.

8. To select a column you want the filter to extract:

- In **Select Col**, type the column number.

**Tip:** **-1** selects all the columns in the data.

- In the **Select Height** box, type the number of rows in that column that you want extracted.

**Tip:** -1 selects all the remaining rows in the data below the row specified in **Select Row**.

For example, to extract the first 5 rows of the 2<sup>nd</sup> through 4<sup>th</sup> columns, you would specify the following. In these settings, the first two settings define the rows selected, and the second two settings define the rows selected.

- In **Select Row**: 0
- In **Select Height**: 5
- In **Select Col**: 2
- In **Select Width**: 3

## XML filters

XML filters enable you to parse XML within a step, the XML being obtained from the step's input or result, rather than having to create a flow and pass the XML to one of the XML-processing operations in HP OO default content.

Using XML filters in an operation and using the XML-processing operations in default content differs in several respects.

- There is the difference between accomplishing the task within an operation and using the infrastructure of a flow to accomplish the task
- Filters within an operation have some limits that the XML-processing operations do not have. These limitations are described in the following sections on the particular filters. Whether you choose to filter input XML with a filter or an operation may depend on how you obtain the XML.

Following are the XML filters:

- XML Get Attribute
- XML Get Element
- XML Get Element Value
- XPath Query

For illustration of the XML filters, the examples reference the following XML example:

```
<?xml version="1.0" encoding="utf-8"?>
<tickets>
  <ticket id="1448" severity="3">
    <customer firstName="John" lastName="Doe">
```

```
<volume>30000</volume>
<company>myOrg</company>
<position>CIO</position>
<contactInfo>
  <email>jdoe@myorg.com</email>
  <email>johnsSecondEmail@myorg.com</email>
  <mobile>12065551212</mobile>
  <description internal="1">Private contact info</description>
  <description>Partial contact info</description>
</contactInfo>
<description>Our best customer</description>
</customer>
<details>
  <description>A simple Test xml</description>
  <comment user="john"> Initially raising ticket</comment>
  <comment user="frank"> Problem diagnosed, not a real issue</comment>
  <comment user="albert">ok, I'm going to close it.</comment>
  <state>Closed</state>
</details>
</ticket>
<ticket id="1886" severity="5">
  <customer firstName="Elaine" lastName="Benson">
    <volume>50000</volume>
    <company>herCompany</company>
    <position>CEO</position>
    <contactInfo>
      <email>ebenson@herco.com</email>
      <mobile>011445551212</mobile>
      <description internal="1">Private contact info</description>
      <description>Partial contact info</description>
    </contactInfo>
    <description>Our other best customer</description>
  </customer>
  <details>
    <description>datastream bug</description>
    <comment user="jack">Customer found bug.</comment>
    <comment user="elsbeth">It is a third-party supplier bug.</comment>
    <state>Closed</state>
  </details>
</ticket>
</tickets>
```

## XML Get Attribute

The **XML Get Attribute** filter extracts the value for each of one or more instances of the attribute that you specify. In the Filter Editor, you can control which instance of the attribute the filter is applied to by specifying an element path to the attribute.



You can obtain the value for a single instance of the attribute or for multiple instances, returned in a table. In such a table, the columns are comma-delimited and the rows are new line-delimited.

**Details for: XML Get Attribute**  
Filters an xml document for the requested attribute value. For advanced XML filtering use the XPath Filter.

Element Path:

Include sub-elements: ☐

Attribute Name:

Result: ☒ Single Match  
☐ As Table

1. In the **Element Path** box, specify the path of the element that contains the attribute whose value you want to extract. Use forward slashes to separate the parts of the path to the element.

To control which instance of the element the filter gets the attribute's value from, add a specification such as [2] or [3]. The numbering of elements is 1-based (starts with [1]). Thus to specify the second instance of an element, you would use [2].

2. To search child elements of the element you've specified, select the **Include sub-elements** check box.
3. In the **Attribute Name** box, type the name of the attribute whose value you want.
4. For **Result**, select one of the following:
  - To restrict the result extracted to the value of a single instance of an attribute, select **Single Match**.
  - To extract the value of all the instances of the attribute you have named, select **As Table**.

**Example:** To find the name of a user for one of the comments (using the example XML from the topic *XML Filters*):

In the **Element Path** box, type `/ticket/details/comment`.

**Example:** To get the name of the user for a particular comment (in this example, the second one):

1. In the **Element Path** box, type `/ticket/details/comment[2]`.
2. In the **Attribute Name** box, type `user`.
3. Beside **Result**, select **Single Match**.

The output will be john.

**Example:** To find the name of the user for each comment:

1. In the **Element Path** box, type **/ticket/details/comment**.
2. In the **Attribute Name** box, type **user**.
3. Beside **Result**, select **As Table**.

The output will be:

Path,user

/ticket/details/comment[1],john

/ticket/details/comment[2],frank

/ticket/details/comment[3],albert

## XML Get Element

The **XML Get Element** filter enables you to extract an element in its entirety (including child elements, values, and attributes) by describing it in any of the following ways:

- By a relative or absolute path.
- By a child element of the element you want to extract. You can also search by a specific value of the child element.
- By an attribute of the element you want to extract. You can also search by a specific value of the attribute.

Details for: XML Get Element  
Filters an xml document for elements given a path. For advanced XML filtering use the XPath Filter.

Element Path:	<input type="text"/>		
Child Named:	<input type="text"/>	Value:	<input type="text"/>
Attribute Named:	<input type="text"/>	Value:	<input type="text"/>

In the following procedure, you can enter specifications in any one or combination of the text boxes.

1. In the **Element Path** box, type an absolute path to the element.

Within the path, a relative path indicator indicates location relative to the element that precedes the relative path indicator.

- **../** specifies the parent of the last-named element.
- **./** specifies the last-named element.

**Example:** in the XML example, <volume> and <company> are sibling elements, both children of the <customer> element. You could specify the <company> element with the following relative path:

```
/tickets/ticket/customer/volume/../company
```

If there is more than one instance of the identified element, simply specifying the path, as in the preceding example returns all the instances of the element.

You can specify a particular instance of any element in the path with an integer inside square brackets.

**Example:**

/tickets/ticket/details/comment specifies all the comments in the details for all the tickets.

/tickets/ticket/details/comment[2] specifies the second comment for each ticket.

/tickets/ticket[2]/details/comment specifies all the comments for the second ticket.

2. In the **Child Named** box, type the name of an element that is a child of the element (or elements) that you want to extract. If the child element has a value, you can narrow the results by typing that value in the **Value** box.
  - The **Child Named** box works for only one level of child elements. The filter only returns the direct parent of the child element that you specify.
  - The **Value** box is intended for brief values. The value that you type there must be an exact match of the value of the child element of the element that you want to extract.
3. In the **Attribute Named** box, type the name of an attribute that is unique to the element you want to extract. To further narrow the results, you can type a value of the attribute in the **Value** box.

**Example:** In the example XML, there are several ways to extract the customer element and all of its contents:

- In the **Element Path** box, type **/ticket/customer**
- In the **Child Named** box, type any of the child elements of customer:

**company**

**position**

### contactInfo

If you type **company** in the **Child** box, then in the accompanying **Value** box, you could also type **myOrg**

- In the **Attribute named** box, type one of the following:

**firstname**

**lastname**

In the accompanying **Value** box, you could type the respective value for those attributes:

**John**

**Doe**

For each of these filters, the output is the customer element, as follows:

```
<customer firstName="John" lastName="Doe">
  <company>myOrg</company>
  <position>CIO</position>
  <contactInfo>
    <email>jdoe@myorg.com</email>
    <email>johnsSecondEmail@myorg.com</email>
    <mobile>12065551212</mobile>
    <description internal="1">Private contact info</description>
    <description>Partial contact info</description>
  </contactInfo>
  <description>Our best customer</description>
</customer>
```

## XML Get Element Value

The **XML Get Element Value** filter enables you to get the value of a specific element.

### Details for: XML Get Element Value

Filters an xml document for the first element that matches a given path and returns its value.

Element Path:

In the **Element Path** box, type the path to the element whose value you're interested in.

As with the other filters, if there are multiple instances of an element, the filter returns the first one, unless you specify a different instance.

**Example:** Using the sample XML

To get the value for the email element, type **/tickets/ticket/customer/contactInfo/email**

The output will be one of the two emails given:

jdoe@myorg.com

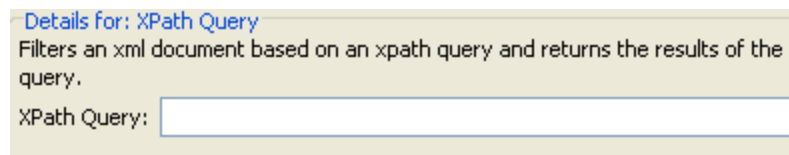
johnsSecondEmail@myorg.com

To specify a particular instance of the email element, type  
**/ticket/customer/contactInfo/email[2]**

The output will be: johnsSecondEmail@myorg.com

## XPath Query

The **XPath Query** filter enables you to extract data from the result with queries that use the standard XPath syntax, which you type in the **XPath Query** box.



In the **XPath Query** box, type the query, using XPath syntax.

- The path that precedes the square brackets identifies the scope of the query with which you are narrowing the results.
- Square brackets contain the filtering portion of the query. There can be more than one set of filters in a query.

### Example: Using the sample XML

You can extract a customer who has a volume of more than 40,000 of some units, with either of the following queries:

- This XPath query finds all the companies whose customer's volume is more than 40,000.

```
/tickets/ticket/customer/company[../volume>40000]
```

The <volume> element is a sibling of the <company> tag, so to locate the element <volume>, you use the following sequence inside the square brackets to articulate the path relative to <company>:

```
../
```

- This XPath query finds all the customers whose volume is greater 40,000 units.

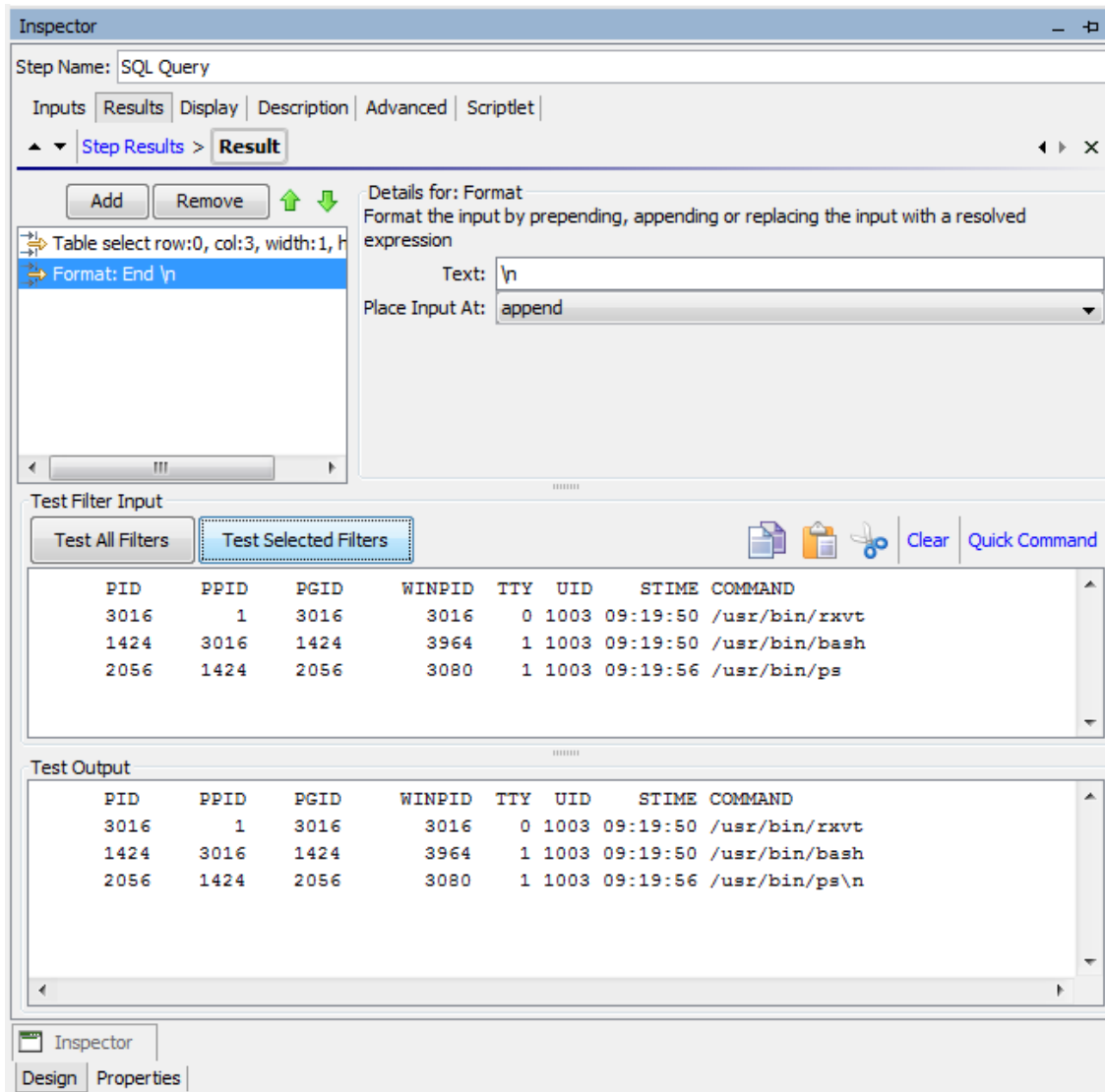
```
/tickets/ticket/customer[volume>40000]
```

Because <volume> is a child of <company>, you do not need to specify its relative path.

## Reference Material


### Filter Editor

The **Filter** list in the upper-left displays a list of the filters as you create them.



When you create a filter and have selected a filter type, the **Details for:** section in the upper-right changes to show controls for modifying filters, depending on the kind of filter you select.

GUI item	Description
<b>Add</b>	Click to add a new filters.
<b>Remove</b>	Click to delete the selected filter or filters.

	Click to move the selected filter up or down in the list. Filters are processed in the order that they appear in the list.
<b>Test Filter Input</b>	This is where you place data in order to test that the filter works as expected.
<b>Test All Filters</b>	Applies the test to all the filters on the output or result.
<b>Test Selected Filters</b>	Applies the test to the selected filters.
<b>Copy</b>	Copy data within the <b>Test Filter Input</b> box.
<b>Paste</b>	Paste data into the <b>Test Filter Input</b> box.
<b>Cut</b>	Cut data within the <b>Test Filter Input</b> box.
<b>Clear</b>	Clear data within the <b>Test Filter Input</b> box.
<b>Quick Command</b>	Type a command that generates the data on which you want to test the filter. The output of the command appears in the <b>Test Filter Input</b> box.
<b>Test Output</b>	After filters are applied to the test data in the <b>Test Filter Input</b> box, the filtered results appear in the <b>Test Output</b> box.

## Working with Variables

You can use variables to move data within and between flows.

For example, if you have several steps that act on a server, you can have the first step get the IP address of a server and assign that value to a flow variable. Then, any subsequent step that has an input of that name will automatically use that server name.

### *Flow Variables*

Flow variables are available only for the flow within which they are defined.

### *Assigning Value to Flow Variables*

You can assign a value to a flow variable from:

- **A step's result** - for example, a step with an operation to count hits will store the result in a flow variable
- **An input value** - for example, a step that gets an IP address as an input value will store the address as a flow variable
- **A scriptlet** - for example, a scriptlet that evaluates data that is returned from a step's operation will store the data in a flow variable

## Using Flow Variables

You can reference a flow variable and the data that it stores in any of the following places:

- **In a different step** in the same flow
- **Within a lane in a parallel split step** – a lane step can use the value of a flow variable, if that value was written to the flow variable by an earlier step in the same lane or prior to the parallel split step. However, a step in one lane cannot use a flow variable value, if this value was written to the variable by a step in a different lane.
- **In an operation input**
- **In flow, step, and transition descriptions** – for example, the **Ping Latency** operation filters out the average duration of the ping. A step associated with this operation could save the average duration as the flow variable latency, then the transition that follows this step could report that value to the user
- **As part of data you are testing with a response rule** – for example, to see whether an output string or error string contains a value that you have stored in a flow variable.
- **In scriptlets** – to make a scriptlet result available outside of the step, the scriptlet must create a flow variable (if the desired one doesn't already exist), and assign the result to it.
- **In operation parameters** – If an operation parameter takes a value, you can access that value by referencing a flow variable that contains it.

The **Flow Variables** pane helps you to keep track of the flow variables that you have created.

## Global Variables

Global variables are keyname-and-value pairs that are part of the global context, and so are always available for use or reference in any flow run.

If a flow variable and a global variable have the same name, a reference to that variable name accesses the (local) flow variable of that name, not the global variable. This is the case either for assigning a value to the variable or for getting its value.

When you specify that an input gets its value from a global variable, a flow variable is created with the value of that global variable, and the value is supplied from the flow variable to the input.

## Best Practices

- Be consistent about case. For example, use camel case for all flow variable names.
- Use naming conventions for different types of flow variable. For example, add prefixes to variable names, based on the variable type, such as FI for flow input, SI for step input, OI for operation input, and so on.




- Be aware that flow variables are accessible to the entire flow. Pay attention to flow variable manipulation, because data can accidentally be altered in one step and then used incorrectly in the flow's subsequent steps.

## What do you want to do?

### Assign a value to a flow variable from an input

By default, the value of the input is assigned to a flow variable with the same name as the input.

1. Open the **Properties** sheet (for an operation) or the Step Inspector (for a step).
2. On the **Inputs** tab, select an input or create a new one.
3. Click the right-pointing arrow  at the end of the row to open the Input Editor.
4. In the **Assign to Variable** box, name the variable to which you want the value to be assigned.
5. Save.

For more information about creating an input, see ["Creating Input" on page 120](#).

### Assign a value to a flow variable from a result

1. Open the **Properties** sheet (for an operation) or the Step Inspector (for a step).
2. On the **Results** tab, select the row of the relevant result.
3. From the **Assign To** list, select **Flow Variable**.
4. Under **Name**, specify the name of the flow variable.
5. Under **From**, specify the source of the value.

For more information about creating a result, see ["Creating Outputs and Results" on page 149](#).

If necessary, to obtain the precise results you want, create one or more filters for the result. See ["Filtering Output and Results" on page 158](#).

### Assign a value to a flow variable from a scriptlet

You can also create and assign a value to a flow variable from a scriptlet.

In the scriptlet, include a command with the following syntax:

```
scriptletContext.putLocal("<localflowvariablename>", <value>);
```

where <value> can be a variable or an object created within the scriptlet.

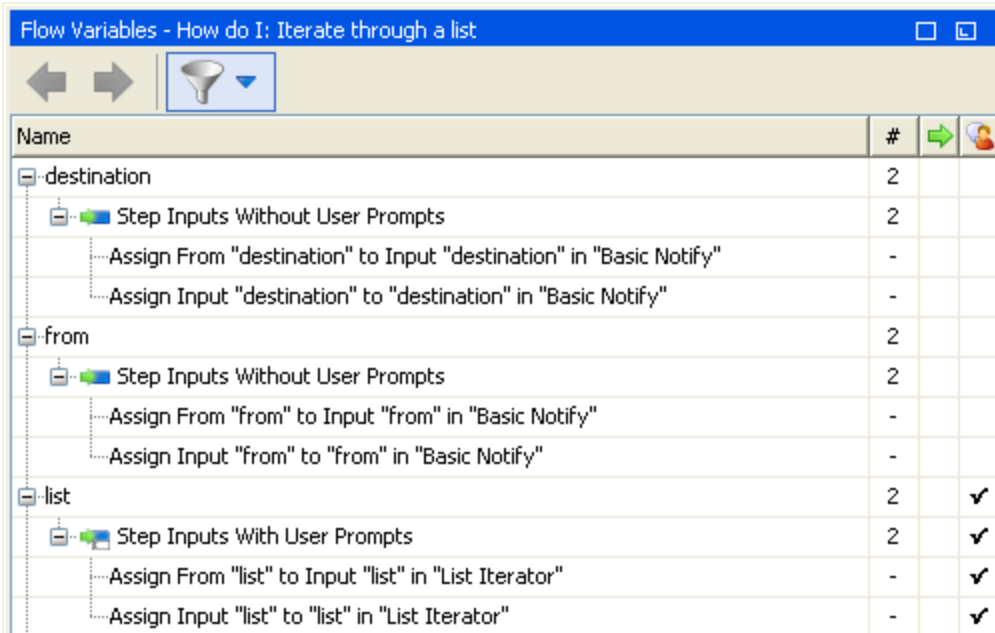
### View information in the Flow Variables pane

The **Flow Variables** pane helps you track the storage of data in flow variables:

- How the flow uses flow variables to make data available where it's needed
- Where the flow variables obtain their data

The **Flow Variables** pane presents this information in a tree structure. The **Flow Variables** pane shows all the flow variables used in the current flow, listing each flow variable's creation and/or usages. Any changes you make to a flow variable in the flow are automatically reflected in the **Flow Variables** pane.


