

# HP Operations Orchestration

Software Version: 10.50  
Windows Operating System

## Studio Authoring Guide

Document Release Date: Sept. 2015  
Software Release Date: Sept. 2015

## 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 2005-2015 Hewlett-Packard Development Company, L.P.

### Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

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

## 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: <https://softwaresupport.hp.com/group/softwaresupport/>

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: <https://softwaresupport.hp.com/>

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)

**HP Software Solutions Now** accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is

<http://h20230.www2.hp.com/sc/solutions/index.jsp>

## About this 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. Because this content was originally created to be viewed as online help in a web browser, some topics may not be formatted properly. Some interactive topics may not be present in this PDF version. Those topics can be successfully printed from within the online help.

# Contents

Welcome to HP Operations Orchestration Studio Authoring Guide .....	9
Visual Overview of HP OO Studio .....	9
Getting Started with HP OO Studio – Major Steps in the Workflow .....	23
Adjusting the Appearance of the HP OO Studio Window .....	23
Authoring Best Practices .....	26
General Best Practices .....	26
Best Practices for Sharing Content .....	27
Best Practices for Naming .....	29
Best Practices for Flows .....	31
Best Practices for Operations .....	32
Best Practices for Steps .....	34
Best Practices for Transitions .....	36
Best Practices for Inputs .....	37
Best Practices for Debugging .....	38
Best Practices for Configuring Studio .....	38
Best Practices for Descriptions .....	39
Best Practices for Source Control Management .....	46
Subversion .....	46
Git .....	47
Working with Different Languages in HP OO Studio - Localization .....	47
Content Pack Localization .....	48
Project Localization .....	48
cp.properties File .....	48
Working with Projects .....	51
Managing Projects .....	51
Managing Folders in the Project Pane .....	58
Working with Source Control in HP OO Studio .....	60
What is Source Control in HP OO Studio? .....	60
Reference Material .....	61

Working with Subversion Source Control Management .....	62
Terminology .....	62
Creating an Initial Source Control Repository .....	63
Working with Multiple Authors in SVN .....	69
Enforce Locking Policy .....	70
Troubleshooting SVN .....	86
Working with Git Source Management System .....	88
Git Terminology .....	88
Getting Started with Git in Studio .....	90
Git Branch support .....	90
Understanding the Git Repository Log .....	90
Conflict Handling in Git .....	94
Authentication Options .....	96
<b>Working with Content Packs .....</b>	<b>129</b>
Importing Content Packs to a Project .....	129
Managing Content Packs and Dependencies in a Project .....	131
Dependencies .....	131
<b>Managing Configuration Items .....</b>	<b>146</b>
Working with Configuration Items .....	146
Searching for Configuration Items .....	146
Working with Configuration Item Folders .....	146
Configuring Categories .....	149
Configuring Domain Terms .....	151
Configuring Group Aliases .....	154
Configuring Role Aliases .....	159
Configuring Scriptlets .....	161
Configuring Selection Lists .....	166
Configuring System Accounts .....	169
Configuring System Evaluators .....	172
Configuring System Filters .....	179
Configuring System Properties .....	184

<b>Authoring a Flow – Basics</b> .....	<b>189</b>
Creating a Flow – Step-by-Step .....	189
Creating a New Flow .....	192
Creating Steps in a Flow .....	196
Adjusting the Appearance of a Flow .....	203
Modifying a Flow .....	205
Creating Input .....	211
Specifying the Input Source .....	222
Evaluating Input Data .....	236
Creating Transitions .....	237
Setting Responses .....	242
Creating Outputs and Results .....	252
Setting Operation Outputs .....	252
Setting Step Results .....	256
Filtering Output and Results .....	265
Working with Variables .....	282
Creating Return Steps .....	288
<b>Advanced Authoring</b> .....	<b>292</b>
Creating a Subflow Within a Flow .....	292
Creating a Flow with Parallel Split Steps .....	295
Creating a Flow with Multi-Instance Steps .....	299
Using Scriptlets in a Flow .....	308
Using Regular Expressions in a Flow .....	313
<b>Searching Content on HP Live Network from Studio</b> .....	<b>318</b>
Setting up the HPLN Connection from Studio .....	318
Searching HP Live Network from Studio .....	319
Search Types .....	319
Search Results .....	320
<b>hpln-index-generator Tool</b> .....	<b>321</b>
<b>Validating Content</b> .....	<b>322</b>

Validating Flows in the Problems Pane .....	322
Testing and Debugging a Flow .....	324
Debugging Flows that Require User Authentication .....	325
Debugging Complex Flows .....	336
Debugging a Remote Central with Studio .....	337
Typical Workflow .....	339
Import Certificates Automatically with a Remote Debugger Connection .....	343
Debugging a Flow on a Remote Central .....	344
<b>Exporting a Content Pack .....</b>	<b>347</b>
Content Pack Versioning Lifecycle .....	347
Content Pack Versioning Without Revision Control .....	348
Content Pack Versioning With Revision Control .....	349
<b>Managing Flows and Operations .....</b>	<b>360</b>
Creating Operations .....	360
Finding a Flow or Operation .....	369
Copying Flows and Operations .....	375
Changing From a Soft Copy to a Hard Copy .....	377
Replacing a Plugin in a Hard Copy .....	377
Finding Out How Flows and Operations are Used .....	378
Generating Documentation about Flow and Operations .....	380
Managing Version History of Flows and Operations .....	385
Bookmarking Flows and Operations .....	388
<b>Configuring Studio Properties .....</b>	<b>393</b>
Defining the default value of Assign from and Assign to .....	398
Changing the default value of Otherwise to Use Constant .....	398
Showing special characters in the Context inspector .....	399
<b>Troubleshooting .....</b>	<b>406</b>
Troubleshooting for Those Upgrading from HP OO 9.x .....	406
Where's the Studio User Interface Item? .....	406
Comparison of versions HP OO 9.x and 10.x .....	407
HPLN Troubleshooting .....	408

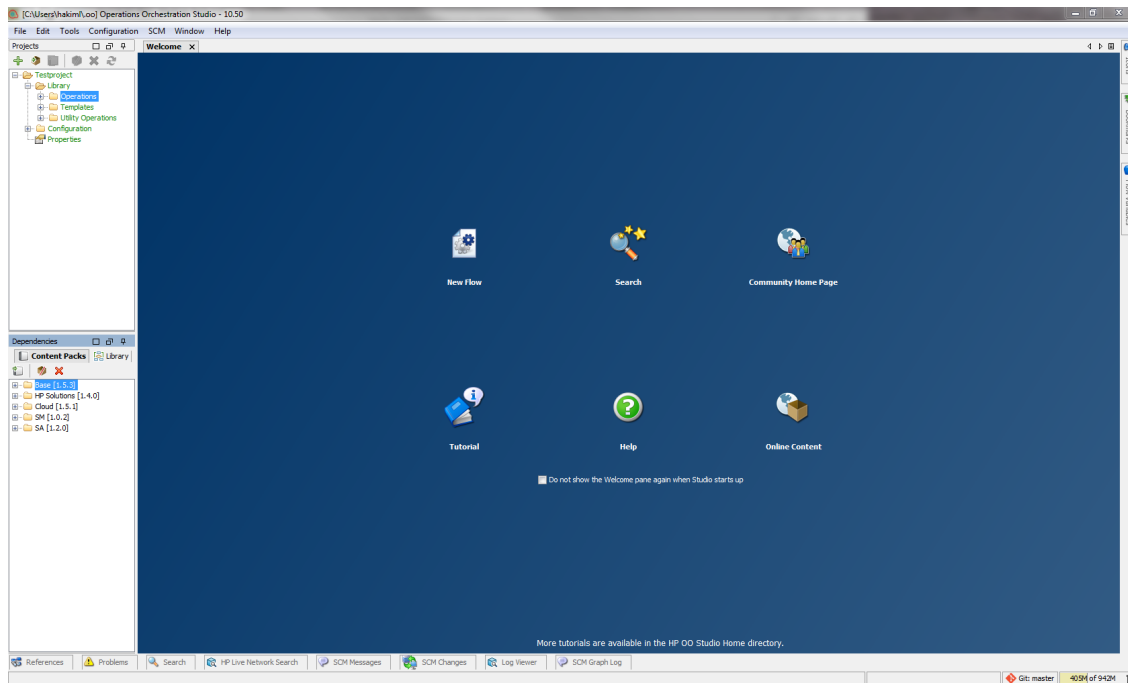
GIT Troubleshooting .....	413
Viewing Studio Errors in the Log Viewer .....	414



# 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



The main elements of Studio are:

- **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. The **Dependencies** pane contains two tabs:
  - **Content Packs** - Displays multiple trees, for multiple content packs. From this view, it is possible to close, delete, or import a content pack.
  - **Library** - Displays a single tree, with all of the content merged together under a general **Library** folder. From this view, it is possible to import a content pack.
- **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)
- **Welcome** tab (in the center). When you first open Studio, the **Welcome** tab is displayed in the authoring pane.
- **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.
- **SCM Repository Log** pane (on the bottom), displays a log of the changes made to the SCM repository (for Git only). Open this pane by clicking the **SCM Repository Log** tab. See ["Working with Git Source Management System" on page 88](#) for more information.
- **Log Viewer** pane (on the bottom), displays all the errors that have occurred in the current user session. Open this pane by clicking the **Log Viewer** tab. See ["Viewing Studio Errors in the Log Viewer" on page 414](#) for more information.