Click the **Flow Variables** tab in the upper-right of the Studio window to open the **Flow Variables** pane.



Name	#	Filter	Info
destination	2		
Step Inputs Without User Prompts	2		
Assign From "destination" to Input "destination" in "Basic Notify"	-		
Assign Input "destination" to "destination" in "Basic Notify"	-		
from	2		
Step Inputs Without User Prompts	2		
Assign From "from" to Input "from" in "Basic Notify"	-		
Assign Input "from" to "from" in "Basic Notify"	-		
list	2	✓	
Step Inputs With User Prompts	2	✓	
Assign From "list" to Input "list" in "List Iterator"	-	✓	
Assign Input "list" to "list" in "List Iterator"	-	✓	

### Filter information in the Flow Variables pane

To zero in on the flow variable uses that you're most interested in, you can select which flow variable uses you want to display in the pane.

1. Display the filter buttons by clicking the **Filter** button  in the **Flow Variables** toolbar.
2. In the row of buttons that appears, click the buttons to toggle each filter type on or off. As you toggle each type of data source, that type is displayed or removed from the display:
  - Flow input
  - User-prompt step input

**Note:** User prompts are not supported in this version of HP OO.


- Step input that does not have a user prompt

- Result
- Scriptlet

## Locate the input that the flow variable listing references

To view a particular use and open the editor that defines that use, select the use instance in the **Flow Variables** pane.

- If the use is for a flow input, the flow's **Properties** sheet opens on the **Inputs** tab, with the Input Editor for the relevant input open.
- If the use is for a step input or result, the flow diagram opens with the step selected. Beneath the flow diagram the editor for the input or result is opened.

**Tip:** Use the **Previous**  and **Next**  buttons to move up or down in the list of uses.

## View a global variable

To view all the global variables in a flow, debug the flow. The global variables (as well as the flow variables) and their current values are listed in the Context Inspector.

For more information on the Context Inspector, see ["Validating Content" on page 209](#).

## Change a global variable

**Important!** Before you change the value of a global variable, remember that global variables are available in any run of any flow. Changing the value of a global variable will affect other flows and operations that use that global variable.

To change a global variable, complete the task *Assign a value to a flow variable from an input*. In the **Assign to Variable** box, enter the name of the global variable to which you want the value to be assigned.

## Make flow variables global by default

It is possible to set up a flow so that all the flow variables in it are global by default, unless they occur in a subflow.

1. Display the Flow Properties sheet for the flow.
2. Select the **Advanced** tab.
3. Select the **Make flow variables global when not a subflow** check box.

# Reference Material

## Flow Variables pane

When you have a flow open in the authoring pane, the **Flow Variables** pane lists each flow variable

in alphabetical order, and describes each use of the flow—each place in the flow where the flow variable may be used.

Flow Variables - How do I: Iterate through a list			
<div> </div>			
Name	#		
destination	2		
<div>  Step Inputs Without User Prompts                 </div>	2		
Assign From "destination" to Input "destination" in "Basic Notify"	-		
Assign Input "destination" to "destination" in "Basic Notify"	-		
from	2		
<div>  Step Inputs Without User Prompts                 </div>	2		
Assign From "from" to Input "from" in "Basic Notify"	-		
Assign Input "from" to "from" in "Basic Notify"	-		
list	2	✓	
<div>  Step Inputs With User Prompts                 </div>	2	✓	
Assign From "list" to Input "list" in "List Iterator"	-	✓	
Assign Input "list" to "list" in "List Iterator"	-	✓	






GUI item	Description
	Click to move up in the list of uses.
	Click to move down in the list of uses.
	Click to display the filter buttons, in order to filter the information visible in the in the <b>Flow Variables</b> pane.
#	Displays the number of times the flow variable is used in the flow.
	Is checked when a particular use of the flow variable occurs in a flow input.
	Is checked when a particular use of the flow variable gets its value from user input.

## Filter buttons in the Flow Variables pane

Display the filter buttons by clicking the **Filter** button in the **Flow Variables** toolbar.

Click the filter buttons to toggle each filter type on or off. As you toggle each type of data source, that type is displayed or removed from the display.

Filter button	Description
---------------	-------------




	Flow input—the flow variable is referenced in an input
	Step inputs with user prompts
	Step inputs without user prompts  <b>Note:</b> User prompts are not supported in this version of HP OO.
	Results—the flow variable is associated with a step result
	Scriptlets—the flow variable is referenced in a scriptlet

## Creating Return Steps

A flow needs one or more return steps to end the flow.



The four types of return step are:

- **Resolved**  – This is the standard return step for a flow that runs correctly.
- **Diagnosed**  – This return step indicates that a flow has determined what a problem is and has opted not to take action on it other than notification.
- **No Action Taken**  – This return step is used when a remediation flow gathers data but cannot determine any diagnosis or remediation.

**Note:** A flow that is intended solely to gather data should return **Resolved**  when it is

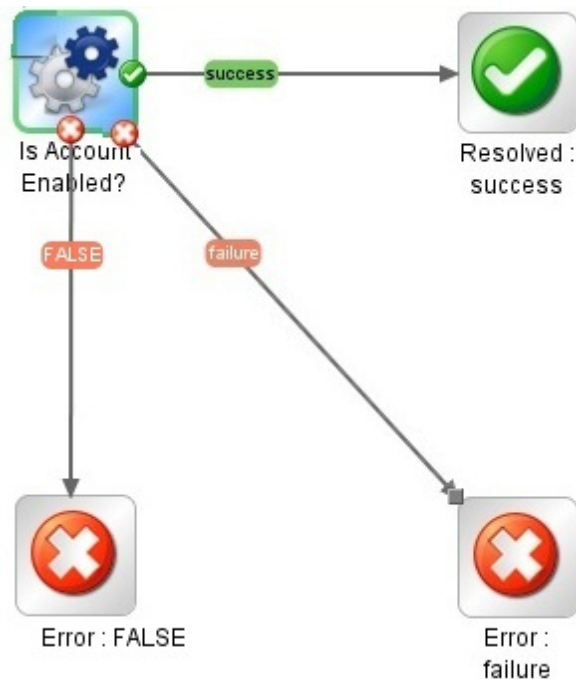
complete, rather than **No Action Taken** 🚫.

- **Error** 🚫 – This return step is used if the flow fails to run all the way to the end. For example, because of bad input, failure to reach a system, or a problem with the flow.

In each return step name, the response of the return step is shown after the colon; for example, **Error: failure**. You can modify this response. For example, if the outcome that led to an **Error: failure** return step was not a failure in an operation but a result that did not meet a required threshold, then you might want to create a new response for the **Error: failure** step that reflects this outcome, such as **Error: threshold not met**.

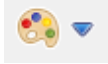
## Best Practices

- If you have multiple end steps of the same type in a flow (for example, multiple error end steps), rename the end steps to include the cause of the failure.
- Avoid confusing a failed operation with a negative result. For example, if an operation asks a question for which the answer may be TRUE or FALSE, a FALSE answer is not the same as a failure. In such a case, you need two **Error** return results, one for a FALSE result and one for a failure of the operation.




## What do you want to do?

### Add a return step to a flow

1. On the authoring pane toolbar, click the **Step Palette** button  to display the **Step** palette.
2. From the **Step** palette, drag the appropriate return step icon to the authoring canvas.
3. Create transitions from the flow steps to the return step.

### Change a return step's response

You can change the response of a return step to more accurately reflect the outcome that led to the return step. For example, if your flow has multiple error responses ( **Error: failure** and **Error: threshold not met**), when you drag the **Error**  icon to the authoring canvas, the error return step may not contain the response that you want.

1. Right-click the return step on the authoring pane and select **Select Response**.
2. Select the response that you want for the return step. For example, **Error: threshold not met**.

### Create and assign a new response to the return step

If the list of available responses does not include the response that you need, you can create a custom response.

1. Right-click the return step on the authoring pane and select **Select Response**.
2. Select **Add New Response**.
3. In the dialog box, enter a name for the new response and click **OK**.

## Reference Material









### Step palette

The **Step** palette contains buttons for dragging return steps, parallel split steps, multi-instance steps, and callouts onto the flow. Display the **Step** palette by clicking the **Step Palette** button



from the authoring pane toolbar.



Button	Description
<b>Success</b> 	Lets you drag a <b>Success</b> return step to the flow.
<b>Diagnosed</b> 	Lets you drag a <b>Diagnosed</b> return step to the flow.
<b>No Action Taken</b> 	Lets you drag a <b>No Action Taken</b> return step to the flow.
<b>Failure</b> 	Lets you drag a <b>Failure</b> return step to the flow.
<b>Parallel Split Step</b> 	Lets you drag a parallel split step to the flow.
<b>Multi-instance Step</b> 	Lets you drag a multi-instance step to the flow.
<b>Callout</b> 	Lets you drag a callout to the flow, providing information to users.
<b>Docking bar</b> 	Click to dock and undock the palette.



## Advanced Authoring

This chapter covers creating more complex flows. For information about creating simple flows, see ["Authoring a Flow – Basics" on page 99](#).

When creating flows, make sure not to create flows that create unlimited growth in memory. For example, do not create a flow that runs an infinite loop, in which the flow sleeps, performs some tasks, then goes back to sleep. In this case, the Run History grows until the system runs out of memory.

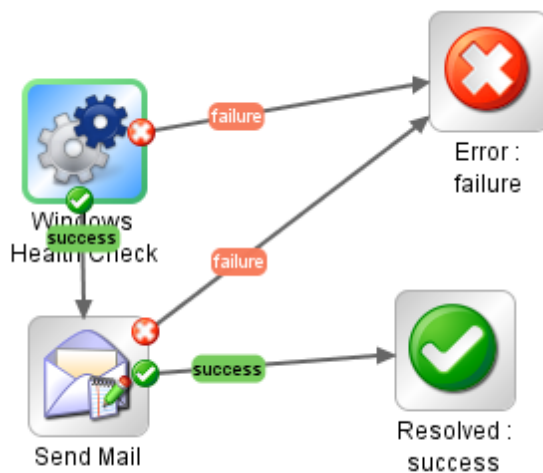
Creating a Subflow Within a Flow .....	185
Creating a Flow with Parallel Split Steps .....	188
Creating a Flow with Multi-Instance Steps .....	192
Using Scriptlets in a Flow .....	200
Using Regular Expressions in a Flow .....	204

## Creating a Subflow Within a Flow

You can simplify a flow by creating steps from subflows. This way, you can:

- Separate the programming tasks into smaller, more manageable pieces
- Test parts of the flow individually
- Reuse the pieces that you create

For example, in the flow below, the **Windows Health Check** step is a subflow.



A subflow is treated as a single step even though it may contain many operations.

Subflows often generate data that steps in the parent flow need to access. Flow variables that you create within a flow cannot be referenced outside that flow. However, you can pass values from a subflow to a parent flow, by saving the subflow results as a **flow output field**.

## Best Practices

- A flow should fit on the canvas on a 1024 x 768 screen with Studio maximized and a 1:1 view magnification. Larger flows are not strictly prohibited, but if a flow is larger, examine it carefully to see whether you can break down some of its sequences of steps into subflows.
- Supply a description and name for all transitions on a top-level parent flow. These transition descriptions should describe what happened in the step that preceded the transition. You need not add descriptions of transitions in a subflow unless the data is critical to see during a run.

## What do you want to do?

### Create a flow with a subflow

1. Create a flow first and save it.
2. Create a new flow, to act as parent flow.
3. Drag the subflow from the **Projects** pane onto the parent flow, to create a step from the subflow.

### Pass data from a subflow to a parent flow

1. Open the subflow on the authoring canvas, and open the Step Inspector for the step whose data you want to make available to the parent flow.
2. Click the **Results** tab and add a result (for more information, see ["Setting Step Results" on page 153](#)).
3. Configure the result so that result data is stored in a **flow output field**. This makes the data available outside of the subflow.

Name	From	Assign To	Assignment Action	Filters
FailureMessage	Result Field: Result	Flow Output Field	OVERWRITE	No Filters

- a. Under **Name**, type a name for the flow output field.
- b. Under **From**, select **Result Field: Result**.
- c. Under **Assign To**, select **Flow Output Field**.

- d. If required, create a filter to filter the result (for more information, see ["Filtering Output and Results" on page 158](#)).
4. In the parent flow's authoring canvas, open the Step Inspector for the step that was created from the subflow.
5. Click the **Results** tab and create a step result. By default, this new result:
  - Obtains its value from the result field that has the name of the subflow's flow output field
  - Has the same name as that of the subflow's flow output field
  - Is assigned to a flow variable, which by default has the same name as the result, and which is now available for use in transitions and steps that follow this step

## Example

1. Copy a command operation, and have it execute "dir C:\". Name it **dir**.
2. Create a flow called **flowdir**.
3. In the **flowdir** flow, create a step using the operation **dir**.
4. On the **dir** step, add a result that comes from the operation's output string.
5. Assign the result to a flow output field, and name the result **foo**. Now the flow has a flow output field also named **foo**.
6. Create another flow called **parentflow**.
7. In **parentflow**, create a step from **flowdir**.
8. Add a result to the **flowdir** step.

By default, the new result is named **foo**. It obtains its value from the **Result Field: foo**, and the value is assigned to a flow variable also named **foo**. The result **foo** of the subflow's step **dir** is now available to transitions and steps that come after the **flowdir** step in the parent flow.

9. To test this, after the flowdir step, add a step created from the **Basic Notify** operation.
10. In this new step:
  - a. Define the `notifyData` input as being a single value, that uses a constant value, and specify that the constant value is `${foo}`.
  - b. Define the `notifyMethod` input as being a single value, which uses a constant value, and specify that the constant value is `Display`.

- c. Define the subject input as being a single value, which uses a constant value, and specify that the constant value is something like: If this worked, the flow output field says: "contents of outputString, aka foo".
- d. Debug the flow.

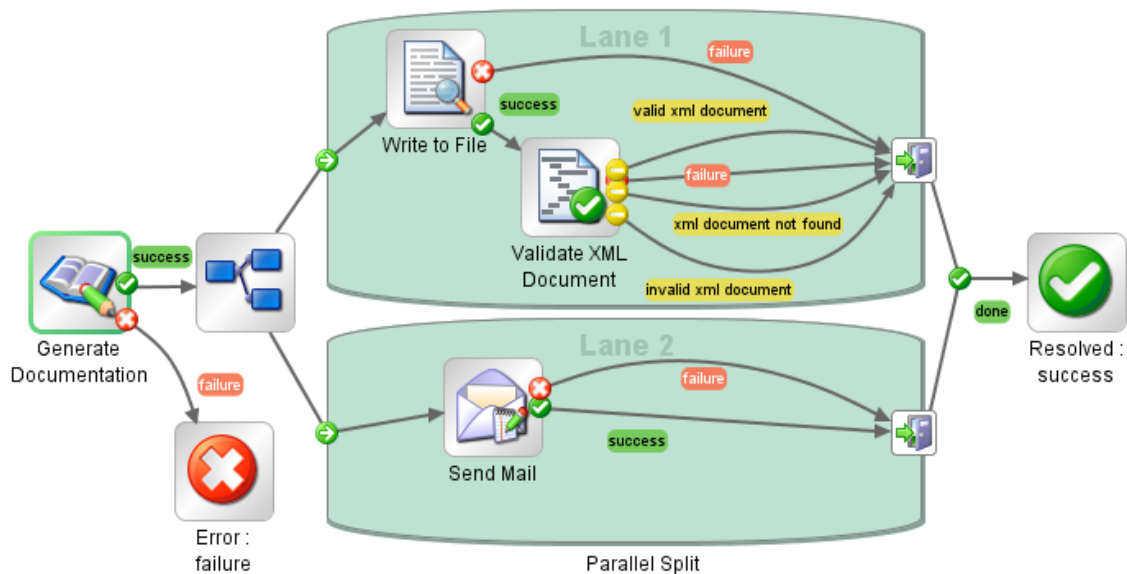
## Creating a Flow with Parallel Split Steps

A parallel split step is a set of step sequences that are carried out simultaneously. Each series of steps is represented visually in the flow diagram as a lane. The steps contained in each lane are called "lane steps". When you run the flow, the lanes start simultaneously.

Parallel split steps are best used for doing dissimilar things simultaneously and independently of each other. Note the contrast with multi-instance steps, whose instances do the same thing with multiple variations of a single input.

For example, you might use a parallel split step for writing and validating an XML file, and, at the same time, sending email about this to the appropriate person:

- One lane contains the steps for writing and validating the file.
- The second lane sends the email.







## What do you want to do?

### Create a parallel split step

1. On the authoring pane toolbar, click the **Step Palette** button  to display the **Step**

palette.


2. From the **Step** palette, drag the **Parallel Split Step**  icon to the authoring canvas. By default, the step has two lanes.
3. Create the step sequence you want within each lane.
  - a. Add steps (flows or operations) in the lane.
  - b. Connect the steps within each lane.
  - c. Connect the last step in the lane to the **Lane-end** icon .
4. Connect the multi-instance step to the rest of the flow:
  - a. If the parallel split step is not the start step, connect the step that precedes it to the **Parallel Split Step**  icon.
  - b. Connect the parallel split step's **done**  response to the next step in the flow.

## Change the visual order of lanes


You can change the visual order of the lanes in the flow diagram, but note that when the flow is run, all the lanes begin at the same time. Their graphical order in the flow diagram does not affect the order in which their processing occurs.

1. Right-click the lane that you want to move.
2. From the drop-down menu, select **Move Lane Up** or **Move Lane Down**.

## Move a parallel split step or its components



- To move a parallel split step, click the **Parallel Split Step**  icon in the flow diagram, and drag.
- To move an individual lane step, select the step and drag, either within the lane or to another lane.

## Copy a parallel split step

1. Right-click the **Parallel Split Step**  icon in the flow diagram, and select **Copy**.
2. Right-click on the authoring canvas and select **Paste**.

## Copy components of a parallel split step

To copy components of the parallel split step, use any of the following tools:

- The **Edit > Copy** and **Edit > Paste** menu commands
- The right-click menu
- Keyboard combinations CTRL+C, CTRL+V
- The **Copy**  and **Paste**  buttons on the authoring pane toolbar

**Note:** If you are copying a lane, keep the cursor inside the lane when you carry out the **Paste** command.

## Add a new lane

1. Right-click an existing lane.
2. From the drop-down menu, select **Add Lane**.

A new, empty lane is added below the currently selected lane.


## Duplicate a lane

1. Right-click an existing lane.
2. From the drop-down menu, select **Duplicate Lane**.

A new lane with the same title as the one you copied appears directly below it.

## Delete a lane

To delete a lane, use any of the following tools:

- The **Edit > Remove Lane** menu command
- The right-click menu
- Keyboard combination CTRL+X
- The **Remove** button  on the authoring pane toolbar

## Resize a lane

1. Select a lane by clicking in a blank part of it. Handles appear at the sides and corners.
2. Drag the side or corner handles.


## Rename a lane

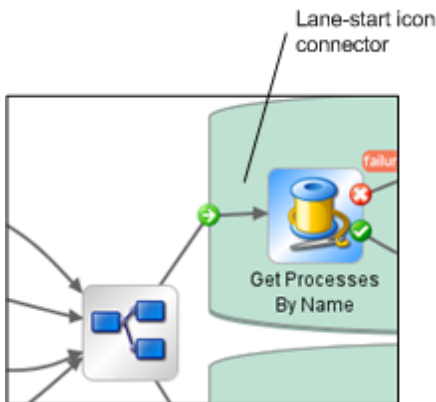
By default, lanes are called **Lane 1**, **Lane 2**, and so on.

1. Right-click the lane and select **Rename**.
2. In the text box that appears, type the new name of the lane.

## Change the start step of a lane

Note that the start step of a lane does not have the green outline that the start step of a flow does.

Drag the **Lane-start** icon  connector from the lane step that is its current target to the step that you want to be the lane's start step.



## Move data into and out of a parallel split step

When a parallel split step starts, each of its lanes obtains copies of the global context flow variables, local context variables, and the inputs of the parallel split step itself. Each lane can use these variables and can create, modify, or delete these variables according to normal flow rules, independently of any other lane.

A step within one lane cannot pass values to a step in another lane. The steps in each lane only have the values that were available when the parallel split step (all its lanes) started.

As the lanes finish execution, the flow variables in each of the contexts are merged back into the context of the calling flow (the flow that the parallel split step is part of). The order of merging is the order in which the lanes terminate. As a result, if two lanes write to the same flow variable, the last one to finish provides the final value of the variable.

Steps within a parallel split step's lanes can obtain data from the local and global contexts and save data to local context. Lane steps can only write to the global context by means of a scriptlet that uses the `scriptletcontext.putGlobal()` method. For the syntax for using `scriptletcontext.putGlobal()`, on the **Scriptlet** tab of an operation or step, insert the **JavaScript** template.

## Debug a parallel split step

In an actual run, the lanes will start and run simultaneously when the flow is run, but when you test them in the debugger, they are executed as a series. You cannot control the order in which lanes are run in the debugger, but by giving them unique names, you can see the order in which they ran.

This is one way in which the debugger does not precisely reproduce the behavior of a flow in a production environment. On the other hand, serial execution in the debugger of parallel split steps

enables you to perform controlled tests for various conditions. For more information, see ["Debugging Complex Flows" on page 222](#).

## Creating a Flow with Multi-Instance Steps

A multi-instance step is a step that executes simultaneously on multiple targets. For example, if you want to run the Windows Diagnostic flow on 100 servers, you can create a multi-instance step that runs the flow on all 100 servers at the same time.

The targets of the operation (in our example, the 100 servers) are defined in an input list in the multi-instance step.



Inside a multi-instance step, you can include one or more operations or subflows. The operations and/or subflows in the multi-instance step run once for each target—these runs are known as *instances*.

Each instance gets, at its beginning, a duplication of the global and local contexts. As it runs, each step in the instance can change the global variables, flow variables, and flow output fields within the multi-instance step.

**Note:** When an exception is thrown in one of the instances, that instance is stopped. The others continue to run, because they are running in parallel.

### ***Differences Between a Multi-Instance Step and a Parallel Split Step***

In a multi-instance step, each instance performs the same task on a different target, while in a parallel split step, each parallel step can be set to do something different.

In a multi-instance step, the number of instances can change during runtime, while in a parallel split step, the number of parallel steps is constant.

### ***Saving Flow Data***

Flow variables, global variables, and flow output fields that are created in an instance of a multi-instance step are local to the instance in which they are created and populated. These variables and flow output field variables will disappear at the end of the lane, unless you use one of the following ways to make this data available to the rest of the flow:



- Bind the data to results on the multi-instance step
- Create a scriptlet in the multi-instance step to save the data

## Saving Data via Results

In order to make the data from flow variables available after the multi-instance step has ended, you can define step results in the multi-instance step, which take their value from flow variables created in the instances. In the **Results** tab of the Step Inspector for a multi-instance step, select a flow variable created in the instances, by selecting **Result <result>** in the **From** column.

You can also save the data from flow output fields created in a subflow in the instances, in a similar way. In the **Results** tab of the Step Inspector for a multi-instance step, select a flow output field created in the instances, by selecting **Result Field <result>** in the **From** column.

You can set the **Assignment Action** field to do different things with the values that are collected. For example, you could append the results of the different instances, or add them together, or get later instances to overwrite previous ones.

In the example below, there are five variables set up for the results of a multi-instance step. The first three take their value from flow variables and the last two from flow output fields.

Inputs	Results	Display	Description	Advanced	Scriptlet
▲ ▼ <b>Step Results</b>					
					Add Result
Name	From	Assign To	Assignment Action		
var1	Result: Result	Flow Variable	OVERWRITE		
var2	Result: Result	Flow Variable	OVERWRITE		
var3	Result: Result	Flow Variable	APPEND		
var4	Result Field: Result	Flow Variable	OVERWRITE		
var5	Result Field: Result	Flow Variable	APPEND		

Assuming that there are two instances, **Instance1** and **Instance2**, that the main flow has empty contexts, and that **Instance2** ends after **Instance1**, the instances provide the following variables:

- **Instance1:**
  - Flow variables:
    - var1 = x
    - var2 = y
    - var3 = w
  - Flow output fields:
    - var1 = z

- **Instance2:**

- Flow variables:

```
var2 = t
```

```
var3 = v
```

- Flow output fields:

```
var5 = u
```

When the multi-instance step finishes, the values of the variables will be:

var1 = NULL (because in **Instance2**, there is no value for this variable, and the action is to overwrite)

var2 = t (the value in **Instance2** overwrites the value from **Instance1**)

var3 = vv (the value in **Instance2** was appended to the value from **Instance1**)

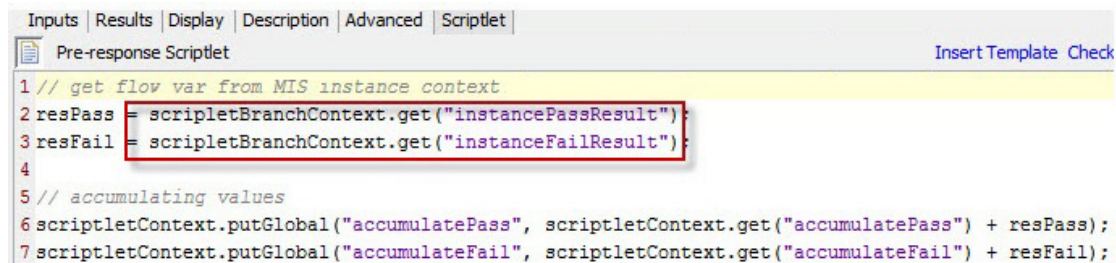
var4 = NULL (because in **Instance2**, there is no value for this variable, and the action is to overwrite)

var5 = u

## ***Saving Data via a Scriptlet***

Another way to make data generated within the step available to the rest of the flow is by creating a scriptlet that collects the data and saves it as a variable that will continue to exist after the instance's run completes.

In the example shown, the scriptlet tracks whether each instance run passes or fails, accumulates this data, and saves it as a variable that is available in the global context.



```
Inputs | Results | Display | Description | Advanced | Scriptlet |
Pre-response Scriptlet Insert Template Check
1 // get flow var from MIS instance context
2 resPass = scriptletBranchContext.get("instancePassResult");
3 resFail = scriptletBranchContext.get("instanceFailResult");
4
5 // accumulating values
6 scriptletContext.putGlobal("accumulatePass", scriptletContext.get("accumulatePass") + resPass);
7 scriptletContext.putGlobal("accumulateFail", scriptletContext.get("accumulateFail") + resFail);
```

This scriptlet will be executed multiple times, once for each instance. Each time, it will be able to access the ScriptletContext of the current instance (called scriptletBranchContext), and can modify the parent flow context (by accessing the scriptletContext).

scriptletBranchContext has same method access as the scriptletContext.

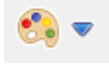

For more information about scriptlets, see ["Using Scriptlets in a Flow" on page 200](#).

## Merging after Upgrade


After upgrading from a previous version of HP OO, if a flow contains multi-instance steps that were created using the **Toggle Multi-instance** option, the global variables created in the step are updated, with later instances overwriting earlier ones.

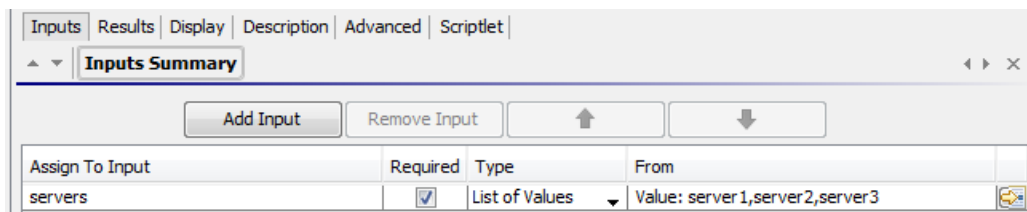
## What do you want to do?

### Create a multi-instance step


1. On the authoring pane toolbar, click the **Step Palette** button  to display the **Step** palette.
2. From the **Step** palette, drag the **Multi-instance**  icon to the authoring canvas.
3. From the **Projects** pane, drag the flow or operation into the multi-instance lane.

**Note:** You can add multiple flows and operations to the multi-instance lane.

4. Set up the list of targets for the multi-instance step, by creating an input that is a list of multiple values. For example, the list of servers that the flow will run against:
  - a. Open the Step Inspector for the multi-instance step by double-clicking the **Multi-step**  icon at the start of the step.
  - b. Create an input. In our example, this could be named **servers**.
  - c. Select the **Required** checkbox and set the type to **List of Values**.





Assign To Input	Required	Type	From
servers	<input checked="" type="checkbox"/>	List of Values	Value: server 1,server 2,server 3

- d. Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.
- e. In the **Input Delimiter** box, type a delimiter (a character that separates the elements in the list).
- f. Specify the way that the list of values will be input. For example, if you want the multi-instance step to run against a number of servers, you can select **Use Constant**, and

specify the server names in the **Constant Value** box. Other ways to populate the list of values are to use the results of a previous step or via integration with another program.

For more information about the options for creating a list of values for input, see ["Specifying the Input Source" on page 127](#).

5. Connect the different parts of the multi-instance step:


- Connect the **Lane-start** icon  to the first step in the multi-instance lane.
- If there are multiple steps in the multi-instance step, connect the steps.
- Drag all the response lines from the last step in the lane to the **Lane-end** icon .

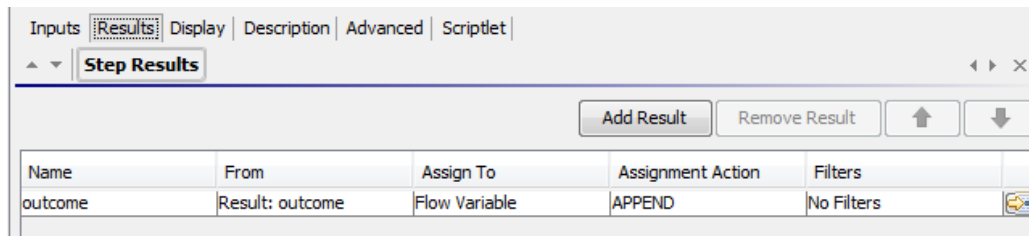


6. Apply the list of targets to each internal step:

- For each of the internal steps inside the multi-instance lane, open the Step Inspector and add an input.
- Open the Input Editor, and in the **Assign from Variable** list, select the variable that you created to hold the list of targets. In our example, this is **servers**.

7. If you want to save the data collected by the different instances of the multi-instance step, create a flow variable to store the result:

- a. Open the Step Inspector for the multi-instance step by double-clicking the **Multi-step**  icon.
- b. Click the **Results** tab and add a result.
- c. In the **Assign To** column, assign the result to a flow variable.
- d. Give a name to the flow variable that will hold the data, for example, **outcome**.
- e. Decide on how you want the data to be stored. In our example, we want to store the results for each server, so the assignment action is **APPEND**. For more details, see *Save output from a multi-instance step*, below.

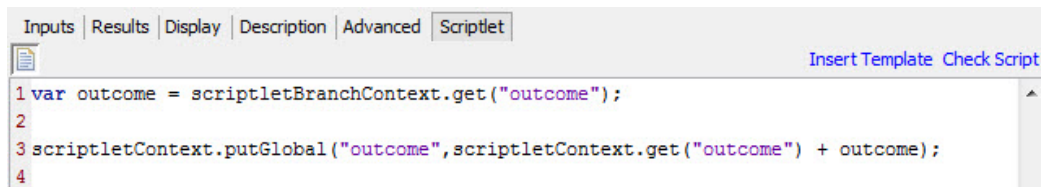


Name	From	Assign To	Assignment Action	Filters
outcome	Result: outcome	Flow Variable	APPEND	No Filters

8. If you want to save the data collected by the different instances of the multi-instance step, to be used in the global context, write a scriptlet to store the result:

- a. In the Step Inspector for the multi-instance step, click the **Scriptlet** tab.
- b. Write a scriptlet that will collect the data from the `scriptletBranchContext`, and will make it available to the `scriptletContext`.

In the example below, the scriptlet tells the flow to accumulate the all values of the **outcome** variable. This is similar to the **APPEND** action that was selected in the previous step.



```

1 var outcome = scriptletBranchContext.get("outcome");
2
3 scriptletContext.putGlobal("outcome",scriptletContext.get("outcome") + outcome);
4

```

9. Connect the multi-instance step to the rest of the flow:

- a. If the multi-instance step is not the start step, connect the step that precedes it to the

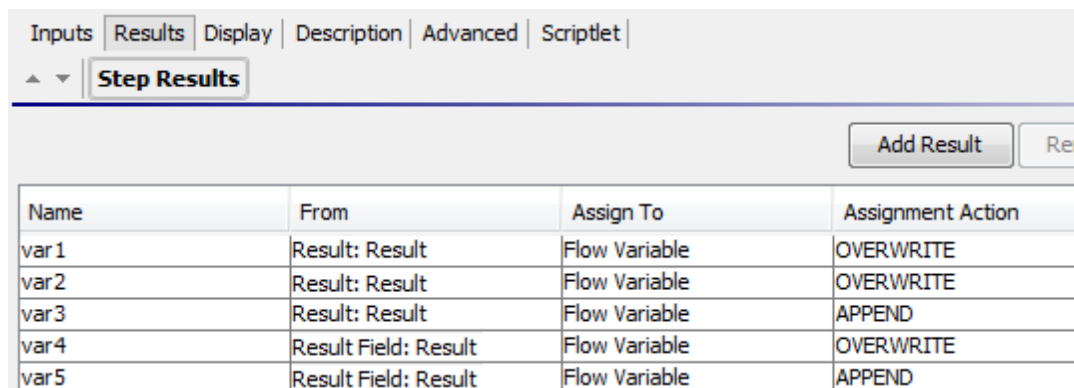
**Multi-instance**  icon.

- b. Connect the multi-instance step's **done**  response to the next step in the flow.

## Save output from a multi-instance step

The data in flow variables and flow output fields in instances are gone after the multi-instance step is completed. To save this data, you can bind it to results in the multi-instance step.

1. Create a multi-instance step, as described above.
2. In the Step Inspector, click the **Results** tab.
3. Add a result line for each flow variable that you want to save.



The screenshot shows the 'Results' tab in the Step Inspector. At the top, there are tabs for 'Inputs', 'Results', 'Display', 'Description', 'Advanced', and 'Scriptlet'. Below these is a 'Step Results' section with an 'Add Result' button. A table is displayed with the following data:


Name	From	Assign To	Assignment Action
var1	Result: Result	Flow Variable	OVERWRITE
var2	Result: Result	Flow Variable	OVERWRITE
var3	Result: Result	Flow Variable	APPEND
var4	Result Field: Result	Flow Variable	OVERWRITE
var5	Result Field: Result	Flow Variable	APPEND

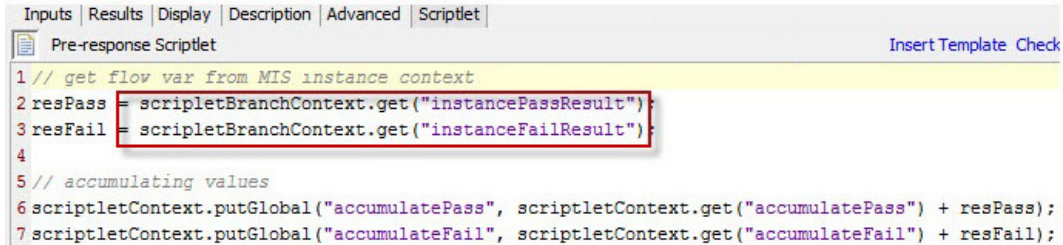
4. In the **Name** column, enter a name for the flow variable that you will be saving the data to.
5. In the **From** column, select the flow variable or output field that is the source of the data that you want to save.
  - To select a flow variable created in the instances, select **Result <result>** in the **From** column.
  - To select a flow output field created in the instances, select **Result Field <result>** in the **From** column.
6. In the **Assignment Action** column, select the action that describes how you want to collect the data.
 

For example, if you wanted to calculate how long it took to run all the instances, you would select **Add**. If you wanted to collect a list of all the servers that were checked in the multi-instance step, you would select **Append**.
7. Save the step. The flow variables that you created will be available for the rest of the flow, after the multi-instance step has finished running.

## Save output from a multi-instance step as a global variable

To save the output from a multi-instance step, so that it can be used outside the flow, you can create a scriptlet to save this output as a global variable.

1. Open the Step Inspector for the multi-instance step by double-clicking the **Multi-step**  icon at the start of the step.
2. Click the **Scriptlet** tab.
3. Write a scriptlet that will collect the data from the `scriptletBranchContext`, and will make it available to the `scriptletContext`. For example:




```

1 // get flow var from MIS instance context
2 resPass = scriptletBranchContext.get("instancePassResult");
3 resFail = scriptletBranchContext.get("instanceFailResult");
4
5 // accumulating values
6 scriptletContext.putGlobal("accumulatePass", scriptletContext.get("accumulatePass") + resPass);
7 scriptletContext.putGlobal("accumulateFail", scriptletContext.get("accumulateFail") + resFail);

```


## Move a multi-instance step

1. Select the **Multi-instance**  icon at the start of the lane, which represents the entire step.
2. Drag the step across the authoring canvas.

## Resize a multi-instance step

1. Select the lane by clicking in a blank part of it. Handles appear at the sides and corners.
2. Drag the side or corner handles to resize the lane.

## Rename a multi-instance step

1. Select the **Multi-instance**  icon at the start of the lane.
2. Right-click and select **Rename**.
3. Type a new name in the text box.

## Debug a multi-instance step

In an actual run, the multiple instances run concurrently, but when you test them in the debugger, they are executed serially. While this means that you are not testing under actual conditions, it does allow you to examine how long it takes each instance to finish.

For more information, see ["Debugging Complex Flows" on page 222](#).

## Using Scriptlets in a Flow

Scriptlets (written in Rhino JavaScript) are optional parts of an operation that you can use to manipulate data from either the operation's inputs or results, for use in other parts of the operation or flow.

You can use scriptlets to test, format, manipulate, or isolate a particular piece of the results.

You can use scriptlets to:

- Filter the results of a operation, flow, or step
- Determine the response of an operation
- Manipulate data in a subflow before passing the data to the parent flow

### *Resources to help you write scriptlets*

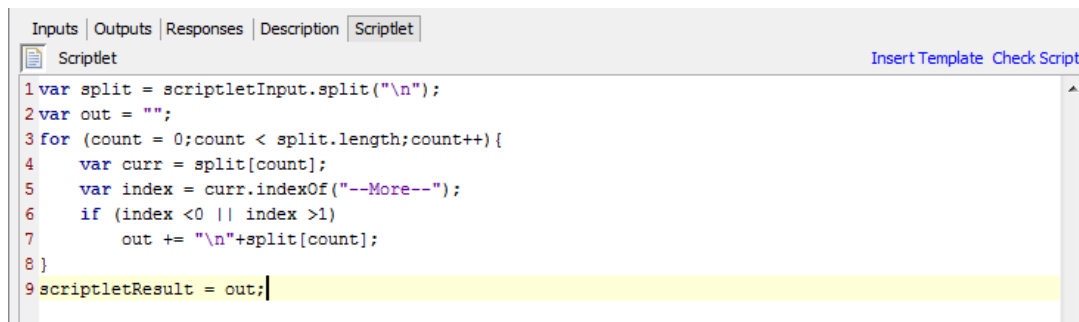
- Scriptlet templates (in Rhino JavaScript) are available in the Scriptlet Editor.
- Default scriptlets are available in the **Configuration\Scriptlets** folder.
- Copy existing scriptlets in default content.