**Note:** The **Log Viewer** tab appears only after you enable it by selecting **Studio Log Viewer** from the **Windows** menu.

- **Search** pane (on the bottom), which enables you to search for a flow, operation or configuration item. Open this pane by clicking the **Search** tab.

- **HP Live Network Search pane** (on the bottom), allows you to search for relevant information and content on the HP Live network based on their HPLN profile access permissions directly from Studio. See the [Searching Content on HP Live Network from Studio](#) for more information.
- **Status bar** (on the bottom), shows the source management system the current project is connected to (Git or SVN) and its current state. In addition the total heap size available and current heap size usage is shown.

**Note:** If you cannot see the entire Studio screen, as it appears above, this may be a screen resolution issue. When working in Studio, you should set your screen resolution to a minimum of 1280x1024 pixels.

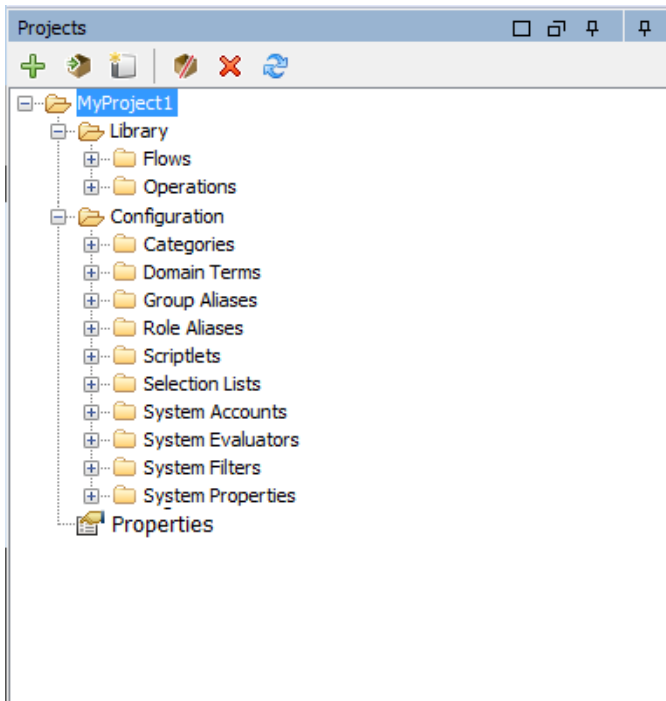
## Projects pane








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.

**Note:** You can create folders in all the configuration items under the existing Configuration Item folder structure.

- The project properties.



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 51](#).

## Dependencies pane

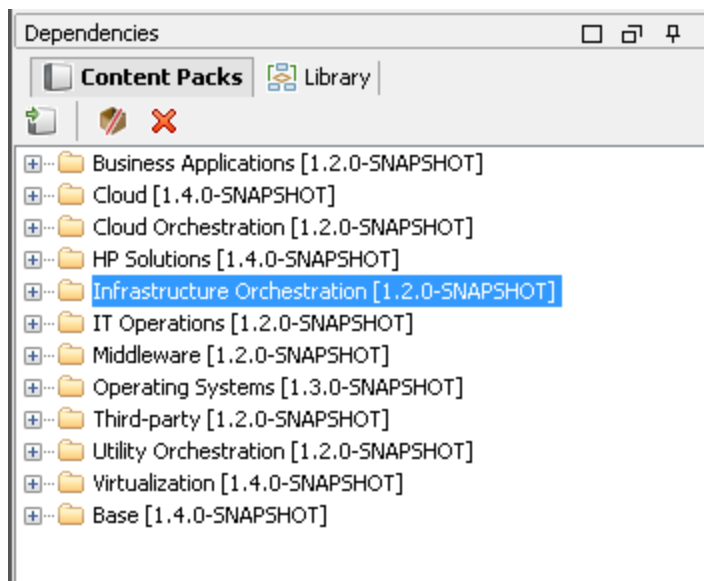
The **Dependencies** pane displays the available content packs, with folders that contain the operations and flows.

The Dependencies pane includes two view options: **Content Packs** tree view and **Library** which is an aggregated view of all the imported content packs.

When you switch the view to the **Library** pane, all the items are merged under the same tree. In addition all the configuration items are merged under a general **Configurations** folder. All the items in the tree will be merged into the same folder if they have a common path.

The next time you open Studio, it automatically opens the last view selected.

- **Content Packs:** Displays multiple trees, for multiple content packs. From this view, it is possible to close, delete, or import a content pack. When you right-click on an item, a drop-down menu with the available options for this view appears.
- **Library:** Displays all the library and configuration items, (including all the folders and sub-folders). From this view, it is possible to import a content pack. Content packs that were removed or deleted from the **Content Packs** view will automatically update the **Library** view. When you right-click on an item, a drop-down menu with the available options for this view appears.

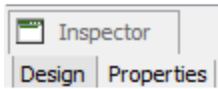


## Authoring pane

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.

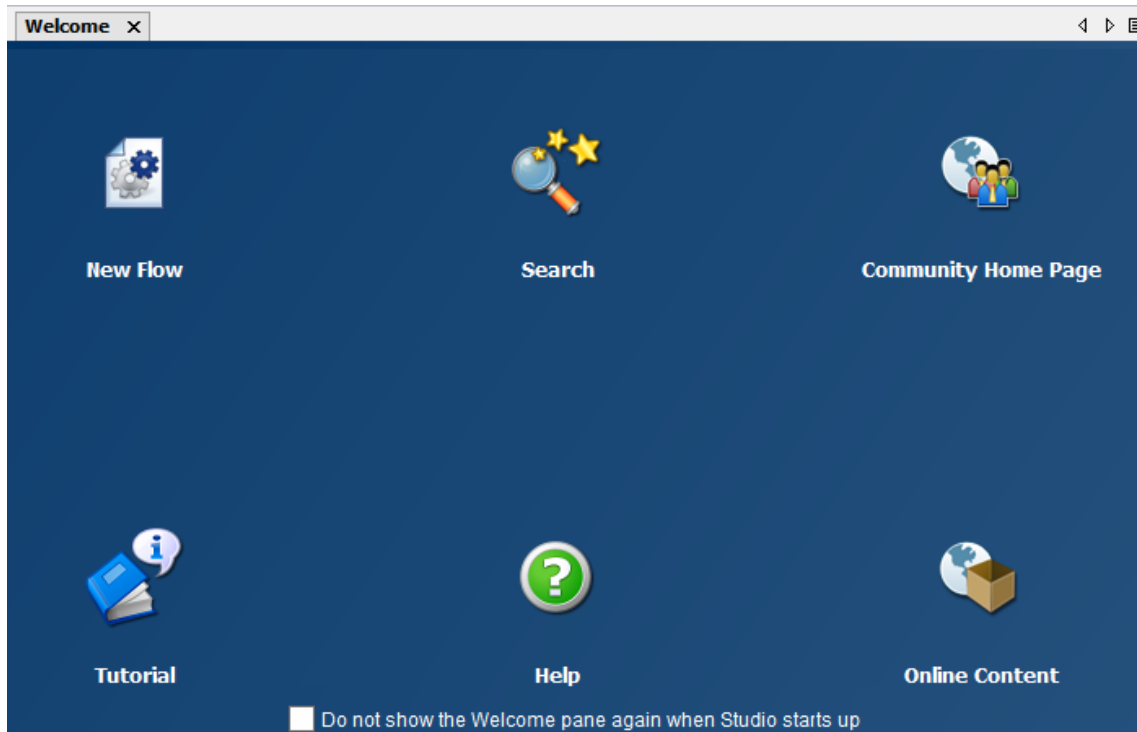


Button	What it does
<b>Step Palette</b> 	Opens the <b>Step</b> palette to drag step objects onto the canvas
<b>View Options</b> 	Opens the <b>View Options</b> palette
<b>Find this object in Library</b> 	Expands the Library tree to select the flow or operation that you're working on
<b>Debug Flow</b> 	Adds or edits Central debugger connections.
<b>Go to Step</b> 	Lets you jump to a specific step in the flow. Type the name of the step to jump to it, or the first letters of the step to select it from a list.

For information about working with the authoring pane, see "[Authoring a Flow – Basics](#)" on [page 189](#).

## Welcome tab

When you first open Studio, the **Welcome** tab is displayed in the authoring pane. If it has been closed, select **Help > Show Welcome Panel** to reopen it.



Button	Description
<b>New Flow</b>	Click to create a new flow from a predefined template.
<b>Search</b>	Click to open the <b>Search</b> pane, so you can search for flows from the content repository.
<b>Community Home Page</b>	Click to go to the HPLN Community Home Page.
<b>Tutorial</b>	Click to view the HP OO tutorials.
<b>Help</b>	Click to open the HP OO Help.
<b>Online Content</b>	Click to go to the HP OO Content Catalog download page on HPLN.
<b>Do not show the Welcome pane again when Studio starts up</b>	Select this check box to specify that the Welcome page will not be shown the next time Studio is opened.