## What do you want to do?

### Create a scriptlet from a template

1. Open the **Properties** sheet or Step Inspector:
  - To add a scriptlet to an operation, right-click the operation in the **Project** pane and select **Properties**.
  - To add a scriptlet to a flow, right-click the flow in the **Project** pane and select **Properties**.
  - To add a scriptlet to a step, double-click the step in the authoring pane.
2. Select the **Scriptlet** tab.





```

1 var split = scriptletInput.split("\n");
2 var out = "";
3 for (count = 0; count < split.length; count++) {
4     var curr = split[count];
5     var index = curr.indexOf("--More--");
6     if (index < 0 || index > 1)
7         out += "\n"+split[count];
8 }
9 scriptletResult = out;

```


3. Click **Insert Template**.
4. Follow the guidelines in the template to write your script.
5. Click **Check Script** to check for errors.
6. Save.

## Use an existing scriptlet

1. Open the **Properties** sheet or Step Inspector:
  - To add a scriptlet to an operation, right-click the operation in the **Project** pane and select **Properties**.
  - To add a scriptlet to a flow, right-click the flow in the **Project** pane and select **Properties**.
  - To add a scriptlet to a step, double-click the step in the authoring pane.
2. Select the **Scriptlet** tab.
3. Open an existing scriptlet in a separate window:
  - Double-click a scriptlet from the **Configuration\Scriptlets** folder.
  - Open an operation that contains scriptlets (for example, the operations in the **Operations\Operating Systems\Linux\Red Hat** folder).
4. Copy the scriptlet text and paste it into the **Scriptlet** text box for your operation, flow, or step.
5. Modify the scriptlet if required.
6. Click **Check Script** to check for errors.
7. Save.

## Filter step or flow results with a scriptlet

You can filter step or flow results using a scriptlet.

1. Double-click a step in the authoring pane.
2. Select the **Results** tab and select the result that you want to filter.
3. Click the right-pointing arrow  at the end of the result row to open the Filter Editor.
4. In the Filter Editor, click the **Add** button.
5. From the **Select Filter** list, select **Scriptlet**.
6. Create a scriptlet to filter the data, in one of the following ways:
  - Click **Insert Template** to use the scriptlet template as a basis.
  - Copy and paste the text from an existing scriptlet in another operation or from the **Configuration\Scriptlets** folder.



For more information about creating filters, see ["Filtering Output and Results" on page 158](#).

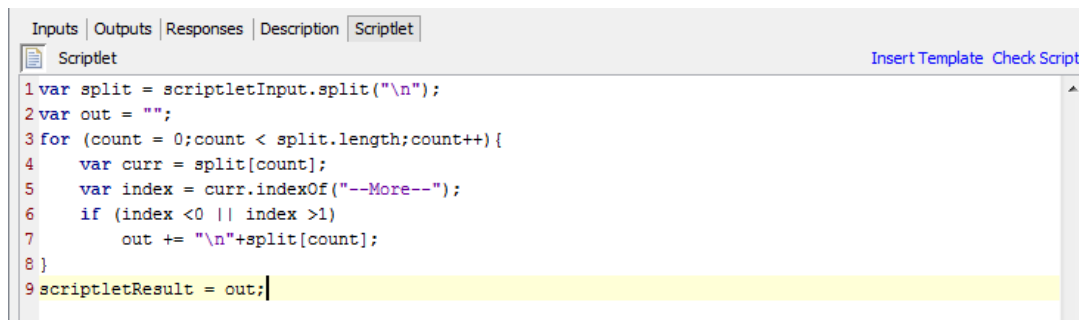
7. Click **Check Script** to check for errors.
8. Test the filter and save your work.

**Tip:** When creating a scriptlet operation, in the scriptlet, specify the scriptlet response as success. Then, on the **Responses** tab of the operation, select failure as the default response.

## Create a scriptlet rule for an operation response

You can use a scriptlet to control the response in an operation.

1. Open the **Responses** tab of the operation and select a response.
2. Click the right-pointing arrow  at the right end of the response's row, to open the Rule Editor.
3. From the **Rule Type** list, select **Scriptlet**.
4. Click the right-pointing arrow  at the right end of the rule's row, to open the Rule Details Editor.




The screenshot shows the 'Scriptlet' tab in the Rule Editor. The scriptlet code is as follows:

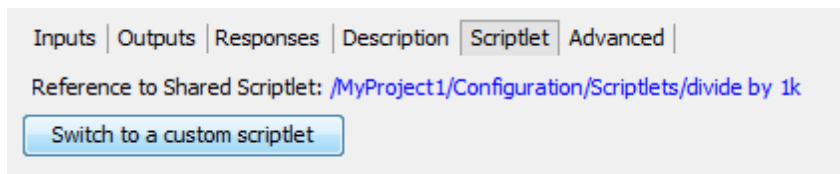
```
1 var split = scriptletInput.split("\n");
2 var out = "";
3 for (count = 0; count < split.length; count++) {
4     var curr = split[count];
5     var index = curr.indexOf("--More--");
6     if (index < 0 || index > 1)
7         out += "\n"+split[count];
8 }
9 scriptletResult = out;
```

5. Create the scriptlet in one of the following ways:
  - Click **Insert Template** to use the scriptlet template as a basis.
  - Copy and paste the text from an existing scriptlet in another operation or from the **Configuration\Scriptlets** folder.
6. Create the scriptlet and click **Check Script** to check for errors.
7. Save your work.


### Use a system scriptlet in an operation, flow, or step

1. Open the **Scriptlet** tab of the **Properties** sheet or Step Inspector for the operation, flow, or step on which you want to use the system scriptlet .
2. In the **Projects** pane, expand the **Configuration** and **Scriptlets** folders.
3. Drag the scriptlet from the **Scriptlets** folder to the **Scriptlet**  icon in the **Scriptlet** tab of the **Properties** sheet or Step Inspector.

The Scriptlet tab shows that there is now a reference to a shared scriptlet.



### Save a scriptlet to the Configuration\Scriptlets folder

1. Under the **Scriptlet** tab in the **Properties** sheet or Step Inspector, open the scriptlet that you want to save.
2. Drag the **Scriptlet**  icon to the **Configuration\Scriptlets** folder in the **Projects** pane.
3. Enter a name for the scriptlet.

## Reference Material

### Scriptlet Editor

The scriptlet editor has the same appearance, whether you get to it via the **Scriptlet** tab in the **Properties** sheet or Step Inspector or by double-clicking a scriptlet from the **Configuration\Scriptlets** folder.

InputsOutputsResponsesDescriptionScriptlet


Scriptlet

[Insert Template](#)
[Check Script](#)

```

1 var split = scriptletInput.split("\n");
2 var out = "";
3 for (count = 0; count < split.length; count++){
4     var curr = split[count];
5     var index = curr.indexOf("--More--");
6     if (index < 0 || index > 1)
7         out += "\n"+split[count];
8 }
9 scriptletResult = out;

```

GUI item	Description
<b>Scriptlet icon</b> 	Drag this icon to the <b>Configuration\Scriptlets</b> folder, to save a scriptlet there for reuse.
<b>Insert Template</b>	Click <b>Insert Template</b> to see guidelines to help you write your scriptlet.
<b>Check Script</b>	Click <b>Check Script</b> to check the scriptlet for errors.

## Using Regular Expressions in a Flow

A regular expression (also known as a regex) allows you to search not only for exact text but also for classes of characters. For example, to match any digit, you can use the wildcard `\d`.

You can use regular expressions to:

- Create result/output filters that extract key pieces of data for:
  - Saving in variables for use in later operations
  - Testing to determine a step's response

## Wildcards and Modifiers for Regular Expressions

The key wildcards for regular expressions are:

Wildcard	Uses
<code>^</code>	Matches the beginning of a string
<code>\$</code>	Matches the end of a string
<code>.</code>	Any character except new line
<code>\b</code>	Word boundary
<code>\B</code>	Any except a word boundary

\d	Any digit 0-9
\D	Any non-digit
\n	New line
\r	Carriage return
\s	Any white space character
\S	Any non-white space character
\t	Tab
\w	Any letter, number, or underscore
\W	Anything except a letter, number, or underscore

The modifiers for regular expressions are:

Modifier	Effect
*	Match zero or more
+	Match one or more
?	Match zero or one
{n}	Match exactly n occurrences
{n,}	Match n or more occurrences
{n,m}	Match between n and m occurrences
[abc]	Match either a, b, or c
[^abc]	Match anything except a, b or c
[a-c]	Match anything between a and c
a b	Match a or b
\	Escape a special character (for example \. Means '.' not match anything)

## What do you want to do?

### Use a regular expression to filter test output

1. Open the Filter Editor for an output or result, and create a new filter. For more information, see ["Filtering Output and Results" on page 158](#).
2. From the **Select Filter** list, select **Regular Expression** as the filter type. The **Details for**:

section in the upper-right shows controls for creating a regular expression.

Details for: Regular Expression  
Filter the input string by extracting the parts that match a regular expression. See the help documentation for a detailed description of regular expressions.

Expression Type: Java Style

Expression Value: .\*

Filter Style: ☒ Filter Entire Input  
☐ Filter Line-by-line

Ignore Case: ☒

- From the **Expression Type** list, select **Java Style**. Do not use the other styles; they have been deprecated.
- In the **Expression Value** box, type a regular expression.

**Example:** To extract the number of packets lost, you can use the regular expression `Lost = \d`.

This expression tells HP OO to search for the string “Lost = ” followed by any number.

The wildcard `\d` tells HP OO to match any digit.

- For **Filter Style**, select **Filter Entire Input** or **Filter Line-by-line**, according to how you want the filter applied to the raw results.
- To make the regular expression not case-sensitive, select **Ignore Case**.
- Click **Test Selected Filters** to test the filter.
- Save the filter.

## Combine multiple regexes to isolate a value

You can combine multiple regular expressions to isolate the value in a filter.

For example, in the output of the Unix `ps` command, extracting the time for `ps` requires two regular expressions: one to filter the output down to the line for `ps` and the second to extract the time.

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
0	S	512	2160 4	2160 3	0	75	0	-	1096	wait	pts/1	00:00:00	Bash
0	R	512	2659	2160 4	0	76	0	-	1110	-	pts/1	00:00:00	Ps

1. Open the Filter Editor for an output or result.
2. Add a new regular expression filter.
3. In the **Expression Value** box, type the first regular expression.

In our example, type **.ps**. This extracts any characters ending with “ps”.

**Note:** Make sure not to omit the leading period [.]

4. Select the **Filter line by Line** check box.
5. Click **Test Selected Filters**.

In the **Test Output** box, the only output is the line containing “ps”.

6. Add a second regular expression filter.
7. In the **Expression Value** box, type **\d\*:\d\*:\d\***

This represents three sets of digits separated by colons. In our example, this will extract the time from the line.

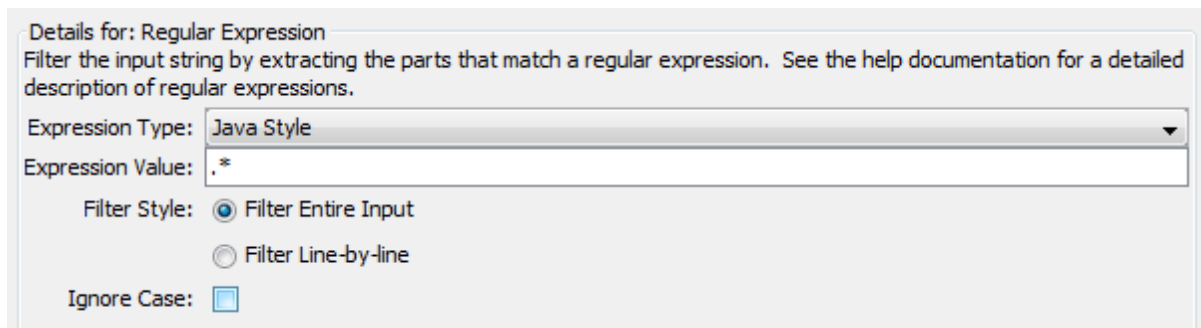
8. Click **Test Selected Filters**.
9. Save.

The test output now shows only the time from the ps line. Now you can assign this value to a variable.

## Reference Material

### Filter Editor > Details for: Regular Expression

When you select **Regular Expression** as the filter type, the **Details for:** section in the upper-right shows controls for creating and modifying a regular expression.



Details for: Regular Expression

Filter the input string by extracting the parts that match a regular expression. See the help documentation for a detailed description of regular expressions.

Expression Type: Java Style

Expression Value: .ps

Filter Style: ☒ Filter Entire Input ☐ Filter Line-by-line

Ignore Case: ☐

GUI item	Description
<b>Expression Type</b>	Select <b>Java Style</b> as the type of regular expression to filter the data with. Do not use the other styles; they have been deprecated.
<b>Expression Value</b>	Type the regular expression.
<b>Filter Style &gt; Filter Entire Input</b>	Select to apply the filter to the entire raw result.
<b>Filter Style &gt; Filter Line-by-line</b>	Select to apply the filter to each line separately.
<b>Ignore Case</b>	Select to make the regular expression not case-sensitive.



# Validating Content

Before releasing your content, it is important to test and validate the flows in your project. Studio provides the following tools to help you do this:

- The **Problems** pane displays a list of any problems, with their locations and descriptions, to guide you in repairing these problems
- The debugger helps you track down the causes of errors and unexpected behaviors in flows

## Validating Flows in the Problems Pane

For a flow to run, the flow itself, its operations, and any system accounts used in the flow must be valid.

Using the **Problems** pane, you can check an individual flow or operation for problems, or you can validate an entire project. This validates all the flows, operations, and system accounts in the project.

Problems				
Source Type	Name	Description	Location	
Step	Get Row Index by Condition	User Input: hasHeader references missing selection list	/My_Project/Content/Library/Get Cell by...	
Step	Is rowsCount Greater Than 0	Operation cannot be found	/My_Project/Content/Library/Get Cell by...	
Step	SQL Query	User Input: host references missing selection list	/My_Project/Content/Library/Health Ch...	
Transition	success	Transition source step has no operation linked to it	/My_Project/Content/Library/Get Cell by...	
Transition	failure	Transition source step has no operation linked to it	/My_Project/Content/Library/Get Cell by...	

### *What a flow needs to be valid*

To be valid, a flow must have the following:

- At least one step
- One of the steps designated as the start step
- For each step, from each response, a transition connecting the step to a subsequent step
- A way for each step in the flow to be reached in one run or another
- A return step to return a value and end the flow
- Assignment of how each input gets its value

## What do you want to do?

### Validate a flow or operation

1. Select a flow or operation in the **Projects** pane.
2. Click the **Problems** tab to display the **Problems** pane.

### Validate all the flows and operations in a project

1. Open the project that you want to validate.
2. From the **Tools** menu, select **Validate Flows and Operations**.

A list of any problems appears, with their locations and descriptions, to guide you in repairing the problems.

## Reference Material

### Problems pane

The **Problems** pane, which you open with the **Problems** tab on the lower edge of the Studio window, lets you check whether a selected flow or operation is valid.

Problems				
Source Type	Name	Description	Location	
Step	Get Row Index by Condition	User Input: hasHeader references missing selection list	/My_Project/Content/Library/Get Cell by...	
Step	Is rowCount Greater Than 0	Operation cannot be found	/My_Project/Content/Library/Get Cell by...	
Step	SQL Query	User Input: host references missing selection list	/My_Project/Content/Library/Health Ch...	
Transition	success	Transition source step has no operation linked to it	/My_Project/Content/Library/Get Cell by...	
Transition	failure	Transition source step has no operation linked to it	/My_Project/Content/Library/Get Cell by...	

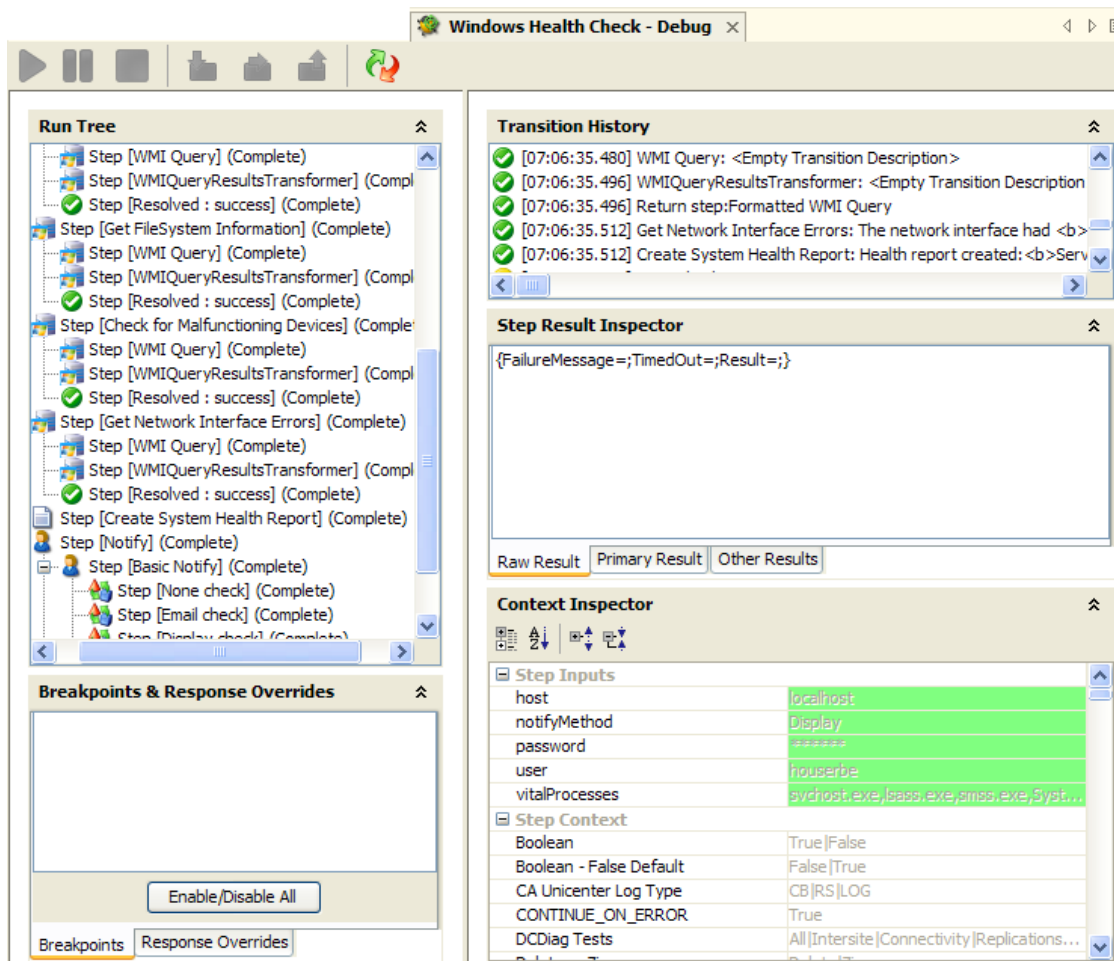
GUI item	Description
Source Type	Displays the type of the element in which there is a problem.
Name	Displays the name of the element in which there is a problem.
Description	Describes the problem, to guide you in how to repair it.
Location	Displays the location of the element with the problem.

## Testing and Debugging a Flow



The debugger helps you track down the causes of errors and unexpected behaviors in flows, by displaying the following information:

- A tree showing the steps executed
- Step results and operation outputs generated for each step
- Flow variable values in the various contexts current to each step
- The transition description for each transition followed



You can also set breakpoints for the debugger and force response choices in order to zero in on the behavior you want to test.


## Best Practices

It is recommended to debug subflows before debugging their parent flows.

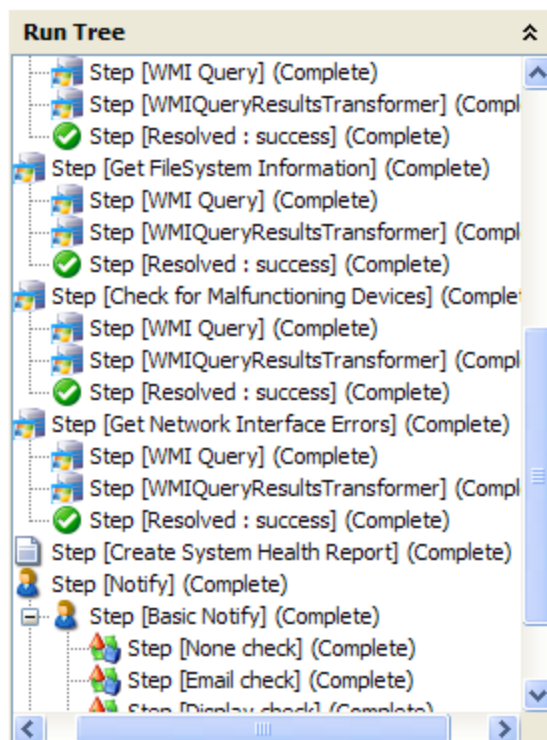
## What do you want to do?

### Debug a flow

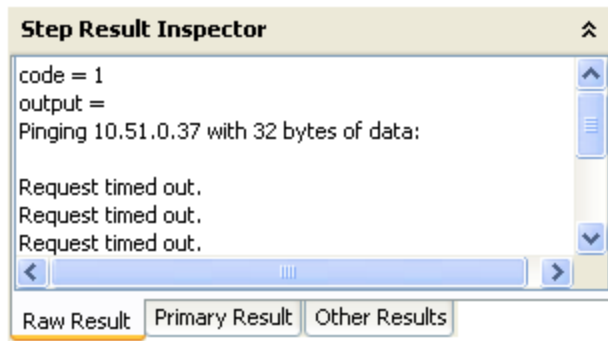
1. Right-click the flow in the **Projects** pane, and then click **Debug**.

**Note:** Alternatively, you can open the flow in the authoring pane and click the **Debug**  button.

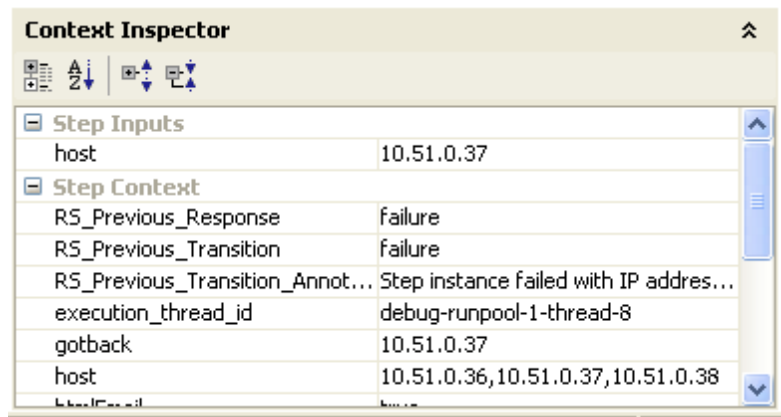
2. To run the flow to its end, click the **Play**  button in the debugger toolbar.
3. To see the information for a completed step, click the step in the **Run Tree** pane.




4. In the **Step Result Inspector** pane, you can view the step's raw results, primary result, or other filtered results.



5. To see global variables, flow variables, and their values for the step's inputs and the step and global context, navigate to the appropriate section of the **Context Inspector** pane.





## Debug a flow - step by step

1. Open the flow in the debugger.
2. To run the flow step by step, click the **Step Over**  button.



## Step into and out of a subflow

It is recommended to debug subflows before debugging their parent flows.

- To step into a step's subflow, click the **Step Into**  button.
- To step out of the subflow, click the **Step Out**  button.



## Collapse/restore panes in the debugger

You may want to collapse some of the panes in the debugger, in order to make more room for another of the panes.

- To collapse a pane, click the upward-facing double chevron  at the top right of the panel.
- To restore a collapsed pane, click the downward-facing double chevron .


## Reset and restart a flow in the debugger


When you reset and restart a flow, the values of its flow variables are reset to the values that they had when you opened the debugger.

1. In the debugger toolbar, click the **Reset**  button.
2. Click the **Play**  button.

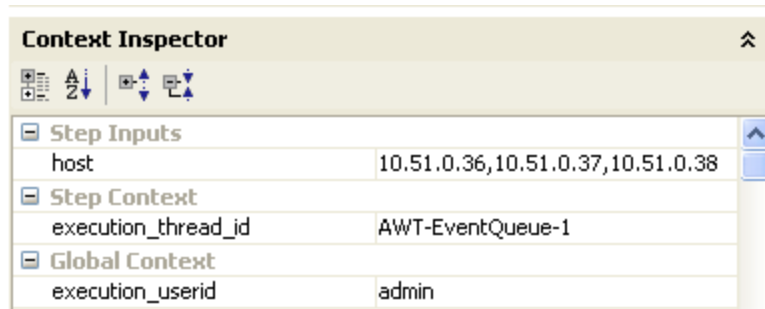
## Change values of flow variables within the debugger

To see how a flow behaves with different values for its flow variables, you can change the value of a flow variable before running a step.

1. Open the flow in the debugger.
2. Click **Step-Over**  until the step in which you're interested is pending.


**Note:** If you have set a breakpoint before the step, you can click **Play**  to run the flow until it pauses at the step.

The **Context Inspector** pane shows the current values of **Step Inputs** and **Step Context** as of the point at which the step is pending.



- The values in the **Step Input** section are the values that were assigned to the input before the step started.
  - The values in the **Step Context** section are the values that were updated after the step began.
3. To change the value of a flow variable used in this step, under **Step Inputs**, find the listing for the flow variable, highlight its value, and type a new value to replace it. In the above example,

the step is a multi-instance step. You could add another IP address to the list in the host flow variable.

4. To change the value of a flow variable that is accessible in this step but is used in a later step, change the value for the flow variable's listing under **Step Context**.
5. Continue to play or step through the flow.
6. To reset any flow variable values that you have changed to the values that were set the last time you saved the flow, click the **Reset**  button.

## Set a breakpoint in a flow

Breakpoints provide automatic pauses in the running of a flow in the debugger. This can come in handy when you want to, for example:

- Examine the value of a flow variable
- Change the value a flow variable to see its effect on the flow in the rest of the run

You set breakpoints in the flow's diagram, but you can enable or disable any breakpoints that you have set from inside the debugger.

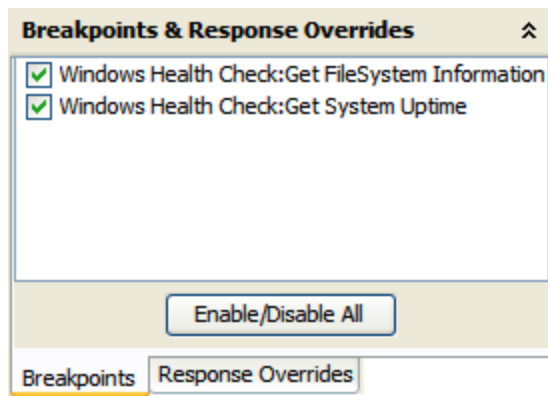
1. Open the flow open in the authoring pane, and right-click the step where you want to set the breakpoint.
2. Select **Debugging > Set Breakpoint**.

In the flow diagram, the breakpoint is indicated by a yellow-and-black border surrounding the step.



3. Open the flow in the debugger.

In the debugger's **Breakpoints & Response Overrides** pane, the **Breakpoints** tab shows the existing breakpoints.



4. Do one of the following:

- To enable a single breakpoint, select the breakpoint's check box.
- To disable a single breakpoint, clear the breakpoint's check box.
- To enable or disable all the breakpoints, click **Enable/Disable All**.
- To clear all the breakpoints, from the **Tools** menu, select **Remove All Breakpoints**.

### Override a response in a debug run for a single step

Response overrides force the response that you selected, even if the operation fails.

By overriding a response, you can test a particular path of the flow without having to exit the debugger and change input values.

For example, if you have a step in a flow for which you don't have the necessary information, you might want to test the rest of the flow, regardless of the certain failure of that step. You can force the run to follow the response and transition that you want, rather than the failure response that would come about without your intervention.

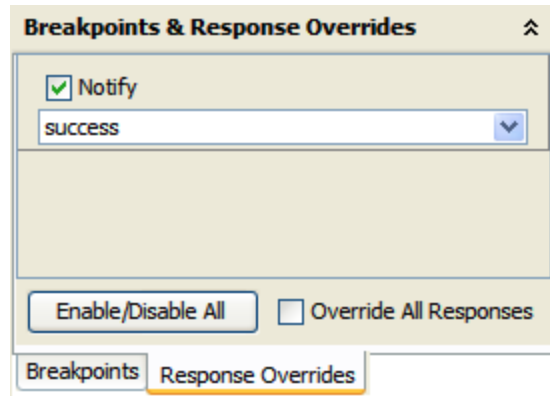
1. Open the flow open in the authoring pane, and right-click the step whose response you want to override.
2. Select **Debugging > Override Response**, and then click the response you want to force the step to have:
  - **None**
  - **Success**
  - **Failure**
  - **Prompt**

After you have created a response override, you can enable or disable the override in the debugger, or choose a different response for it.



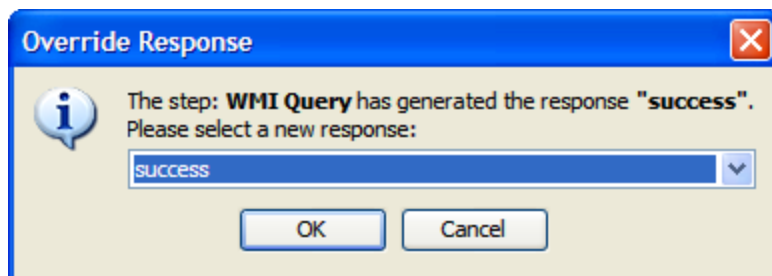
3. Open the flow in the debugger.

In the debugger **Breakpoints & Response Overrides** pane, the **Response Overrides** tab shows the existing response overrides.



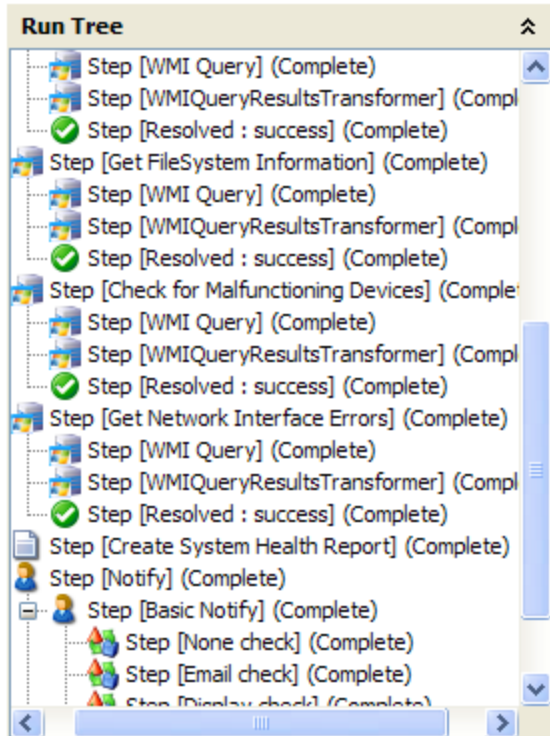
4. Scroll up or down to the response override of interest.
5. Do one of the following:
  - To choose a different response for an override, click the down arrow and select the response.
  - To enable a single response override, select its check box.
  - To disable a single response override, clear its check box.
  - To enable or disable all the response overrides, click **Enable/Disable All**.
  - To clear all the response overrides, from the **Tools** menu, select **Remove All Response Overrides**.
  - To override the response on every step, select the **Override All Responses** checkbox.

When you run the flow in the debugger after overriding all responses, you are prompted at each step to manually select a response for the step.



## Reference Material

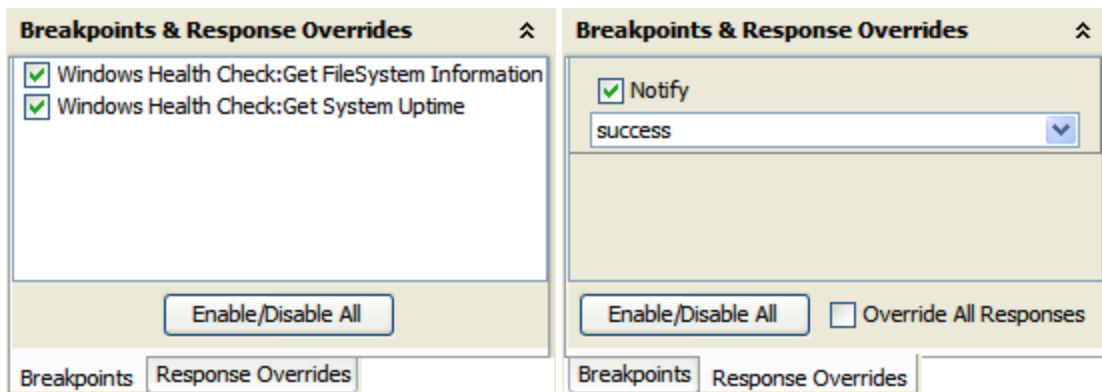
### Run Tree pane



The **Run Tree** pane shows each step that runs, including steps in subflows of the flow.

Steps that will run simultaneously in an actual execution are run in a serial sequence in the debugger.

### Breakpoints & Response Overrides pane

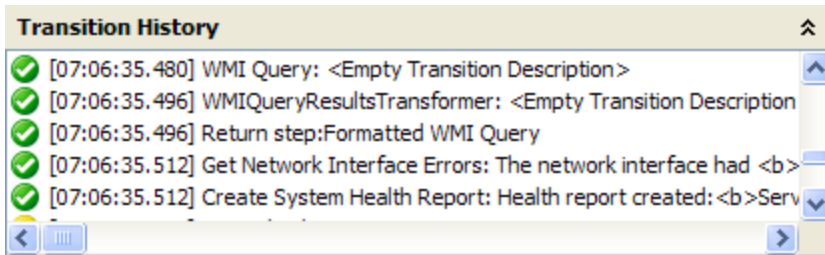


- Breakpoints are flags that enable you to automatically pause a run at a certain step in order to examine the results, the path of the run, or the values in the flow variables at a that point.

- Response overrides force the response that you select, regardless of the result of the operation.

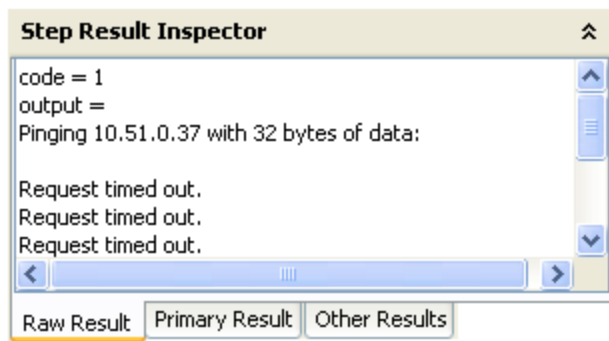
The **Breakpoints & Response Overrides** pane displays breakpoints and response overrides, and enables you to remove them or enable or disable them for this run.

## Transition History pane



The **Transition History** pane lists the transitions that have been followed in the run and displays their descriptions.

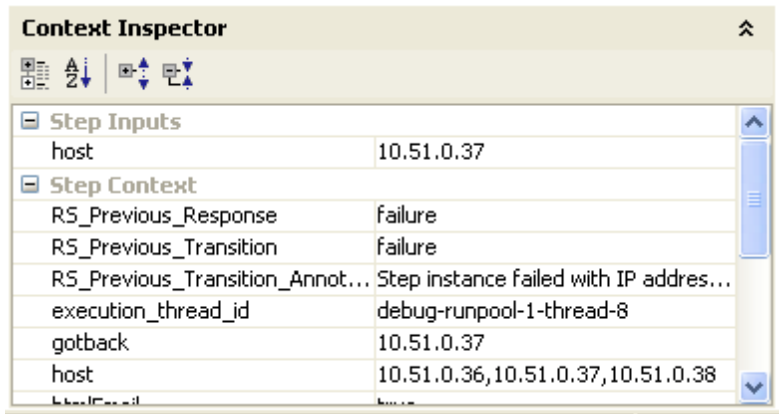
## Step Result Inspector pane



The **Step Result Inspector** pane displays the results of the selected step.

- Click the **Raw Result** tab to see the raw results (the results of the step's operation).
- Click the **Primary Result** tab to see the primary result of the step.
- Click the **Other Results** tab to see other results that you may have created.

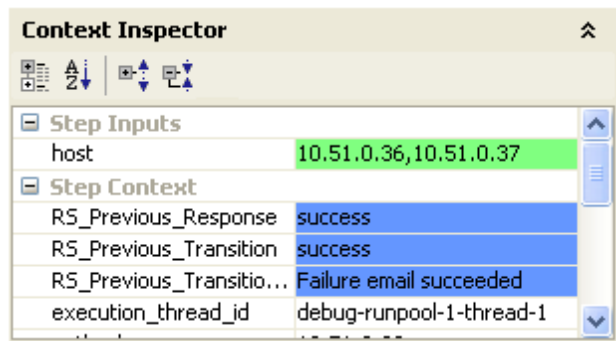
## Context Inspector pane



The **Context Inspector** pane displays the current values of flow variables (global as well as local) for each step.

Navigate to the appropriate section of the **Context Inspector** pane, to see global variables, flow variables, and their values for the step's inputs and the step and global context.

- The values in the **Step Input** section are the values that were assigned to the input before the step started. The text boxes that contain the values for flow variables are color coded.

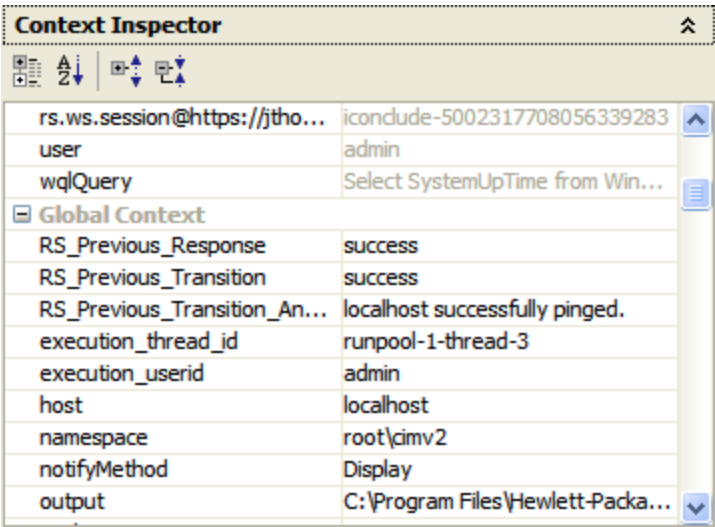


- The values in the **Step Context** section are the values that were updated after the step began.

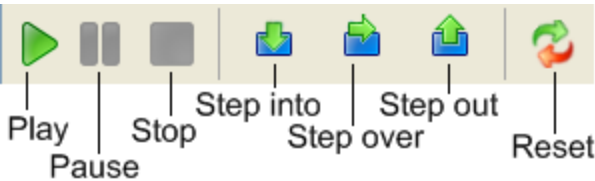
A step's context is the collection of flow variables and their value assignments in the local contexts of the step's flow and any parent flows. (If a flow is a step in another flow, the relationship between the two flows is subflow to parent flow.)









- The values in the **Global Context** section are system properties and any global variables that

have been created.



Debugger toolbar



GUI item	Description	Keyboard shortcut
<b>Play</b> 	Run the flow to its end.	F11
<b>Pause</b> 	Pause a flow that is running in the debugger. You can click the <b>Play</b>  button to start it running again from the point at which it was paused.	ALT + P
<b>Stop</b> 	Stop a flow that is running in the debugger.	ALT + C
<b>Step Over</b> 	Run the flow step by step	F5
<b>Step Into</b> 	Step into a step's subflow.	F6
<b>Step Out</b> 	Step out of a step's subflow.	F7
<b>Reset</b> 	Reset the flow variable values to the values that they had when you opened the debugger.	F12

# Debugging Complex Flows

## *Debugging Flows with Parallel Processing Steps*

Studio debugs steps, multi-instance, or parallel split step, with parallel processing. To learn how a flow with steps that use parallel processing will behave in execution, there is no substitute for running the flow in a staging environment after testing the flow in the Studio debugger.

You debug a flow containing a parallel split or multi-instance step the same way you debug a flow without such steps, but you should take into account that they run differently in the debugger.

## What do you want to do?

### **Debug a parallel split step in a flow**

In a flow run, the debugger starts the flows at the start time, and the order in which they finish depends on variable factors that cannot be predicted in Studio. Thus the debugger cannot predict considerations such as in the case of conflicting writes to the same flow variable, which lane writes to the flow variable last.