**Note:** Links to the HP OO Help, HPLN Community home page and the online content are also available from the **Help** menu.

For more information about creating a new flow from a predefined template, see "[Creating a New Flow](#)" on page 192.









For more information about searching for a flow, see "[Finding a Flow or Operation](#)" on page 369.

## 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.

For information about working with return steps, see ["Creating Return Steps" on page 288](#).

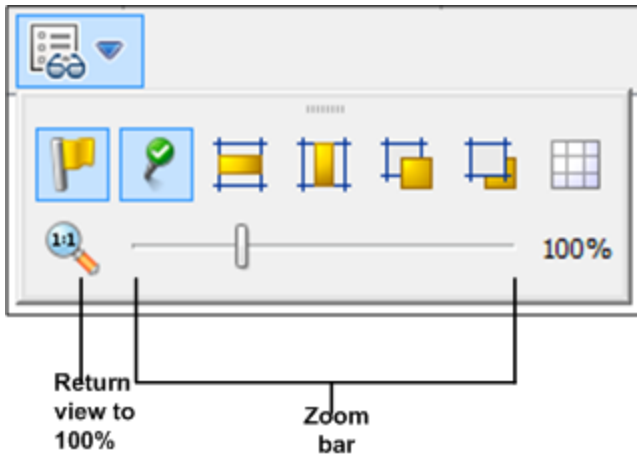
For information about parallel split steps and multi-instance steps, see ["Advanced Authoring" on page 292](#).








## 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. Using the Zoom bar, you can zoom into the flow up to 300%.





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
<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 203](#).

## 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	Sensitive ...	Required	Type	Assign From	Otherwise	Assign To
xml	<input type="checkbox"/>	<input checked="" type="checkbox"/>	xml	xml	Prompt user from the text	xml
xpathQuery	<input type="checkbox"/>	<input checked="" type="checkbox"/>	xpathQuery	xpathQuery	Prompt user from the text	xpathQuery
delimiter	<input type="checkbox"/>	<input type="checkbox"/>	delimiter	delimiter	Prompt user from the text	delimiter

For information about working with the **Properties** sheet, see ["Creating Input" on page 211](#) and ["Setting Operation Outputs" on page 252](#).

## Step Inspector

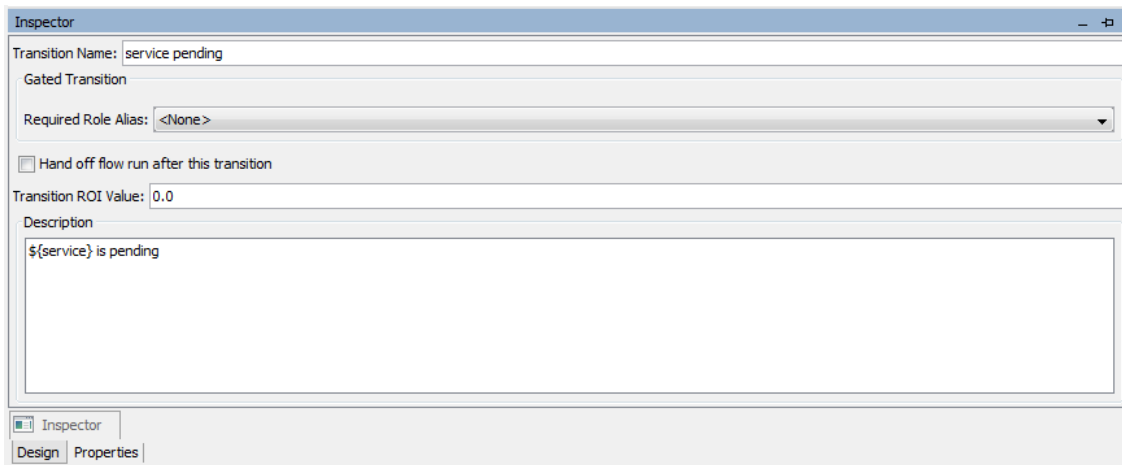
The Step Inspector is similar to the **Properties** sheet for an operation, but it relates to a single step in a flow. If you modify the properties of a step in the Step Inspector, the changes only affect this step, which is an instance of the operation.

Input	Required	Type	Assign From	Otherwise	Assign To
host	<input checked="" type="checkbox"/>	host	host	Prompt user from the text	host
baseDN	<input checked="" type="checkbox"/>	baseDN	baseDN	Prompt user from the text	baseDN
DN	<input checked="" type="checkbox"/>	DN	DN	Use the constant: CN=Users,\${baseDN}	DN
groupDN	<input checked="" type="checkbox"/>	groupDN	groupDN	Use the constant: CN=Domain Admins,CN...	groupDN
daysOld	<input checked="" type="checkbox"/>	daysOld	daysOld	Use the constant: 30	daysOld
username	<input type="checkbox"/>	username	username	Prompt user from the text	username
password	<input type="checkbox"/>	password	password	Prompt user from the text	password

For information about working with the Step Inspector, see ["Creating Input" on page 211](#).

## 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**.



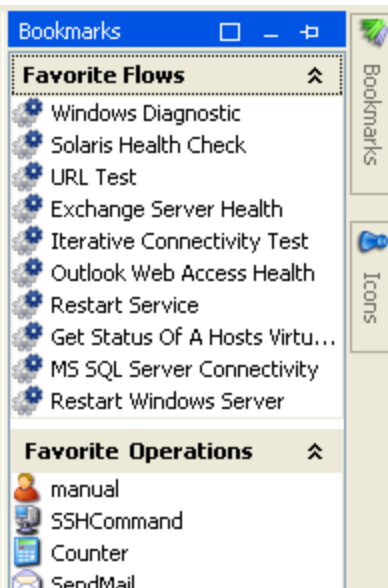
For information about working with the Transition Inspector, see ["Creating Transitions" on page 237](#).

## 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.

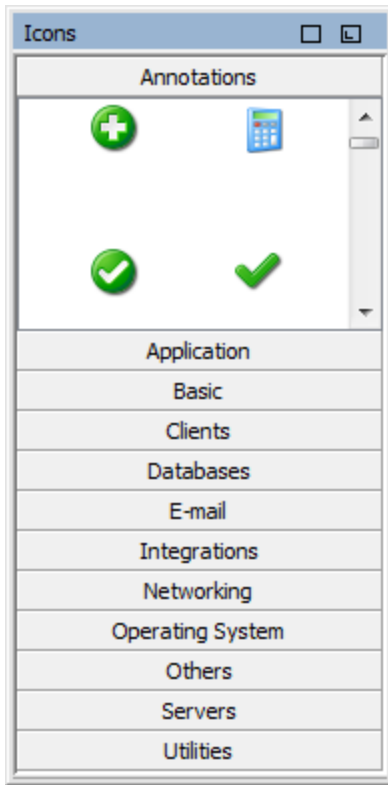
You can add flows and operations to the **Bookmarks** pane by dragging them from the Library. You can also drag flows and operations from the **Bookmarks** pane to the Projects pane, to copy them to a project.

For more information about bookmarks, see ["Bookmarking Flows and Operations" on page 388](#).



## 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 205.

## 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.

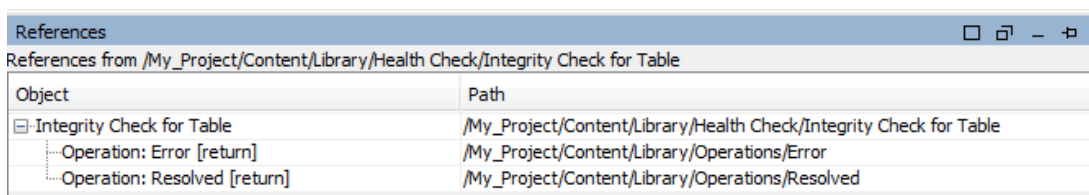
Name	#	→	👤
- destination	2		
- Step Inputs Without User Prompts	2		
- Assign From "destination" to Input "destination" in "Basic Notify"	-		
- Assign Input "destination" to "destination" in "Basic Notify"	-		
- from	2		
- Step Inputs Without User Prompts	2		
- Assign From "from" to Input "from" in "Basic 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 282](#).

## 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.

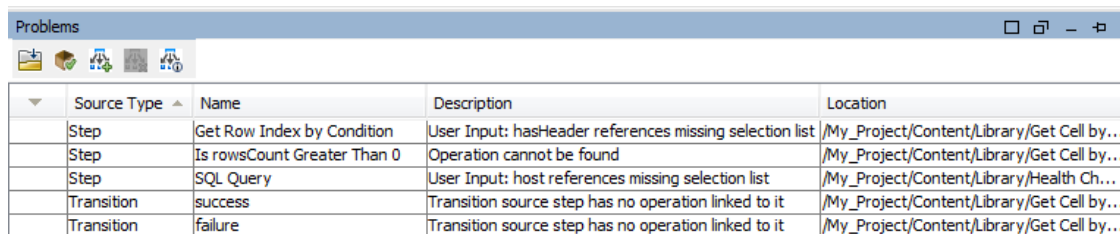


Object	Path
[-] Integrity Check for Table	/My_Project/Content/Library/Health Check/Integrity Check for Table
--Operation: Error [return]	/My_Project/Content/Library/Operations/Error
--Operation: Resolved [return]	/My_Project/Content/Library/Operations/Resolved

For information about working with the **References** pane, see ["Finding Out How Flows and Operations are Used" on page 378](#).