On the other hand, in Studio, you can manipulate the order in which the lanes will finish in the debugger in order to test various scenarios in a controlled fashion.

For more information about parallel split steps, see ["Creating a Flow with Parallel Split Steps" on page 188](#).

### **Debug a multi-instance step in a flow**

In a flow run, the multiple instances run concurrently, and the flow continues with the steps that follow one instance's response while the other instances are processed.

While this means that you are not testing under actual conditions, it does allow you to examine how long it takes each instance to finish.

For more information about multi-instance steps, see ["Creating a Flow with Multi-Instance Steps" on page 192](#).

# Debugging Central with Studio

Central remote debugging capability enables Studio authors to run flows on Central or Central configured with as an external RAS and Database.

Studio exposes a set of properties which the user must configure to be able to connect and debug on a remote or local Central.

Studio supports two connection protocols HTTP and HTTPS. Studio also supports authentication. When authentication is enabled you must make sure that the user has entitlement on all running flows.

## Studio Configuration File

Studio expose the Central connection configuration with a studio configuration file, located in [STUDIO\_HOME]\conf\studio.properties.

The studio.properties file contains the following properties:

- engine.connection.protocol: HTTP and HTTPS protocols.
- engine.connection.host: Central host name or IP.
- engine.connection.port: Central port.
- engine.connection.authenticated: False or true in case the central is using authentication.

## What do you want to do?

**Note:** To debug any flow on Central from Studio, you must deploy all the content that flow uses, for example, the base content pack into Central.

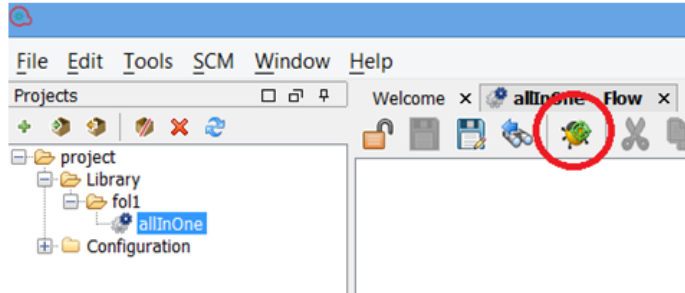
### Debugging in Central using an authenticated HTTP connection

**Note:** The executed flow must be deployed with entitlement for the selected user.

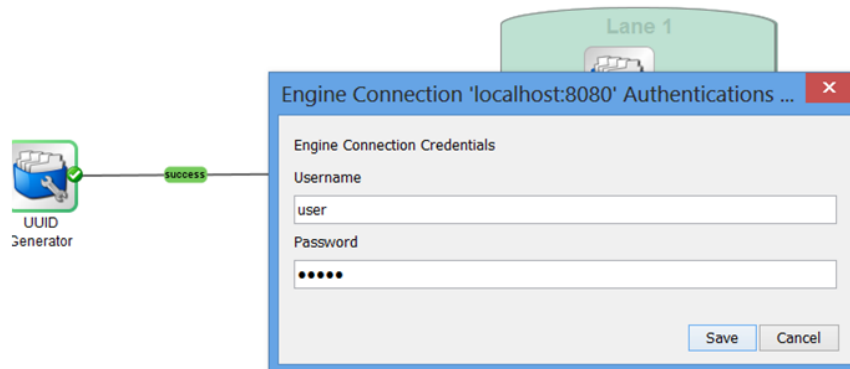
1. Open STUDIO\_HOME/conf/studio.properties file.
2. Change the connection properties as follows:
  - engine.connection.protocol: http
  - engine.connection.host: localhost
  - engine.connection.port: 8080
  - engine.connection.authenticated: true
3. Open Central.
4. Create an internal user, or configure LDAP for a secure user. See [Setting Up Security – LDAP Authentication](#) in the Central user guide for details on configuring LDAP in Central.
5. Enable security settings in Central (select **Enable authentication** in Central, See [Setting Up Security – LDAP Authentication](#) in the Central user guide for details on configuring LDAP in Central).

The next step is to build a flow and debug it in Central:

1. Build a flow using operations and flows from imported content pack or use an existing one.
2. Run and debug flow.



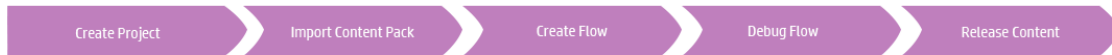
3. A pop-up window is displayed, prompting the user to enter the user name and password for Central authentication.



4. Enter the user name and password of the authenticated user of Central, and then click **Save**.
5. Run the flow. The flow runs and completes successfully.



## Exporting a Content Pack



After you have finished validating your flow, you're ready to release it into a content pack, so that it can be deployed and run.

A content pack is the outcome of a project. It contains entities in the project and reference IDs. A content pack includes not only flows and operations but also actions and configuration items.

The content pack is the element that you release for deployment.

**Note:** Invalid flows or operations will not be included in the content pack.

When you create a content pack from a project, this content pack is given the same unique identification number (UUID) as the project. This means that you can create subsequent versions of the content pack from the project, and once deployed, a new version will overwrite the previous version with the same UUID. If you want to create a new content pack that does not overwrite the previous one, you will need to create a copy of the project, which will have a new UUID, and create the content pack from that copy.

## Recommended Best Practices

- When you create a new version of a content pack, make sure that you give it an updated version number, in order to distinguish it from previous versions of the content pack.
- When you import a new version of a content pack into Studio, it is recommended to close or delete the previous version from Studio, to avoid duplication issues.
- If you are creating multiple content packs, make sure that they don't contain configuration items with identical names. It is recommended however that you prevent duplicate configuration item names. It is recommended to have a separate shared project for shared configuration items.
- If you have moved content from one content pack to another, deploy Contact Pack A that contains flow1 and flow1, Central now contains flow1,flow2 that are assigned to CP-A). Then, deploy Contact Pack B that contains flow1, Central now contains flow2 assigned to Contact Pack A and flow1 assigned to Contact Pack B.

## What do you want to do?

### Create a content pack

1. In the **Projects** pane, select the project from which you want to create a content pack.

2. Select **File > Create New Content Pack**.

**Note:** Alternatively, you can select the **Create Content Pack**  button in the **Projects** pane.

3. In the Create Content Pack dialog box, enter the content pack details:
  - **Location** – enter or browse to the location where you want to save the content pack. By default, the path to the project workspace is selected.
  - **Version** – if you are producing multiple iterations of the same content pack, it is recommended to give them each a version number.
  - **Publisher**
  - **Description** (optional)

The **Name** field displays the name of the project. This is read-only.

4. Click **OK**. A new content pack is created in the location that was specified.

This content pack can be deployed and run, or imported into another project.

## Reference Material

### Create Content Pack dialog box

GUI item	Description
<b>Name</b>	The name of the content pack is taken from the project name. This field is read-only.
<b>Location</b>	Enter or browse to the location where you want to store the content pack. By default, the path to the project workspace is selected.
<b>Version</b>	Enter the version of the content pack. This information will appear in the content pack's Properties sheet.
<b>Publisher</b>	Enter the publisher of the content pack. This information will appear in the content pack's Properties sheet.
<b>Description</b>	Enter a description of the content pack. This information will appear in the content pack's Properties sheet.

# Managing Flows and Operations

Your project library may contain a large number of flows and operations. This chapter discusses how to manage this library—how to locate, copy, and bookmark items, how to see how they are used, and how to create new operations.

Creating Operations .....	227
Finding a Flow or Operation .....	235
Copying Flows and Operations .....	240
Changing Between Hard Copy and Soft Copy .....	242
Replacing a Plugin in a Hard Copy .....	242
Finding Out How Flows and Operations are Used .....	243
Generating Documentation about Flow and Operations .....	245
Managing Version History of Flows and Operations .....	250
Bookmarking Flows and Operations .....	252

## Creating Operations

There are three ways to create operations in Studio:

- Copying and modifying existing operations.
- Importing an operation from an already existing plug in.
- Creating an action plug in in Java and importing that action plug in to Studio

## Creating Operations from Action plug ins

An action plug in is a jar file containing `IActions` or `@Actions`. You can import an action plug in to Studio, in order to create an operation from one of the actions in it.

An action plug in may include multiple actions, and you can create an operation from each one of these actions.

For information about developing action plug ins, see the *Extension Developers Guide*.

## ***Copying Operations that were Created from Action plug ins***

When you copy an operation that was created by importing an action plug in, the copied operation continues to reference the original operation. If the action plug in is upgraded, when you update the

original operation to call the new version, the copied operations are all updated automatically. This is known as a "soft copy".

The source operation, which this operation was copied from, is displayed in the **Advanced** tab.

Note that the link to the action plug in is the only item that is automatically updated in the copied operations. Changes that you make to the original operation's input, output, variables, scriptlets, and so on, are not updated in the copies.

## Valid Operations

A valid operation requires:

- There are operations that require no inputs such as UUID Generator.
- At least one response that is mapped to valid expressions describing outcomes of the operation.

If the new operation is invalid or incomplete, its name is displayed in the **Projects** pane in red zig-zag underscore. Moving the cursor over the name displays a tool tip that specifies how the operation is incomplete.

## Best Practice

To help authors who will create flows using the operations you create, add the following information to the operation's **Description** tab:

- A description of what the operation does.
- Inputs that the operation requires, including where authors can find the data that the inputs require and the required format for the data.
- Responses, including the meaning of each response.
- Result fields, including a description of the data supplied in each result field.
- Any additional implementation notes, such as:
  - Supported platforms or applications, including version information
  - Application or Web service APIs that the flow interacts with
  - Other environmental or usage requirements

## What do you want to do?

### Copy and modify an operation in Studio

1. In the **Content Packs** pane or **Projects** pane, select the operation that you want to copy.

2. Select **Edit > Copy**.
3. Select the location in the project tree where you want to paste the copy, and select **Edit > Paste**. The operation is treated as a new object.
4. To assign the operation to a category for search purposes, click **Assign Categories** and then select a category from the list.
5. To create an input, click the **Inputs** tab and then click **Add Input**.
6. In the dialog that appears, type the input name and then click **OK**.
  - For information on what inputs are and how to use them, see ["Creating Input" on page 120](#).
  - For information on defining an input data source, see ["Specifying the Input Source" on page 127](#).
7. Add and define any output data.

For information on adding and working with output data, see ["Setting Operation Outputs" on page 150](#).
8. Create any responses needed and map results to the responses.

For information on defining rules that govern which responses are chosen for the operation, see ["Setting Responses" on page 139](#).
9. Click the **Description** tab and write the description in the text box.
10. Click **OK**.

### Change the source operation that the operation is based on

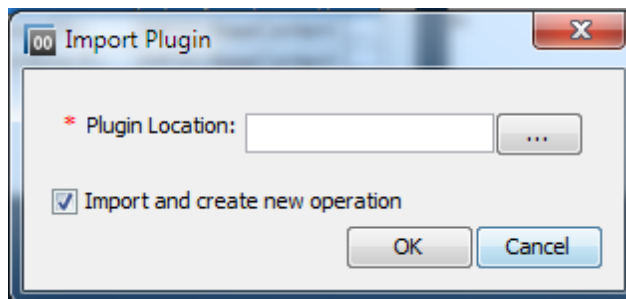
If you created an operation by copying an existing operation, you can change the source operation. Your copied operation will become a copy of the operation that you select as the source.

1. Open a copied operation, and select **Properties**.
2. In the Properties sheet, click the **Advanced** tab.
3. Under **Source Operation**, click the **Select** button.
4. In the Select Source Operation dialog box, navigate to and select the source operation that you want to base the copy on, and then click **OK**.
5. If required, rename the operation to reflect the change.
6. Review and make any changes necessary to the value assignments for inputs, to reflect any differences between the old operation's inputs and those of the new one.

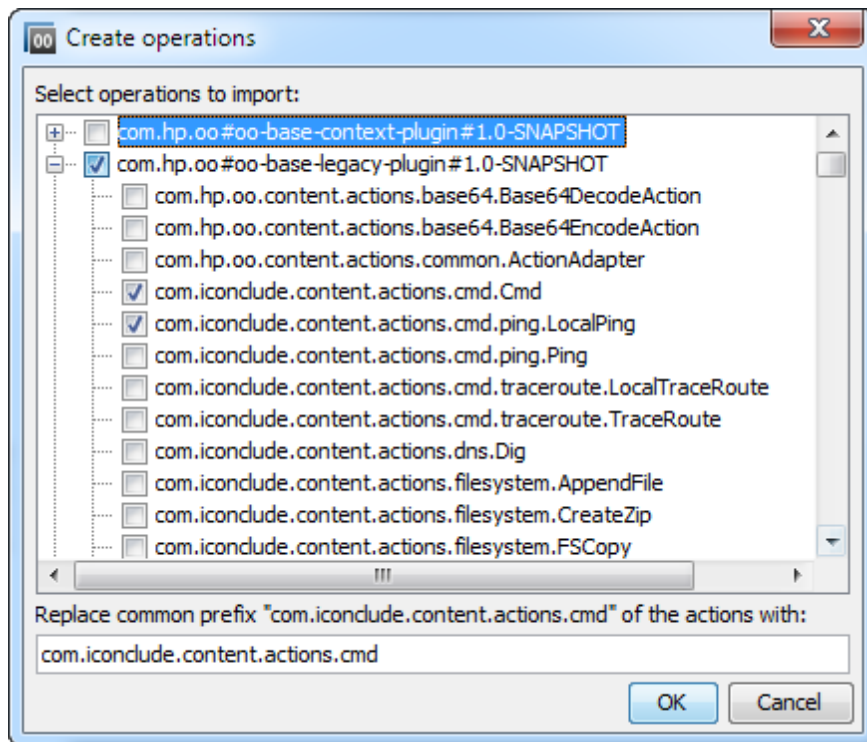
## Create an operation by importing an action plug in

The most direct way to create an operation from an action plug in is to import it and create the operation at the same time.

1. Create and pack an action plug in, so that it includes the actions that were developed. For information about how to develop an action plug in, see the *Extension Developers Guide*.
2. In Studio, right-click the folder where you want to create the new operation, and select **Import plug in**.
3. In the Import plug in dialog box, browse to and select the HP OO plug in that you want to import.



4. Select the **Import and create new operation** check box and click **OK**.
5. In the Create Operations dialog box, expand the plug in containing the actions that you need, and select the actions that you want use to create the operation from.



**Note:** If a plug in contains multiple actions, you can select more than one action, to create multiple operations.

For each action that you selected, a new operation is created in the folder where you right-clicked.

In each operation, the information about the action plug in is displayed in the **Operation Fields** section at the top of the **Inputs** tab, in the operation's Properties window.

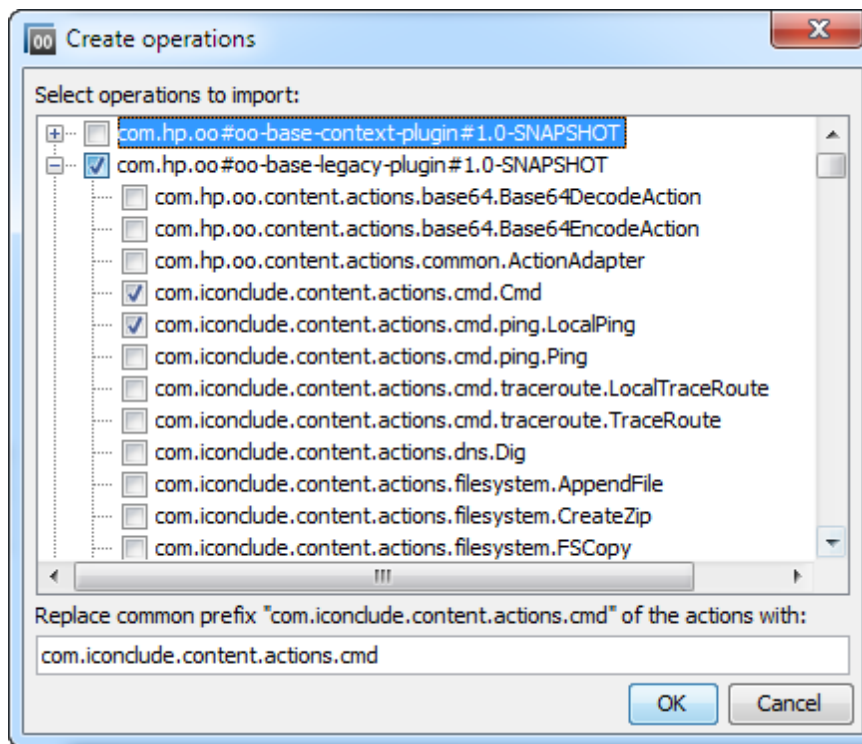
Operation fields	
Group Id:	com.hp.oo
Artifact Id:	oo-excel-legacy-plugin
Version:	1.0-SNAPSHOT
Action Name:	com.iconclude.content.actions.excel.ops.ModifyExcelData

6. Save the operation.

## Create an operation from an imported action plug in

After action plug ins have been imported to Studio's repository, you can create operations from the actions in them.

1. In Studio, right-click the folder where you want to create the new operation, and select **New > Operation**.
2. Browse to locate the plug in, from Studio's repository, and click **OK**.
3. In the Create Operations dialog box, select the action that you want use to create the operation from.



**Note:** If the plug in contains multiple actions, you can select more than one action, to create multiple operations.

For each action that you selected, a new operation is created in the folder that you right-clicked on.

In each operation, the information about the action plug in is displayed at the top of the **Inputs** tab, in the operation's Properties window.

## Import action plug ins

It is possible to simply import action plug ins to Studio's repository, so that they are available for you or other authors to create operations from, at a later date.

1. In Studio, select **File > Import plug in**.
2. In the Import plug in dialog box, browse to and select the HP OO plug in that you want to import



to Studio's local Maven repository.

3. Click **OK**. The plug in is available for authors to create operations from. For more information, see *Create an operation from an imported action plug in*.

## Create a manual operation

A manual operation is one that offers a choice of actions. The user will need to select an action at runtime.

To create a manual operation, you copy the manual operation template from the base content and define the actions that will be made available to the user.

1. In the **Content Packs** pane, select the manual operation template, located in the content pack, for example, `base-cp/Library/Utility Operations/Manual`.
2. Select **Edit > Copy**.
3. Select the location in the project tree where you want to paste the copy, and select **Edit > Paste**. This operation is treated as a new object and is detached from the content pack it arrived with.
4. In the operation properties, add the actions that will be available to the user.

## Create a display operation

A display operation is one that displays information in a pop-up prompt message, but does not perform any other action. The user will just need to click **Continue** at runtime.

To create a display operation, you copy the display operation template from the base content in `Library/Utility Operations/Display Message` and define the information that will be displayed to the user.

The prompt message can include variables. For example, to tell the user what time the preceding step concluded, you could include a date/time variable (`${dateTime}`) in the message.

1. In the **Content Packs** pane, select the display operation template.
2. Select **Edit > Copy**.
3. Select the location in the project tree where you want to paste the copy, and select **Edit > Paste**. This operation is treated as a new object and is detached from the content pack it arrived with.
4. In the operation properties sheet, select the **Display** tab.
5. Click the **Display** tab in the Step Inspector.
6. Select the **Always prompt user before executing this step** check box.
7. In the **Prompt Title** box, type the prompt's label.

8. In the **Prompt Width** box, type the width of the prompt in pixels.
9. In the **Height** box, the height of the prompt in pixels.
10. In the **Prompt Text** box, type a message to the user.
11. Click **OK**, and save your changes.

## Reference Material

### Step Inspector > Display tab

In the **Display** tab of the Operation Properties sheet, you can create a user prompt that is displayed to the user.

Step Name:

Inputs | Results | **Display** | Description | Advanced | Scriptlet

☐ Always prompt user before executing this step

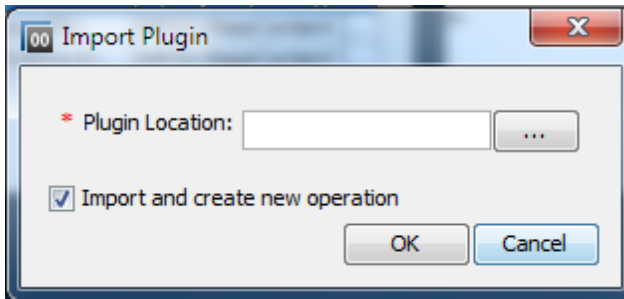
Prompt Title:

Prompt Width:  Height:

Prompt Text:

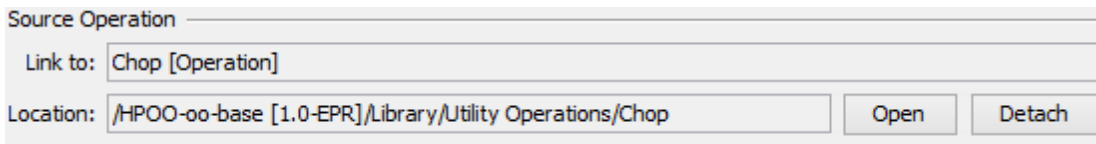
GUI item	Description
<b>Always prompt user before executing this step</b>	Select the checkbox if you want the prompt window to appear every time this step is run.
<b>Prompt Title</b>	Type the label that will appear in the title bar of the prompt window.
<b>Prompt Width</b>	Type the width of the prompt window in pixels.
<b>Height</b>	Type the height of the prompt window in pixels.
<b>Prompt Text</b>	Type the message that will appear in the body of the prompt window. You can include variables in the message. For example, <code>\${dateTime}</code> .

## Import plug in



GUI item	Description
plug in location	Browse to and select the HP OO plug in that you want to import.
Import and create new operation	Select this check box to create a new operation from the imported plug in.

## Operation Properties: Advanced tab



GUI item	Description
Link To	Displays the source operation, from which the selected operation was copied.
Location	Displays the location of the source operation.
Open	Opens the Properties sheet of the source operation.
Detach	Detaches the operation from the parent plug in.

## Finding a Flow or Operation

There are a number of ways to find the flow or operation that you need:

- Browse the folders in the **Projects** pane and **Content Packs** pane
- View the descriptions of the folders, flows, and operations
- Run a search

## What do you want to do?

### Browse the folders to find a flow or operation

The simplest way to locate a flow or operation is to browse through the folders.

If the folders have been named and structured correctly, this should help you to find what you need.

### Use the descriptions to locate a flow or operation

You can view the description in an operation or flow, to see if it is the one you need.

- To view the description of an operation, open the operation in the authoring pane and click the **Description** tab.
- To view the description of a flow, open the flow in the authoring pane, and click **Properties** (at the bottom of the pane), and then click the **Description** tab.

**Note:** Alternatively, it is possible to right-click the flow or operation in the **Projects** pane or **Content Packs** pane and select **Properties**.

### Generate documentation to find a flow or operation

You can also use the Generate Documentation feature to gather this information for many flows and operations into one place. For more information on Generate Documentation, see ["Generating Documentation about Flow and Operations" on page 245](#).

### Search for an operation

Using the **Search** pane, you can perform a full-text search throughout the Library. You can search for flows and operations by searching on the name or other field properties.

1. Click the **Search** tab at the bottom of the Studio window, to open the **Search** pane.
2. From the **Search** list, select a field to search on.

**Note:** To include all the fields in your search, leave the **Search** list set to **<all fields>**.


3. In the **for** text box, type the text you want to search for.

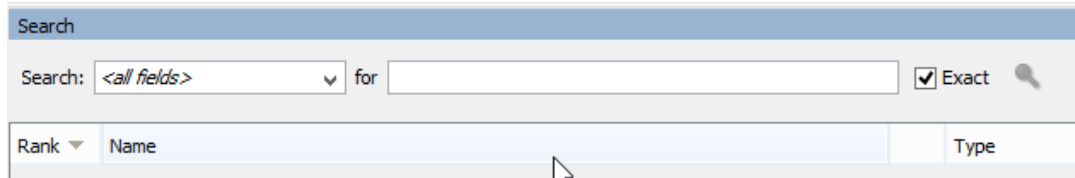
If the search string contains a space, you can specify either an exact search or a search that includes spaces. An exact search treats the entire string, including any spaces that it contains, as a single search value. If you insert space, the search looks for all of the strings that are separated by spaces.

**Note:** The search for the search string you type is not case-sensitive.

4. Define how the search treats spaces:

- To specify a search that treats spaces as part of a single search string, select the **Exact** checkbox.
- To specify a search that treats spaces are separators for alternate search strings, clear the selection from the **Exact** checkbox.

5. Click the search button .



6. Check the text in the **Description** tab to identify the operation that you need.

In search results, the description is taken from the **Description** tab of the operation and includes information that is vital to getting the most from your use of the operation, including:

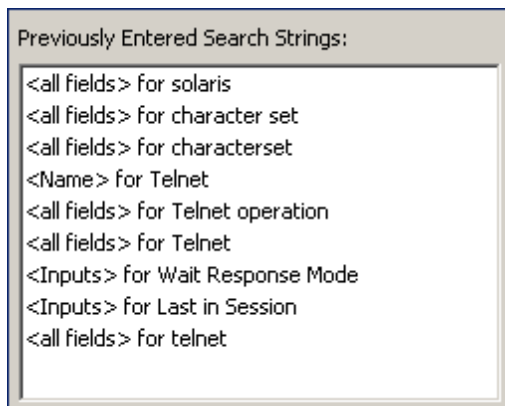
- The kind of information the operation's inputs need.
- The information that the results include.
- The operation's requirements and assumptions.

To read the entire description, see the **Description** tab in the operation's **Properties** sheet.

## Search using a previous search command

1. In the **Search** pane, click the **Search History** button.

The Previously Entered Search Strings window opens.



This window contains a list of up to 25 previous search commands.

The format of the search commands in the list is "<field name> for search text".


2. Double-click a search command from the list to run it. It is then added to the top of the list.

## Sort search results

Click the column header in any of the columns, to sort the search results by that parameter.

## Search with Lucene syntax

To target more-specific results, you can construct a search with the Apache Lucene syntax. For more information on the Lucene search syntax, see the Apache Software Foundation Web site.

1. Click the **Search** tab at the bottom of the Studio window, to open the **Search** pane.
2. From the **Search** drop-down list, select **<with Lucene query>**.
3. In the **for** text box, type your query, using Lucene search syntax, and then click the search button .

The simplest Lucene search syntax is:

```
<searchable_field_name>:<string to search for>
```

Tips for searching:

- The search uses a Boolean AND. If you type two words with AND, the search returns only operations or flows that contain both words. If you type two words without AND, the search finds any results that include either word.
- To obtain results that include only a string that has a space in it, such as in the search category:database server, enclose the string in quotation marks: category:"database server"

You can search for the following field names. Note that this list includes sample search strings.

- Flow or operation name

Examples:

```
name:Get Temp Dir  
name:Clear Temp Dir
```

- Operation type

Example:

```
type:cmd
```

- Category

Example:

category:network

- Input name

Example:

inputs:server

- Flow or operation UUID

Example:

id:1234-3453-3242-32423

- String contained in flow or operation descriptions

Example:

description:clear

## Access an operation from the Search pane

You can work with operations and flows directly from the search results, opening them for editing or adding them to a flow that is open in the authoring pane.

- To open an operation's **Properties** sheet or a flow's diagram, double-click in the row of the operation, in the search results.
- To create a step from an operation in the search results, drag the operation from the **Search** pane onto a flow diagram.

## Reference Material

### Search pane

The screenshot shows the Search pane with a search bar at the top. Below the search bar, there is a table with columns: Rank, Name, and Type. The table contains one row with the text 'GUI item' under the Name column and 'Description' under the Type column.

Rank	Name	Type
	GUI item	Description

<b>Search</b>	<ul style="list-style-type: none"><li>• To run the search on one field, select the field on which to run the search.</li><li>• To include all the fields in your search, select <b>&lt;all fields&gt;</b>.</li><li>• To search using Lucene query, select <b>&lt;with Lucene query&gt;</b>.</li></ul>
<b>for</b>	Type the string you want to search for.
<b>Exact</b>	<ul style="list-style-type: none"><li>• To specify a search that treats spaces as part of a single search string, select the <b>Exact</b> checkbox</li><li>• To specify a search that treats spaces are separators for alternate search strings, clear the selection from the <b>Exact</b> checkbox.</li></ul>
<b>Rank</b>	Displays the ranking of each search result. More stars means a higher rank.
<b>Type</b>	Displays the type of item that was found, for example, a flow.
<b>Path</b>	Displays the location where the item is stored.
<b>Description</b>	Displays the description of the item, taken from the item's <b>Description</b> tab.
<b>Search History</b>	Click to display the Previously Entered Search Strings window, and reuse a search command.

## Copying Flows and Operations

There are different ways to copy flows and operations:

- If you copy a flow or operation, you can paste it in any folder that is not sealed. If you copy a flow, this copies only the flow and not the operations that comprise the flow.
- If you click **Copy Deep**, this copies not only the flow but also all the operations that comprise the flow. You would do this when you will need to modify the operations in the new flow, and do not want to affect the original operations.
- If you duplicate a flow or operation, the duplicate is automatically placed in the same folder as its original, and is named **Copy of <name>**.
- If you cut a flow or operation, you remove it from its current location, to be pasted somewhere else.



## ***Copying Operations that were Created from Action Plugins***

### **Soft copies**

When you copy an operation that is linked to an action plugin jar file, the copied operation continues to reference the original operation. If the action plugin jar file is upgraded, when you update the original operation to call the new version, the copied operations are all updated automatically. This is known as a **soft copy**.

Note that the link to the action plugin jar file is the only item that is automatically updated in the copied operations. Changes that you make to the original operation's input, output, variables, scriptlets, and so on, are not updated in the copies.

You can copy operations from a Content Pack, but keep a reference to the parent Contact Pack. If a future fix is implemented to the parent plugin, then the soft copy will also receive the fix. In certain cases, you may not want to receive fixes to the operation. In this case, you can detach the operation from the parent plug in. However you will need to manually fix the detached operations.

Soft copies have advantages and disadvantages:

- The disadvantage is that if the original operation gets deleted, the copy is parent-less and loses its link to the plugin. In this case, a new parent will have to be selected manually.
- The advantage is that if the original operation is updated with a different plugin version, the soft copy gets updated.

For more information about creating operations from action plugin jar files, see ["Creating Operations" on page 227](#).

### **Hard copies**

In versions of HP OO prior to 10.00, when you copied an operation linked to an action plugin, this created a **hard copy**, meaning that the copy was directly linked to the action plugin in the same way that the original was. When the action was updated—for example if the name of the JAR or the class was changed—this had to be updated in all the hard copied operations.

In HP OO 10.00, you can create a hard copy by creating a new operation and selecting the relevant plugin. This method creates a new operation according to the `IAction getTemplate` or the `@Action` metadata. Is not possible to create a hard-copy of an operation that also duplicates its inputs and outputs.

Hard copies have advantages and disadvantages:

- The advantage is that if the original operation gets deleted, the copy is not affected and doesn't remain parent-less.
- The disadvantage is that if the original operation is updated with a different plugin version, the hard copy does not get updated.

## Changing Between Hard Copy and Soft Copy

It is possible to detach a soft copy from its parent and make it a hard copy. In the operation's **Advanced** tab, you can detach the operation from its parent by clicking the **Detach** button. A confirmation message appears and the plugin's GAV parameters are taken from the original parent.

You can convert a hard copy into a soft copy by clicking the **Select** button and choosing a parent operation.

## Replacing a Plugin in a Hard Copy

You can search for all the hard copied operations that use a specific plugin, and choose the operation, and replace the plugin GAV parameters.

## Best Practices

If you are copying a flow and you think that you may need to modify the properties of the operations, it is best to use the **Copy Deep** command, to copy the operations as well as the flow.

If you are planning to copy a flow using the **Copy Deep** command, it is recommended to create a new folder for the flow and its operations.

## What do you want to do?

### Copy a flow or operation

1. In the **Content Packs** pane or **Projects** pane, right-click the flow or operation that you want to copy.
2. Select **Edit > Copy**.
3. Navigate to the folder in which you want to place the copy, right-click and select **Edit > Paste**.

### Duplicate a flow or operation

When you duplicate a flow or operation, the duplicate is automatically placed in the same folder as its original, and is named **Copy of <name>**.

1. In the **Content Packs** pane or **Projects** pane, right-click the flow or operation that you want to copy.
2. Select **Edit > Duplicate**.

### Deep copy a flow or operation

Deep copying a flow copies not only the flow but also all the operations that comprise the flow.

1. In the **Content Packs** pane or **Projects** pane, right-click the flow that you want to copy.
2. Select **Edit > Copy Deep**.
3. Navigate to the folder in which you want to place the flow and its operations, right-click and select **Edit > Paste**.

### Cut a flow or operation

1. In the **Content Packs** pane or **Projects** pane, right-click the flow or operation that you want to move.
2. Select **Edit > Cut**.
3. Navigate to the folder in which you want to place the flow or operation, right-click and select **Edit > Paste**.

## Finding Out How Flows and Operations are Used

You can learn more about ways to use and implement an operation or flow by looking at how it is used in existing flows. This can be done in the **References** pane.

References	
References from /My_Project/Content/Library/Health Check/Integrity Check for Table	
Object	Path
[-] Integrity Check for Table	/My_Project/Content/Library/Health Check/Integrity Check for Table
Operation: Error [return]	/My_Project/Content/Library/Operations/Error
Operation: Resolved [return]	/My_Project/Content/Library/Operations/Resolved

Studio has two kinds of references:

- References **to** the operation or flow – lists the flows that have a step created from the selected operation or flow
- References **from** the operation or flow – lists the objects (selection lists, permissions assigned to groups, system filters, and so on) that the selected operation or flow makes use of. In the case of flows, these are the operations (including subflows) from which the flow's steps were created.

## Best Practices

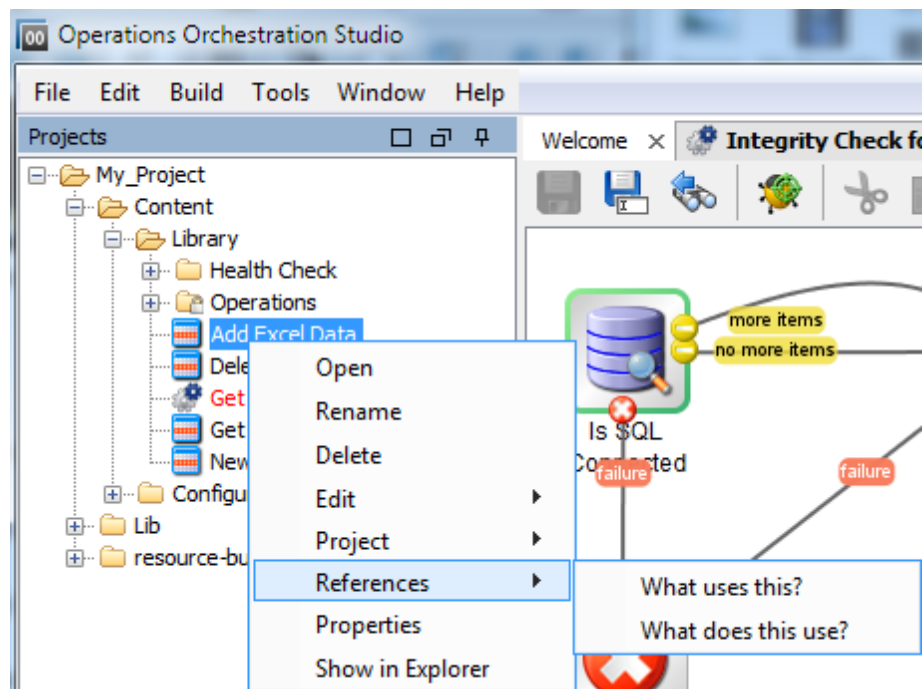
Changing an operation or flow can break other flows that use it. Before making changes to a flow or operation, use **References > What uses this?** to check whether other flows use it.

## What do you want to do?

### Identify what uses a flow or operation

1. In the **Project** pane, right-click the operation or flow.
2. Select **References > What uses this?**.

The **References** pane opens, displaying the references to the operation or flow.



### Identify what a flow or operation uses

1. In the **Project** pane, right-click the operation or flow.
2. Select **References > What does this use?**.

The **References** pane opens, displaying the references from the operation or flow.

**Tip:** The referenced flows and operations are valuable as samples that you can copy, paste, and modify.

## Reference Material

### Reference pane

In the **References** pane, you can see how an operation or flow is used in existing flows.

References	
References from /My_Project/Content/Library/Health Check/Integrity Check for Table	
Object	Path
[-] Integrity Check for Table	/My_Project/Content/Library/Health Check/Integrity Check for Table
[-] Operation: Error [return]	/My_Project/Content/Library/Operations/Error
[-] Operation: Resolved [return]	/My_Project/Content/Library/Operations/Resolved

GUI item	Description
<b>Object</b>	Displays the object that is used or uses the selected flow or operation
<b>Path</b>	Displays the location of the object that is used or uses the selected flow or operation

## Generating Documentation about Flow and Operations

**Important:** In the current version, the Create Documentation functionality is not supported at runtime. You can generate documentation from within Studio, but if you create a flow with a Create Documentation step, this step will not work at runtime.

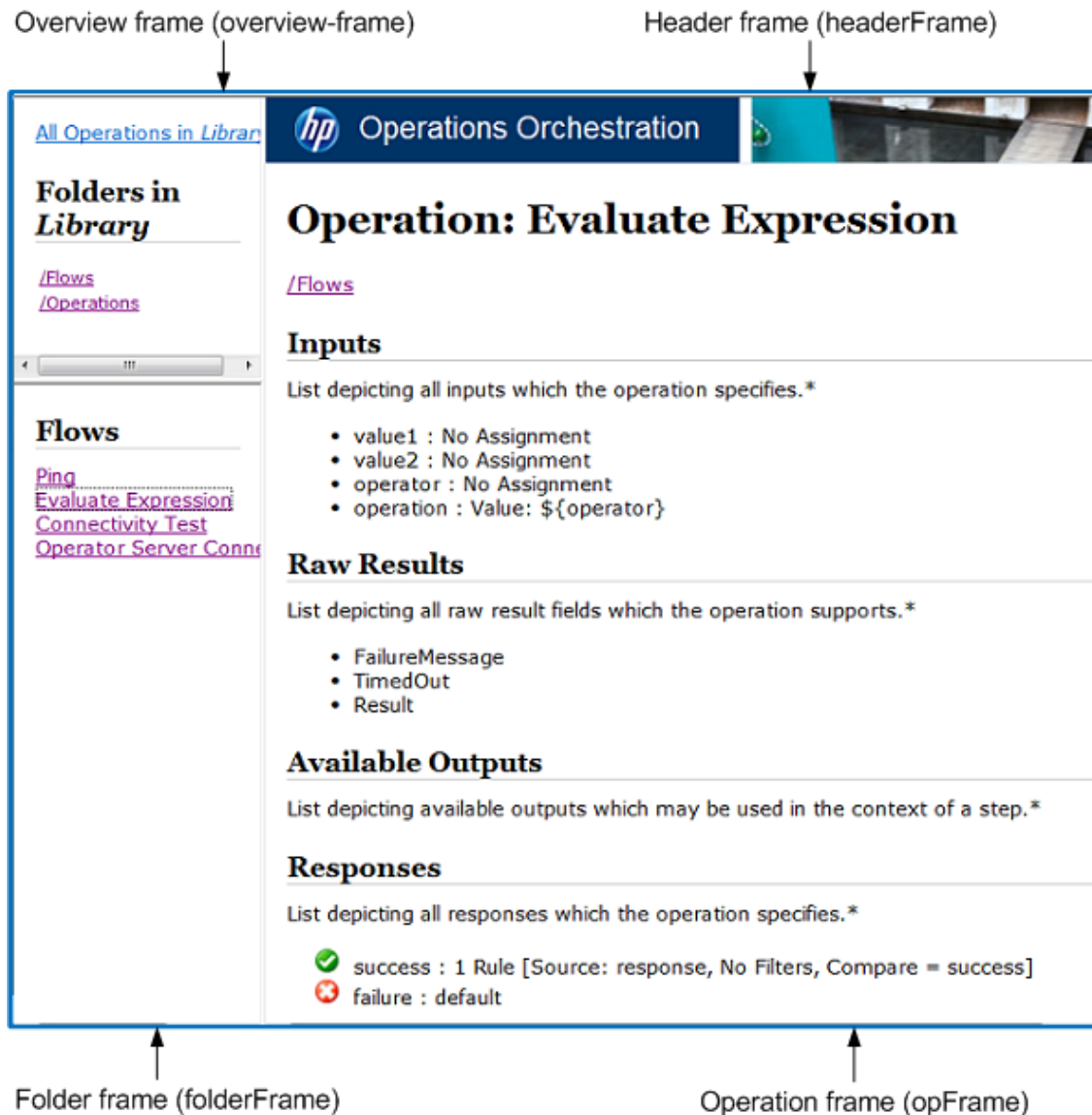
You can document flows and operations, to provide more information about them for other users:

- Export the flow as a PNG image.
- Use the Generate Documentation feature to create an HTML page with information about flows and operations.