## 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.



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...

For information about working with the **Problems** pane, see ["Validating Flows in the Problems Pane" on page 322](#).

## 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  Exact

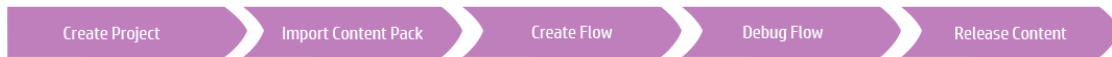
38 hits

Rank	Name	Type	Path	Description
****	SQL Restart	web	/HPOO-MicrosoftSQLServer [10.00.00-713]/Co...	f2dcfd53-a114-4aad-91c4-ce55t
****	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-4e01-8545-4f3b9
****	Is An SQL Server	flow	/HPOO-MicrosoftSQLServer [10.00.00-713]/Co...	86262296-de06-437d-9544-f79f
****	SQL Event Log Diagnostic	flow	/HPOO-MicrosoftSQLServer [10.00.00-713]/Co...	a4ab3782-6a22-482e-9875-f00e
**	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 369.

# 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 51](#).
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 129](#).

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 189](#) and ["Advanced Authoring" on page 292](#).
4. **Run and debug the flow** - Validate your flow in the Debugger.  
  
See ["Testing and Debugging a Flow" on page 324](#).
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 347](#).

## 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**.

**Note:** When you reset the Studio window, this will only be after creating or importing a new project.





## Authoring Best Practices

The following practices are recommended for authoring in Studio, especially when multiple authors are creating flows.

General Best Practices .....	26
Best Practices for Sharing Content .....	27
Best Practices for Naming .....	29
Best Practices for Flows .....	31
Best Practices for Operations .....	32
Best Practices for Steps .....	34
Best Practices for Transitions .....	36
Best Practices for Inputs .....	37
Best Practices for Debugging .....	38
Best Practices for Configuring Studio .....	38
Best Practices for Descriptions .....	39
Best Practices for Source Control Management .....	46

## 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.

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

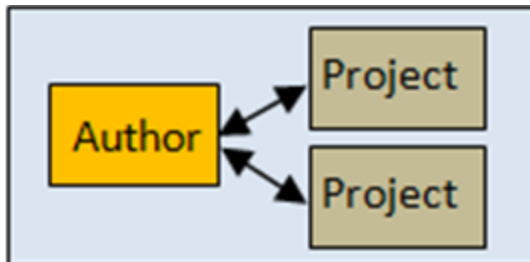
Create a document describing the naming conventions, folder structure, and other guidelines that you want flow authors to follow. For examples, see ["Best Practices for Naming" on page 29](#).

## Best Practices for Sharing Content

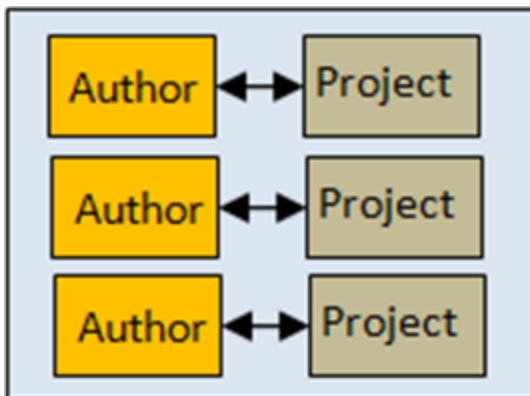
### *Identify your Environment*

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:

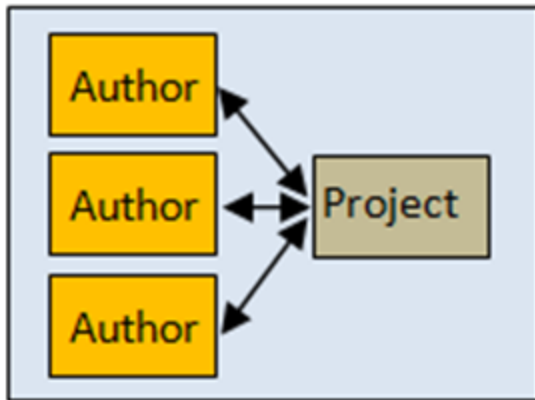
- One author, with one or more projects



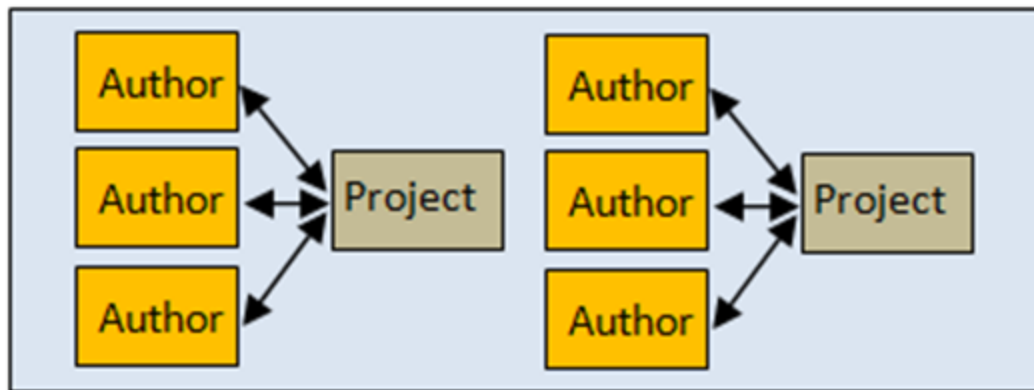
- Multiple projects, each belonging to one author



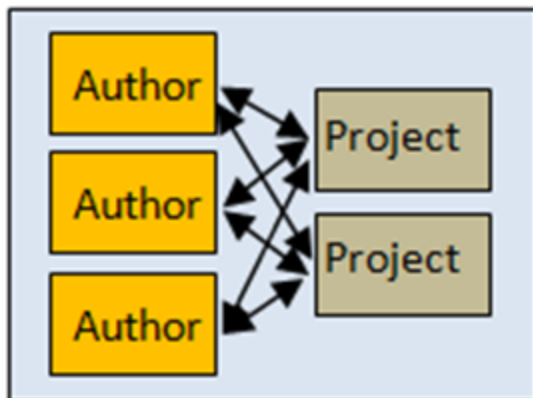
- Single project, multiple authors



- 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 129](#).

### **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:

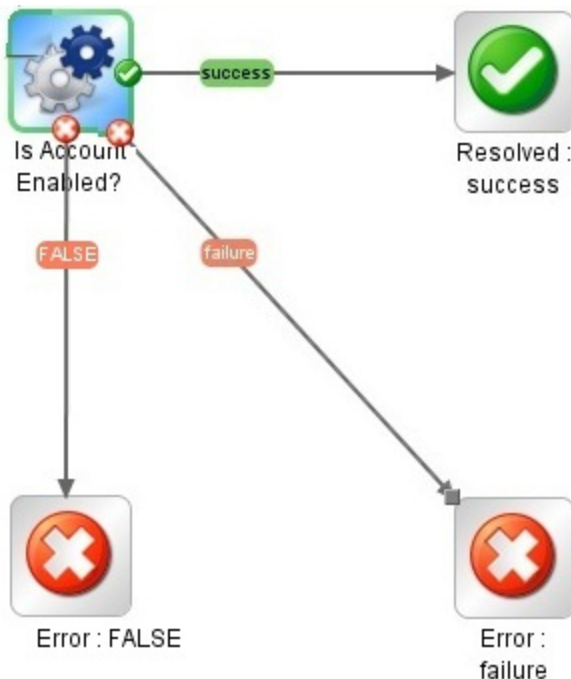
- The start step has many responses, each of which leads to another step.
- Placing the start step in the upper-left corner would cause excessive visual complexity, such as the crossing of transitions.

## 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.

### ***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.

## ***Localized Data***

For localized files that are sent as inputs (containing special characters for languages such as French, Japanese, Chinese, German, Spanish), be aware that Studio uses an UTF-8 file encoding. Other file encodings might not be recognized. It is recommended to use UTF-8 encoding for such files.

## **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 378](#).

Duplication of names of configuration items is detected as an error, which is displayed in the **Problems** pane and as a tooltip over the item. If a duplication occurs, you must rename one item or delete it.

In some situations, you might have configuration items with the same UUID. For example, if you refactor multiple projects and you end up importing a content pack that has the same items as an existing project in the workspace. Another example is if you migrated a repository from HP OO 9.x to OO 10.00 or 10.01 (prior to version 10.02), imported the result either as project or a content pack, and then imported the OO Base Content Pack.

Duplicated UUIDs can cause inconsistent behavior in Studio. It is strongly recommended to fix this issue by deleting one of the duplicated items and then fixing any broken flows that may have resulted.

Duplication of both names and UUIDs is detected throughout the whole workspace, including projects and content packs.

**Note:** Duplicate names are considered case-insensitive, for example, MySystemAccount will be duplicated with mysystemaccount.

## 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

### *Subversion*

- 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. The Enforce Locking policy prevents the flow author from making any changes to an item (flow or configuration item) unless the item is locked. This ensures that only one author can edit an item. See "[Enforce Locking Policy](#)" on page 70 for more information.
- To ensure you are always working on the latest revision, make sure you update an item before locking, modifying or committing.

- @Actions 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 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, it is recommended to create a content pack containing the plugins, for other authors to import

## Git

- Before working with Git make sure that you have no content packs in the current workspace.
- If you want to set up your own HTTP-base Git solution, use the module, or select a dedicated Git server such as [Github](#), [GitLab](#), [Gitblit](#), [Bonobo Git Server](#). All these have the following advantages - web-view of the repository contents, history, branches, and fine-grained access control.
- If you want to use Apache as the Git server, it is recommended to use the dedicated Git server module for Apache, [git-http-backend](#).

**Note:** It is not recommended to set up a Git server with HTTP(S) based on WebDAV.

# Working with Different Languages in HP OO Studio

## - Localization

Localization refers to the adaptation of software for specific environments (countries or regions). HP Operations Orchestration Studio can be localized into the following languages:

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

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

## Content Pack Localization

All text strings contained in content pack flows and operations are 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**.

## Project Localization

Projects are not localized the same way as content packs. You can only edit the default language of a project, 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. For flows, operations and configuration items, the text is stored inside the XML file, and for folders, the descriptions are stored in multiple .properties files, one for each folder. When creating a content pack, the texts are always written 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, all text strings will be empty.

## cp.properties File

When you create 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 or 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.



## What do you want to do?

### Change 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 **C:\Users\<logged in user name>\.oo\studio.properties** file.
2. Search for the following lines:
  - `user.language=`
3. Add the language locale to these fields. The following codes are valid:

Language locale code ( <code>user.language</code> )	Region locale code ( <code>user.region</code> )	Language
fr	FR	French
de	DE	German
ja	JP	Japanese
es	ES	Spanish
zh	cn	Regional Chinese

**Note:** In order to implement a different language interface, you must assign the same valid language locale from the table above 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.

### Set the 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 it is not valid, Studio uses the system locale.

Currently, Studio supports the following display languages:

Language	Language code
French	fr
German	de
Japanese	jp
Spanish	es

Language	Language code
Regional Chinese	cn

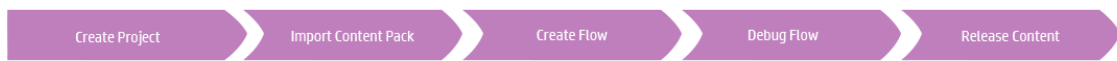
In addition, the content packs can be localized. For details, see ["Content Pack Localization" on page 48](#).

## Working with Projects

A project is a logical unit that can include flows, operations, folders, domain terms, 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



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.x, 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?

### Change the default location of the .oo folder

The **.oo** folder contains information about imported content packs or working projects and other settings. By default, the Studio **.oo** folder is located under the *Users/<user\_name>* folder.

To change the location of the **.oo** folder:

1. Open the **Studio.14j.ini** file under the Studio installation folder.
2. Change/add the location in the following property:

```
-Duser.home="C:/OO_Home"
```

### 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, ["Working with Source Control in HP OO Studio" on page 60](#).

## 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 ["Working with Source Control in HP OO Studio" on page 60](#).

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. You can import multiple projects into Studio at once.
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**.

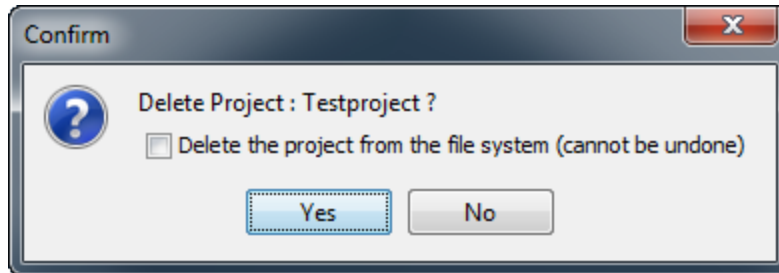
**Note:** Alternatively, you can click the **Open**  button in the **Projects** pane.

## Delete a project

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

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

The Confirmation dialog opens:



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

3. Select the check box to delete the project from the Studio workspace and the file system. If you do not select the check box, the project is removed from Studio workspace but not from the file system. If required, you can reimport the project at a later stage.
4. Click **Yes** in the confirmation dialog box.

## Move flows, operations, and configuration items

You can move flows, operations, and configuration items to a different folder within a project or to a new project.

**Note:** It is not possible to drag and drop flows and operations at the same time as configuration items. It is not possible to drag and drop a flow or operation into the **Configuration** folder, and vice versa.

1. In the **Project** pane, select the flows or operations that you want to move. You can select multiple items using the SHIFT or CONTROL key.
2. Drag and drop the items or cut and paste them into another folder or another project.

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

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

1. In the **Dependencies** 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.

**Tip:** A quick way to copy multiple items from the Dependencies pane is to select them, holding down the SHIFT or CONTROL key, and drag and drop them into the project.

**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 375](#).

## Display the project properties

- Double click on the **Properties** option in the **Projects** pane.

or:

Right-click the project and select **Properties**.

or:

Right-click the project on the **Properties** node in the **Projects** pane and select **Open**.

The Properties window displays information about the project. The project's Properties window allows you to set the **Publisher**, **Version** and the **Description** of the project as well as the dependencies of the project.

You can also perform standard SCM operations from the **Properties** node in the Projects pane (as well as from the **SCM Changes** pane. You can add, commit, revert to old versions, update and see the history of the properties.

This means that you can commit only the project metadata changes (publisher, version or description changes) without committing the entire project's changes. Similarly, you can commit dependency changes without committing the entire project's changes.

For full details on these options, see ["Working with Subversion Source Control Management " on page 62](#) or ["Working with Git Source Management System" on page 88](#).

## Show the project location in Windows explorer

1. Select the **Properties** option in the **Projects** pane.
2. Right-click and from the menu, select **Show in Explorer**.

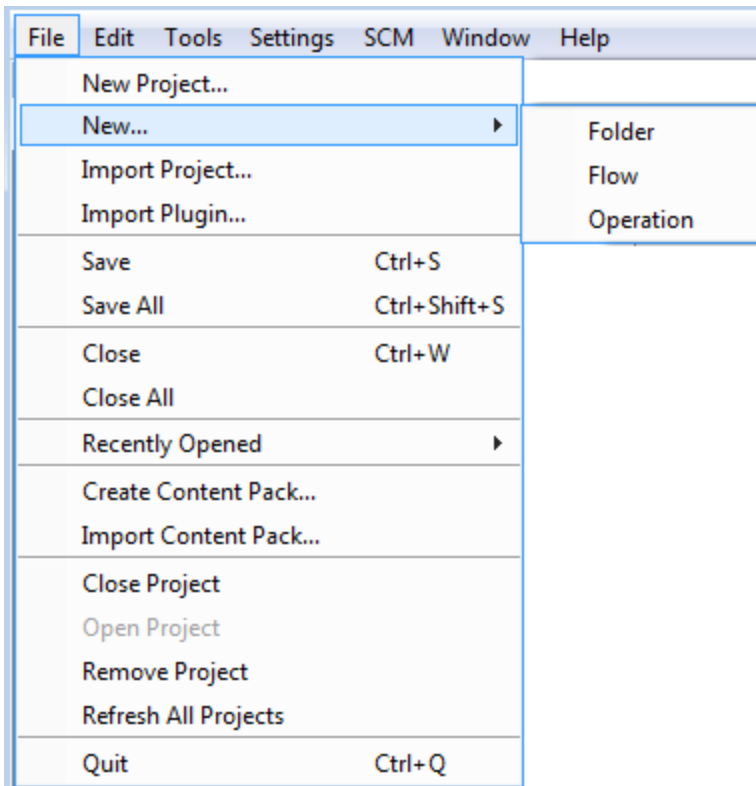
Windows Explorer opens, showing the folder containing the project files.