### Structure of Generated Documentation

When you generate documentation, an HTML page named **index.html** is created. This page contains the following frames:

- **Overview** frame – In the upper-left, the **Overview** frame lists the sub-folders contained in the folder for which you generated documentation. Select a folder to display its contents in the **Folder** frame.
- **Folder** frame – In the lower-left, the **Folder** frame lists the flows and operations in the folder that is selected in the **Overview** frame.
- **Header** frame – In the upper-right, the **Header** frame contains an HP OO banner.
- **Operation** frame – In the lower-right, the **Operation** frame displays the description of the flow or operation. This is the information that was entered in the **Description** tab of the **Properties** sheet.



## What do you want to do?

### Export a flow as a PNG image

1. Open the flow in the authoring pane.
2. Right-click anywhere in the authoring pane and select **Export to PNG**.
3. Browse to the location in which to store the image and click **Save**.

### Generate documentation in standard format

You can generate documentation for a folder containing specific flows and/or operations, or for the entire **Library** folder.

1. Right-click the folder for which you want to create documentation.
2. Select **Generate Documentation > Standard Format**.
3. Browse to the location in which to store the documentation files and click **Save**. The HTML file, **index.html**, opens in a Web browser.
4. If you need to overwrite a previous version of **index.html**, click **Yes To All**.

**Note:** If you prefer not to overwrite the previous documentation, click **Cancel** and repeat the process, while saving the files in a different location.

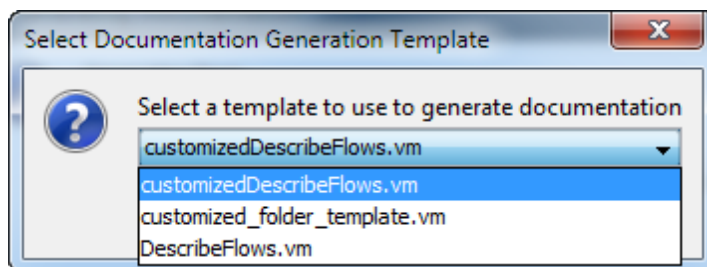
## Generate documentation in a custom format

1. Right-click the folder for which you want to create documentation.
2. Select **Generate Documentation > Custom Format**.
3. In the Select Documentation Generation Template dialog box, select the template to be used when generating the documentation.
4. Browse to the location in which to store the documentation files and click **Save**. The HTML file, **index.html**, opens in a Web browser.

## Create a customized Generate Documentation template

Documentation templates are stored in the **Studio\template** folder. They have the suffix **.vm** and can be edited in a text editor. For information about the templates, see the *Reference Material* section below.

Any new **.vm** files that you create in the **Studio\template** folder will appear in the template list in the Select Documentation Generation Template dialog box.



1. Make a copy of the relevant **.vm** template, and rename the copy.

**Caution:** Do not modify or rename the original **.vm** templates.

2. In a text editor, make your changes to the new template, and save.

3. In Studio, right-click the folder that you want to document and select **Generate Documentation > Custom Format**.
4. In the Select Documentation Generation Template dialog box, select the custom template that you created.

## Reference Material

### .vm Template Files

#### *Folder\_template.vm*

The root template, which generates a frameset and calls the following to populate it:

- **All\_folders\_template.vm** - generates a list of the sub-folders of the folder and places it in **overview-frame** (upper-left).
- **All\_ops\_template.vm** - generates a list of all operations and places it in **folderFrame** (lower left).
- **Header.html** - places the header in **headerFrame** (upper right).
- **Folder\_overview\_template.vm** - generates information about one or more operations and places it in **opFrame** (lower right).

#### *All\_folders\_template.vm*

Generates a table of contents for the folders.

- **Header.css** – style sheet used for general fonts, colors, and so on.
- **All\_ops\_template.vm** – generates a list of all operations and creates a link to display it in **folderFrame** (lower left).
- **Folder\_contents.vm** – generates a list of the selected folder's contents and creates a link to display it in **folderFrame** (lower left).

#### *All\_ops\_template.vm*

Generates a table of contents for all operations and the documentation for every child operation.

- **Header.css** – style sheet used for general fonts, colors, and so on.
- **Op\_template.vm** – generates and creates a link to display it in **opFrame** (lower right).



## ***Folder\_overview\_template.vm***

Generates a tabular summary describing the contents of a folder.

- **Header.css** – style sheet used for general fonts, colors, and so on.
- **Folder\_contents.vm** – generates and creates a link to display it in **folderFrame** (lower left).

## ***Op\_template.vm***

Generates documentation for a single operation.

- **Header.css** – style sheet used for general fonts, colors, and so on.
- **Folder\_template.vm** – generates and creates a link to display it in same frame (up to parent folder).
- **Folder\_contents.vm** – displays the folder contents in **folderFrame**.

## ***Flow\_template.vm***

Generates the documentation for a single flow.

- **Header.css** – style sheet used for general fonts, colors, and so on.
- **Flow\_template.vm** – generates and creates a link to display it in same frame (up to parent folder).
- **Folder\_contents.vm** – generates a list of the folder contents and creates link to display it in **folderFrame** (lower left).
- **Op\_template.vm** – generates and creates a link to display it in **opFrame** (lower right).

## ***Folder\_contents.vm***

Generates a table of contents for a single folder.

- **Header.css** – style sheet used for general fonts, colors, and so on.
- **Op\_template.vm** – generates and creates a link to display it in **opFrame** (lower right).

## ***Header.html***

The Hewlett-Packard banner.

## ***Hp\_rockwell.css***

Style sheet for the Hewlett-Packard banner.

## ***Hp\_steps\_307x39.jpg***

Graphic for the Hewlett-Packard banner.

## ***Logo\_hp\_smallmasthead.gif***

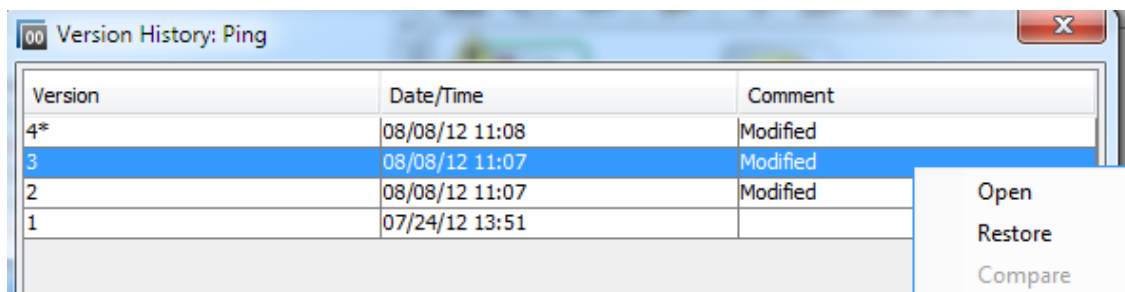
Logo for the Hewlett-Packard banner.

# **Managing Version History of Flows and Operations**

Each time that a configuration item, flow, or operation is saved, a new version of the item is created. The Version History dialog box lists these versions, and lets you:

- View an earlier version of an item
- Save the earlier version under a different name
- Restore an item to its earlier version
- View the differences between two versions

**Note:** Version history functionality only applies to your local authoring work in Studio, and not to versions saved to a source control management repository. Working with version history is something you would do *before* committing your work into the source control repository. If you have committed a flow to a shared repository, it is *not* recommended to use the HP OO local version history control function to revert to a previous version.



Version	Date/Time	Comment
4*	08/08/12 11:08	Modified
3	08/08/12 11:07	Modified
2	08/08/12 11:07	Modified
1	07/24/12 13:51	

Open


Restore

Compare

## What do you want to do?

### Open an earlier version of a configuration item, flow, or operation

When you open an earlier version of an item for viewing, you can save the earlier version under a different name.

1. Right-click the configuration item, flow, or operation and select **Show History**.
2. Right-click the version you need, and then click **Open**. The selected version is opened in the authoring pane.
3. To preserve the version that you have opened, click the **Save As**  button and give it a unique name. The two versions of the project are saved separately.
4. Click **OK** to close the Version History dialog box.

### Restore a configuration item, flow, or operation to a previous version

This procedure restores a configuration item, flow, or operation to an earlier version. To keep both the current and earlier versions, see *Open an earlier version of a configuration item, flow, or operation*.

1. Right-click the configuration item, flow, or operation and select **Show History**.
2. Right-click the version you want to revert to, and then click **Restore**. The version that you are restoring to is opened in the authoring pane.
3. Click **OK** to close the Version History dialog box.
4. Save the project. The opened version is saved over the current version.

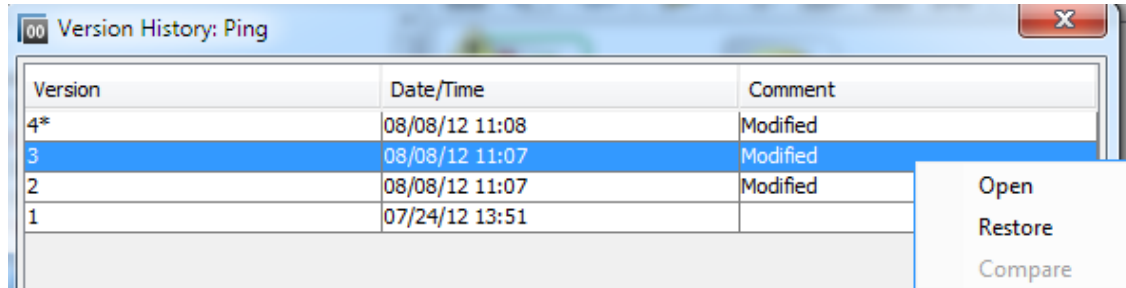
### Compare versions

The Version History dialog box also enables you to compare versions of a configuration item, flow, or operation. The current version is displayed on one side, and a previous version is displayed on the other.

1. Right-click the configuration item, flow, or operation and select **Show History**.
2. Hold down the CTRL key and select both the current version (on the top line) and an earlier version.
3. Right-click and select **Compare**. The differences between the current state of the item and the earlier version are displayed.
4. Click **OK** to close the Version History dialog box.

## Reference Material

### Version History dialog box



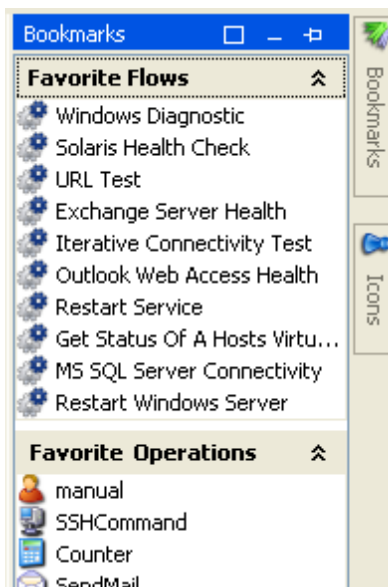
Menu item	Description
Version	The number of the project version, generated automatically.
Date/Time	The date and time that the version was changed.
Comment	The type of change that was made.

## Bookmarking Flows and Operations

The **Bookmarks** pane makes it easier to find and use the operations and flows that you use frequently. Bookmarked flows and operations are still available in their normal location in the Library.


Adding a flow or operation to the **Bookmarks** pane makes it available from the right-click menu in the authoring canvas.

You can export and import bookmarks from one installation of Studio to another.



## What do you want to do?

### Add a bookmark

1. Click the **Bookmarks** tab in the upper-right of the Studio window, to open the **Bookmarks** pane.
2. To keep the pane open, click the **Pin**  icon in the upper-right-hand corner of the pane.
3. Drag a flow or operation from the Library or the **Search** pane results to the appropriate shelf of the **Bookmarks** pane.

### Add a shelf to the Bookmarks pane

The **Bookmarks** pane has two default shelves, for flow and operations, but you can add customized shelves to organize your bookmarks.

1. Right-click in the title bar of one of the shelves of the **Bookmarks** pane, and then select **Add**.
2. Enter a name for the shelf and click **OK**.

### Rename a shelf

1. Right-click in the title bar of the shelf that you want to rename, and then select **Rename**.
2. Enter a new name for the shelf and click **OK**.

### Remove a shelf

1. Right-click in the title bar of the shelf that you want to remove, and then select **Remove**.
2. Click **Yes** in the confirmation window.


### Hide/show a shelf

1. Right-click in the title bar of the shelf that you want to hide, and then select **Hide**. The shelf is no longer visible in the **Bookmarks** pane.
2. To show the hidden shelf, right-click in the **Bookmarks** pane and select **Show**, and then select the name of the hidden shelf.
3. To show all the hidden shelves, right-click in the **Bookmarks** pane and select **Show All**.

### Move a shelf up or down

Right-click in the title bar of the shelf that you want to move up or down, and then select **Move Up** or **Move Down**.

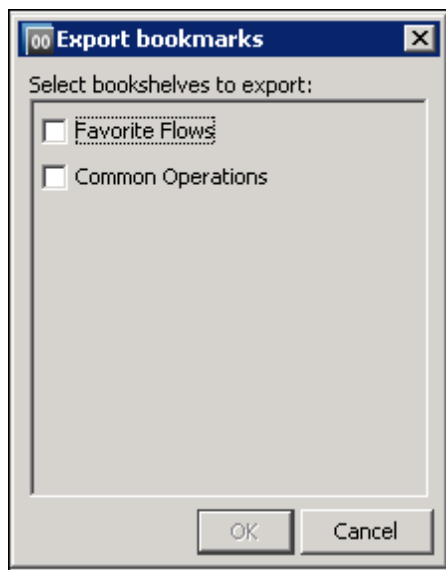
## Collapse/expand a shelf

1. To collapse a shelf, click the double chevrons  in the shelf's title bar. The shelf title is visible but the bookmarks in the shelf are hidden.
2. To expand the shelf, click the double chevrons again.

## Export bookmarks

You can export your bookmarks from one installation of Studio and import them to another.

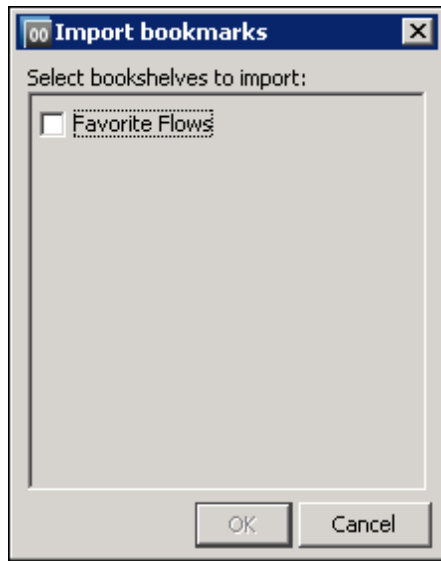
1. Right-click anywhere inside the **Bookmarks** pane, and then select **Export**.
2. In the Export bookmarks dialog box, select the bookshelves that you want to export.



3. Click **OK**. The Select export file dialog box appears.
4. Browse to the location where you want to save the bookmarks, and enter a name for the file.
5. Click **Save**.

## Import bookmarks

1. Right-click anywhere inside the **Bookmarks** pane, and then select **Import**.
2. In the Select import file dialog box, locate and select the bookmarks file, then click **Open**.
3. In the Import bookmarks dialog box, select the bookshelves that you want to import, and then click **OK**.




During import, bookmarks in the same bookshelf are merged, based on the UUID. Existing bookmarks remain, no duplicates are created, and new entries are added to the bookshelf.

## Reference Material

### Bookmarks menu

When you right-click in the **Bookmarks** pane, the Bookmarks menu is displayed. The items shown in the menu vary, depending on the item that was selected when you right-clicked.

Menu item	Description
<b>Add</b>	Adds a new shelf to the <b>Bookmarks</b> pane.
<b>Remove</b>	Removes the selected shelf from the <b>Bookmarks</b> pane.
<b>Rename</b>	Renames the selected shelf in the <b>Bookmarks</b> pane.
<b>Move Up</b>	Moves the selected shelf higher in the <b>Bookmarks</b> pane.
<b>Move Down</b>	Moves the selected shelf lower in the <b>Bookmarks</b> pane.
<b>Hide</b>	Hides the selected shelf in the <b>Bookmarks</b> pane.
<b>Show</b>	Lets you select a hidden shelf, in order to show it in the <b>Bookmarks</b> pane.
<b>Show All</b>	Shows all hidden shelves in the <b>Bookmarks</b> pane.
<b>Collapse</b> 	Collapses the shelf in the <b>Bookmarks</b> pane, so that the title is visible but the bookmarks are hidden.
<b>Import</b>	
<b>Export</b>	

# Troubleshooting for Those Upgrading from HP OO 9.x

## Where's the Studio User Interface Item?

If you are used to working with HP OO 9.x and cannot locate a user interface item in Studio, use these troubleshooting hints to help you find what you're looking for.

### Where's the repository?

HP OO no longer uses repositories. Files are stored locally on your file system and it is recommended to use a source control application for collaboration.

### Where are the Check In Check Out buttons and the My Changes/Checkouts pane?

You can commit and checkout from the changes panel if the user is connected to source control. See [Working with Source Control](#).

### Why do the Projects and Content Packs panes seem to contain the same items?

The **Projects** pane and the **Content Packs** pane are different:

- The **Projects** pane contains the *editable* flows, operations, and other HP OO objects that you can use in the project.
- The **Content Packs** pane contains *read-only* flows, operations, and other HP OO objects. You can use these objects in your project but you cannot edit them. If you want to edit any of these objects, copy them into the **Projects** pane.

### Why can't I create operations?

It still exists but now it creates an operation. What we removed is the option to create builtin operations such as HTTP, SSH, Command line etc. Those have to be copied from an existing template operation.

You cannot create built-in operations, such as HTTP, SSH, or Command line. You will need to copy them from an existing template operation, and create a new operation from an action within the plugin. For more information, see ["Creating Operations" on page 227](#).

### Why can't I create sleep scriptlets?

Sleep scriptlets have been deprecated. In HP OO 10.00, scriptlets must be written in Rhino.

### Where is the Categories domain term?

There is now a **Configuration\Categories** folder, where you can store categories for classifying flows. This replaces the **Categories** domain term.



## Comparison of versions HP OO 9.x and 10.00

Task	How it was done in HP OO 9.x	How it is done in HP OO 10.00
Creating operations	Use the <b>New &gt; Operation</b> menu option, and select the type of operation.	Import actions plugins, or create operations from imported action plugins.  See <a href="#">"Creating Operations" on page 227</a> .
Checking flows into a shared repository	Use the <b>Check In</b> button within Studio.	Save projects locally on the file system, and commit them to a shared repository, using a source control management tool.  See <a href="#">"Managing Projects using a Source Control Tool" on page 1</a> .
Deploy and run a flow	Open the flow in the Central application, and run the flow.	Release the flow as a content pack and deploy it to the HP OO server, via API.  See the <i>HP OO Installation and Deployment Guide</i> and the <i>HP OO Application Program Interface (API) Guide</i> .
Create multi-instance steps	Right-click a step and select the <b>Toggle Multi-instance</b> option to turn it into a multi-instance step. Then, create multiple loops for the different targets of the step.	Drag the <b>Multi-instance</b> icon on the <b>Step</b> palette to the authoring canvas. Add one or more subflows or operations to the multi-instance lane, and set multiple targets for the step via an input list of values.  See <a href="#">"Creating a Flow with Multi-Instance Steps" on page 192</a> .
Create actions for operations	Create IAction implementation classes, compile them into a <b>.dll</b> or <b>.jar</b> file, copy the <b>.dll</b> or <b>.jar</b> file into your Web service, and import the Web service into Studio.	Create and pack an action plugin, import it into Studio, and create a new operation from it.  See <a href="#">"Creating Operations" on page 227</a> .
Create categories to use for classifying flows	In the domain term called <b>Categories</b> , in the <b>Configuration\Domain Terms</b> folder, add a new row for the new category.	Create a new category in the <b>Configuration\Categories</b> folder.