## 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

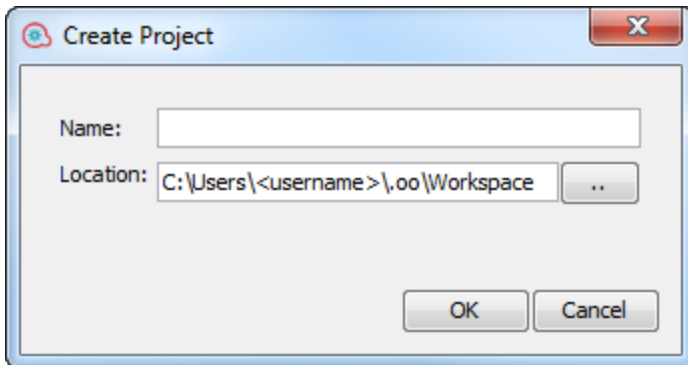
### File menu




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</a> " on page 360.
<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 Content Pack</b>	Creates a content pack from the selected project.
<b>Import Content Pack</b>	Browse to and import a content pack in .jar format. See <a href="#">"Importing Content Packs to a Project" on page 129</a> .
<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>Remove Project</b>	Removes the selected project from the workspace.
<b>Refresh All Projects</b>	Refreshes all the projects in the Projects pane.
<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>



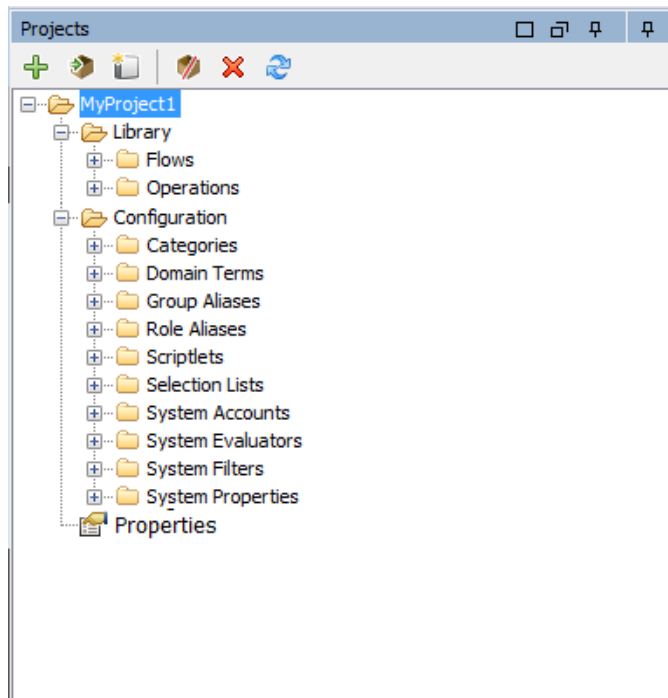
## Projects pane




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.

**Note:** You can create folders in all the configuration items under the existing Configuration Item folder structure.

- The project properties.



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.

**Important!** If you need to delete, create, or rename an element (folder, flow, operation, or configuration item) inside your project, make sure to do this from within Studio, and not by deleting, creating, or renaming the item in the file system.

## 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**.

**Important!** If you need to create an element (folder, flow, operation, or configuration item) inside your project, make sure to do this from within Studio, and not by creating the item in the file system.

**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**.

**Important!** If you need to rename an element (folder, flow, operation, or configuration item) inside your project, make sure to do this from within Studio, and not by renaming the item in the file system.

## Delete a folder

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

**Important!** If you need to delete an element (folder, flow, operation, or configuration item) inside your project, make sure to do this from within Studio, and not by deleting the item in the file system.

## 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 in HP OO Studio

This section describes the common tasks that are used with the Source Control Management (SCM) tools and more advanced tasks that you, as author, may encounter when projects and items are shared with multiple authors.

### What is Source Control in HP OO Studio?

HP OO Studio includes two built-in Source Control Management tools that allow authors to work in their local environment and then synchronize changes with the public version:

- **Subversion:** For details, see "[Working with Subversion Source Control Management](#)" on [page 62](#)
- **Git:** For details, see "[Working with Git Source Management System](#)" on [page 88](#).

**Note:** You must select only *one* of these tools as they cannot be used concurrently. If you are not sure which to use, consult with your system administrator.

### What do you want to do?

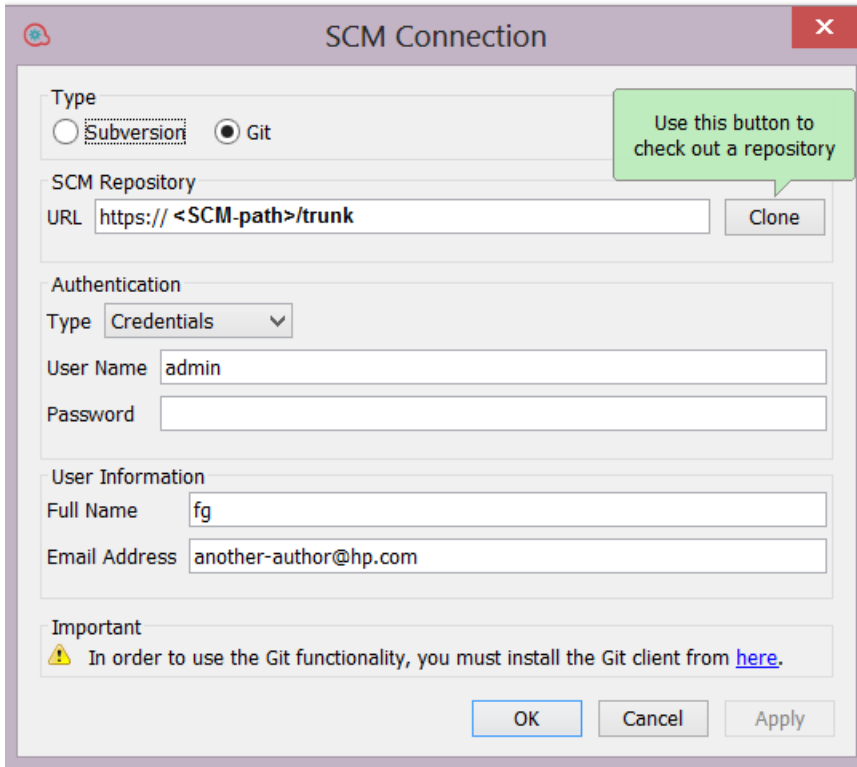
#### Select a source control tool

To select a source control tool:

1. Select **SCM > Connection**
2. In the **SCM Connection** dialog box, select the SCM type: **Subversion** or **Git**.

# Reference Material

## SCM Connection



Menu item	Description
<b>Type</b>	Select the source control management tool you want to use: <b>Subversion</b> or <b>Git</b> .
<b>SCM Repository:</b> <b>URL</b>	Enter the URL of the Subversion or Git repository.  <b>Note:</b> The HTTP and HTTPS protocols can be used for the URL.
<b>Clone / Detach</b>	<b>For Git only:</b>  Clones the main Git repository to the working copy on the local machine in the directory of the current workspace .  After cloning, the Git path is disabled, and this button changes to <b>Detach</b> .

<b>Detach</b>	<p>Detaches the main Git repository from the local directory. Use the <b>Detach</b> option if you are using more than one Git repository and want to switch to a different repository.</p> <p>After selecting <b>Detach</b>, the following message opens:</p> <p>Before detaching from Git, make sure that all your changes are committed and then pushed to the main Git repository. You can clone the Git repository again at a later stage to retrieve your changes.</p> <p><b>Note:</b> After detaching, you can no longer run any Git commands.</p>
<b>Checkout / Detach</b>	<p><b>For Subversion only:</b></p> <p>Copies sources from the SVN repository to the working copy on the local machine in the directory of the current workspace.</p> <p>After checking out, the SVN path is disabled and this button changes to <b>Detach</b>.</p>
<b>Trust server certificate</b>	<p>If you are working with a secure server (SSL/SSH), select this check box, otherwise you will be unable to access the server.</p>
<b>Credentials:</b> <b>Use Windows Authentication</b>	<p>When checked, performs authentication using the currently logged in user.</p> <p>To perform authentication with a different user, uncheck this field, and enter the user name and password in the fields below.</p>
<b>User name</b>	<p>Enter the user name and password to use as credentials for authentication.</p>
<b>Password</b>	

## Working with Subversion Source Control Management

### Terminology

#### 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.

**Note:** The Source Control Management tool does not include historical data for flows, such as date, time, and comment.

## **Creating an Initial Source Control Repository**

In a production environment it is recommended to use a dedicated SVN server, accessed by the protocols http, https or svn.

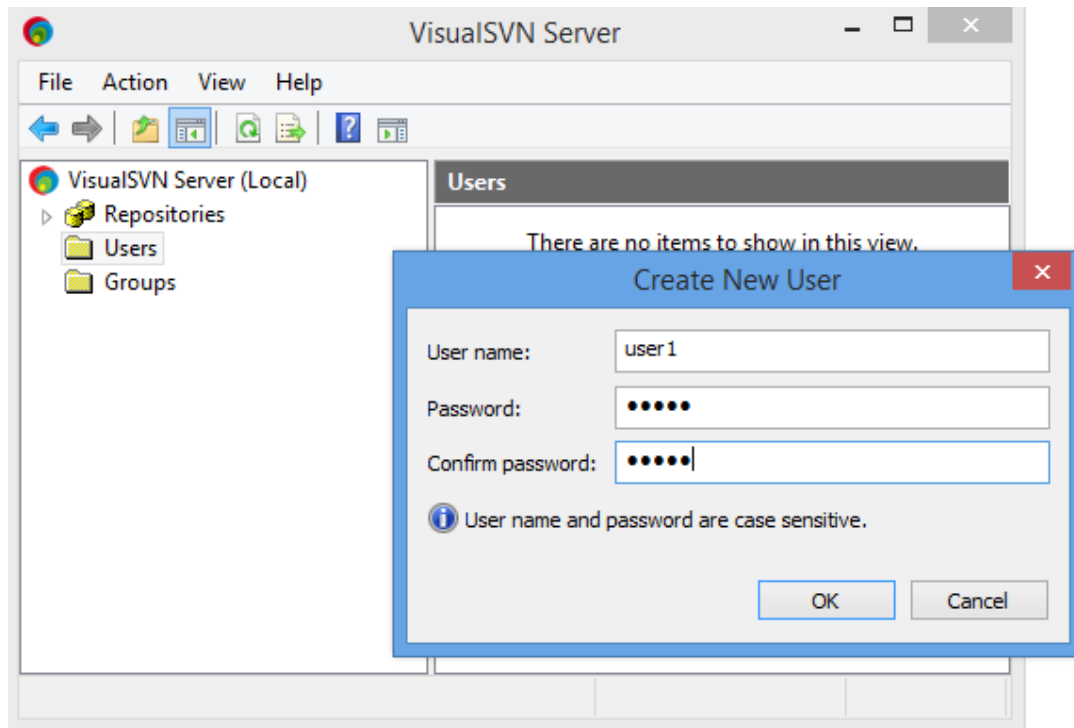
**Note:** Go to the subversion web site, <http://subversion.apache.org/packages.html>, for a list of SVN distributions for different operation systems.

### **Example 1: Using VisualSVN in a Multi-Author Scenario**

The following example illustrates how to configure and use a VisualSVN server in a multi-author scenario.

Before beginning to work with SVN, it is recommended that you get accustomed with its concepts from the following link: <https://subversion.apache.org/docs/>.

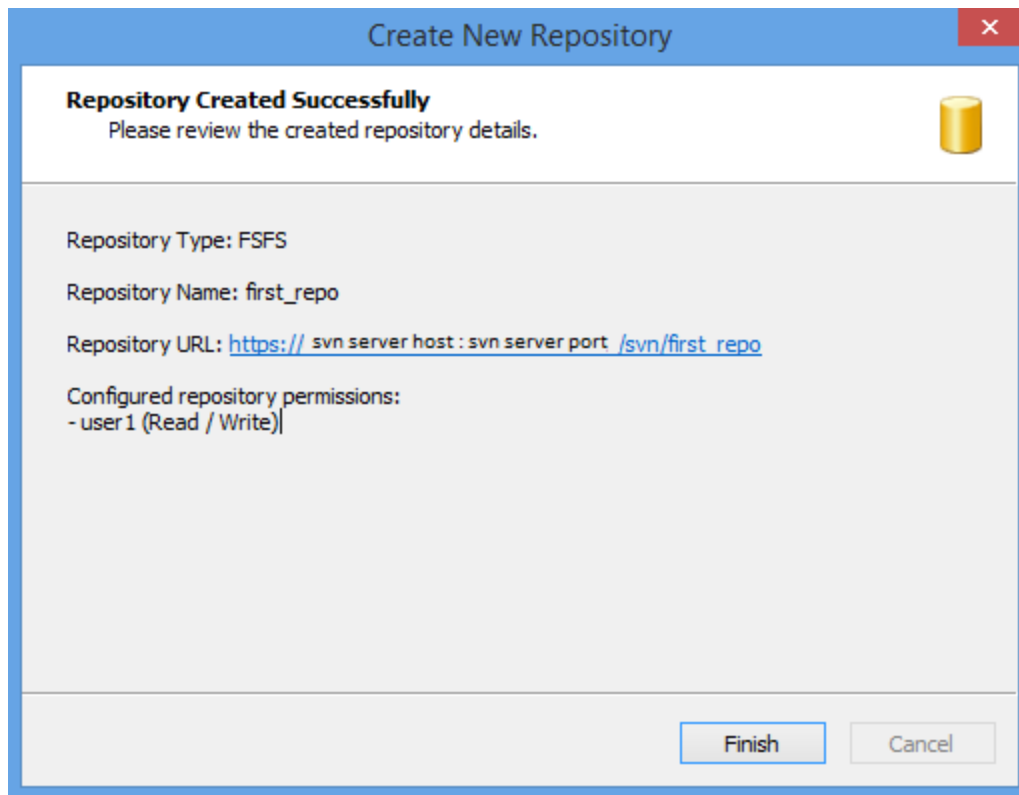
1. Download a 1.8.x SVN compatible version of VisualSVN.
2. To configure the new SVN server users, right-click and select **Users > Create User....**



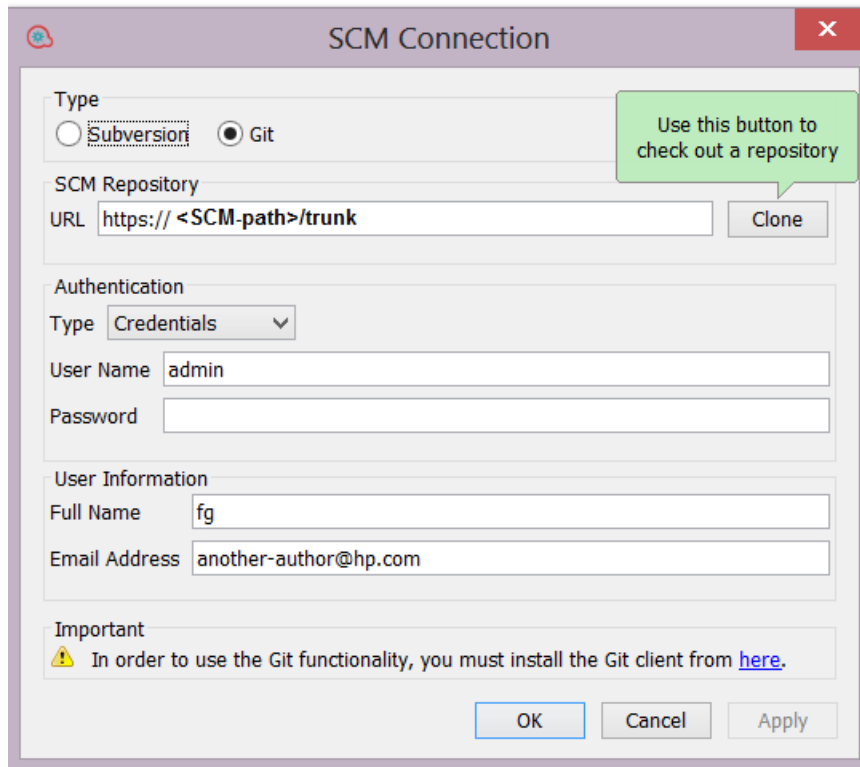
**Note:** Every Studio persona is identified by a Visual SVN user.

3. To create a SVN repository, right-click on the Repository folder, select **Create New Repository** and create an empty repository.





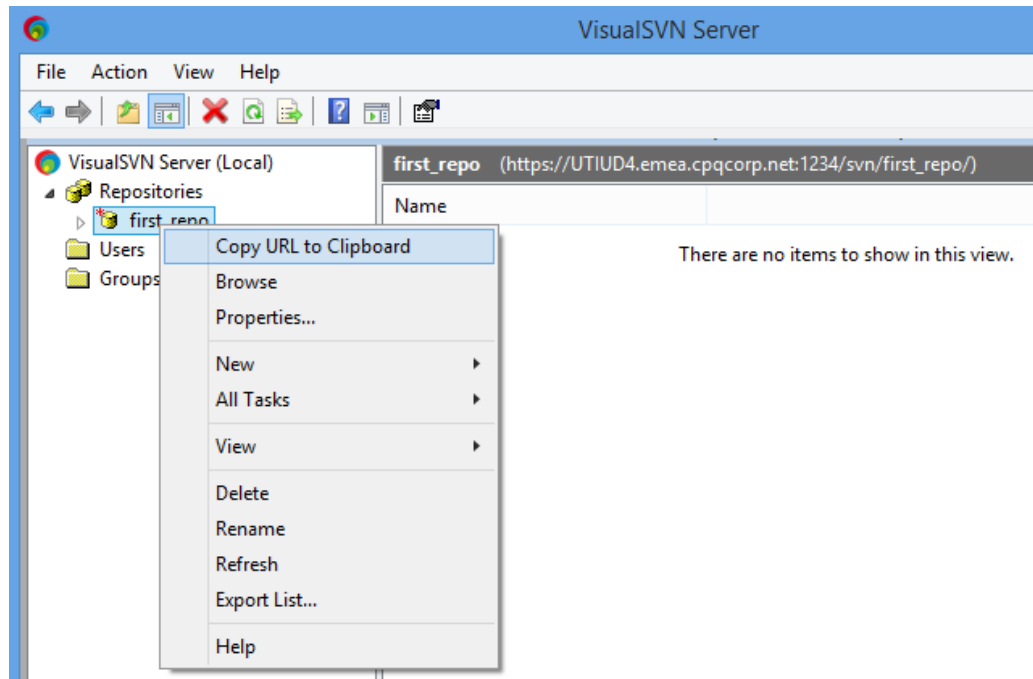
4. In Studio, from the **SCM** menu, select **Settings**.
5. Select the **Subversion** radio button.
6. Enter the SVN user name credentials - **User Name** and **Password**.



7. For an HTTPS protocol, select the **Trust server certificate** check box.
8. Check out the repository.

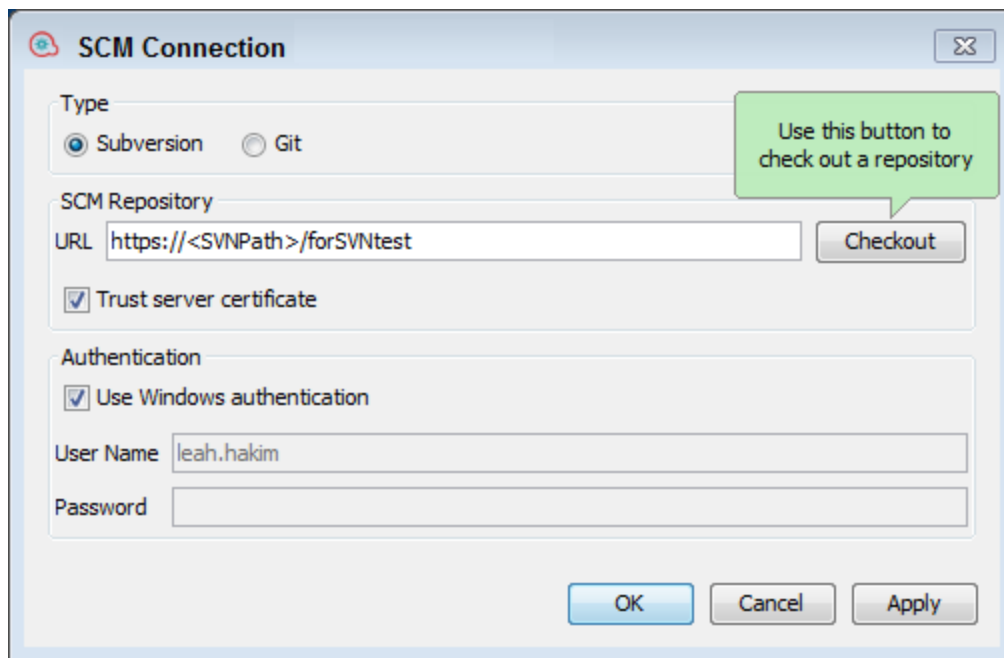
If you have direct access to the VisualSVN console application you can obtain the repository URL as follows:

- a. Right-click on the repository location in VisualSVN.
- b. Select **Copy URL to Clipboard**.



If you do not have access to a VisualSVN console application, ask your SVN administrator to provide you with the repository URL.

9. In Studio, from the **SCM** menu, select **Connection**.



10. In the **SCM Repository** area, **URL** field, paste the URL using Ctrl-V and click **Checkout**.

A Checkout message appears in the SCM Messages pane.

## Example 2: Single-User Scenarios

The following example is only for single-user scenarios, testing, and for debugging purposes.

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

**Note:** It is recommended not to use a file system-based (or network share-based) repository in production.

This repository is intended for local, single-user access only, particularly testing and debugging, as when shared over the network, there are security concerns and possible performance problems.

**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\tools`.
3. In the command window, type: **manageScmRepository.bat** and the full path to the shared folder.

For example:

```
c:\<studio_installation_folder>\Studio> manageScmRepository.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 Multiple Authors in SVN

When multiple authors work on 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.

A flow is automatically locked if the author tries to delete, move, rename or revert a flow. Once the flow is locked, other authors are prompted that the flow is locked when they try to perform one of these actions. This occurs when the **Enforce locking policy** is enabled. See ["Enforce Locking Policy" on the next page](#) for more information.

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

You cannot lock items that do not exist in the SCM repository.

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.

**Note:** A lock is associated to a workspace and not to a particular SCM user. Therefore, if you have locked items and your workspace is destroyed, after recreating the workspace, you will not be able to continue working on your previously locked items nor will you be able to unlock them. In this case, you will need to either contact your SVN administrator, or to use an external SVN tool to break the locks on your items.

## Enforce Locking Policy

This options prevents the flow author from making any changes to an item (flow or configuration item) unless the item is locked. This ensures that only one author can edit an item.

This option is automatically selected once the author checks out an SCM repository.

When the items is locked, the author can choose to unlock the item manually.

The author will then be prompted to select one of the following options after making changes to the item and unlocking it:

- Commit your changes.
- Unlock the item and keep editing.
- Revert the item.

In the following cases the author cannot obtain a lock:

- Another author has already locked the item.
- Network issues.
- The item version is out-of-date.

In the above cases, the **Failed to Acquire Lock** message appears prompting you if there are any local changes. Click **Yes** to continue editing, or **No** to discard the changes.

When **Enforce Locking Policy** is enabled, Studio automatically obtains locks on the affected items of all operations that change the project's structure. For example, renaming, moving or deleting flows, operations or folders.

## What do you want to do?

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.

### Set the authentication settings with the 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 **Connection**.

The screenshot shows the 'SCM Connection' dialog box. The 'Type' section has 'Subversion' selected. The 'SCM Repository' section has a URL field with the text 'https:// <SCM-path>/trunk' and a 'Clone' button. The 'Authentication' section has a 'Type' dropdown set to 'Credentials', a 'User Name' field with 'admin', and a 'Password' field. The 'User Information' section has a 'Full Name' field with 'fg' and an 'Email Address' field with 'another-author@hp.com'. The 'Important' section has a warning icon and text: 'In order to use the Git functionality, you must install the Git client from here.' At the bottom are 'OK', 'Cancel', and 'Apply' buttons. A green callout box points to the 'Clone' button with the text 'Use this button to check out a repository'.

2. Under **Type**, select the **Subversion** radio button.
3. 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.
  - Clear, the **Use Windows Authentication** option. Enter the user name 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.

**Note:** The credentials are only used to access the SCM itself. If the system is based on a simple network fileshare, 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 operating system for a network connection.

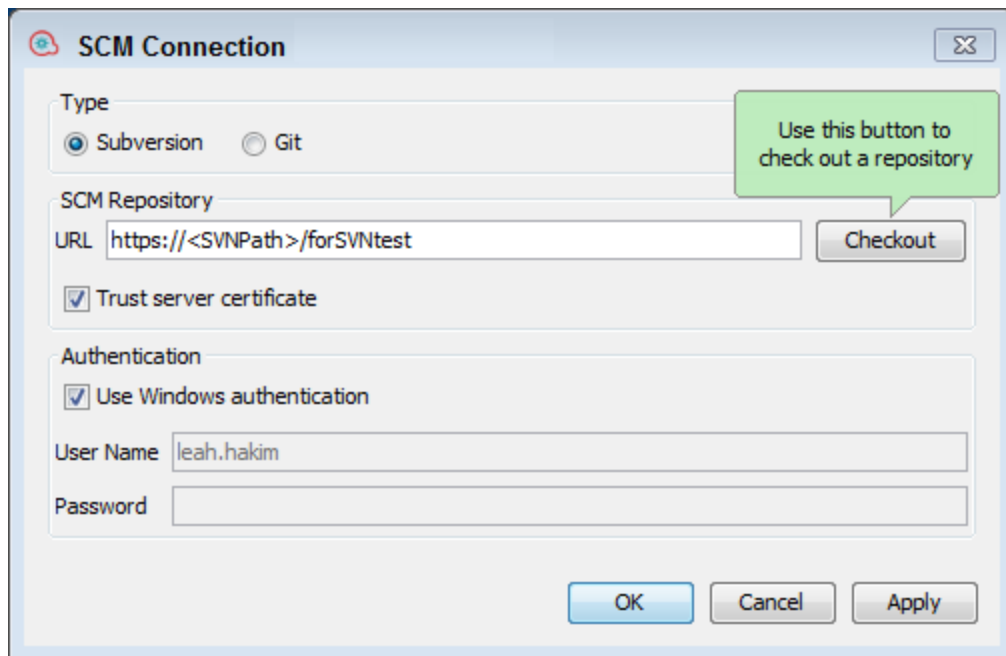
## Check out a repository

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.

You can check out the repository directly from Studio and then add projects to it for sharing with other authors.

To check out all projects:

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



2. In the **SCM Repository** area, **URL** field, paste the URL using Ctrl-V and click **Checkout**. This checks out the files from the URL and into Studio's Workspace directory.

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:



- Web-based URL:

`http://svn.samplehost.com/repo/trunk`

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



`file:///c:/temp/repo/trunk`

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

`file://myshared/repo/trunk`

A Checkout message appears in the SCM Messages pane. The **SCM Messages** pane 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.

**Note:** In the **SCM Messages** pane, the messages coming from the Source Control Management client tool may contain an encoded URL. This is the repository URL in a standard encoded form. The message is coming from an external SVN client tool used by Studio (SlikSVN). Note that a non-encoded version of the URL is also logged in the **SCM Messages** pane.

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

## Detach the Workspace from SVN

After detaching 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 **Connection**.
2. In the SCM Repository area, click **Detach**.
3. Click **OK** to close the window.

**Note:** Detaching from an SVN repository will unlock all the flows, operations and configuration items which were already locked by the user.

## Commit changes to SVN

After making local changes, use the **Commit** menu option to check them into the repository. There are a number of ways to commit changes:

- From the **SCM** menu, select **Commit All** to commit all the changes.
- In the **SCM Changes** pane, click the **Commit** button to commit all the changes.
- In the **Projects** pane, right-click an updated item and select **SCM > Commit**.
- In the **SCM Changes** pane, right-click an updated item and select **SCM > Commit**.

## View a history of SVN operations in the repository

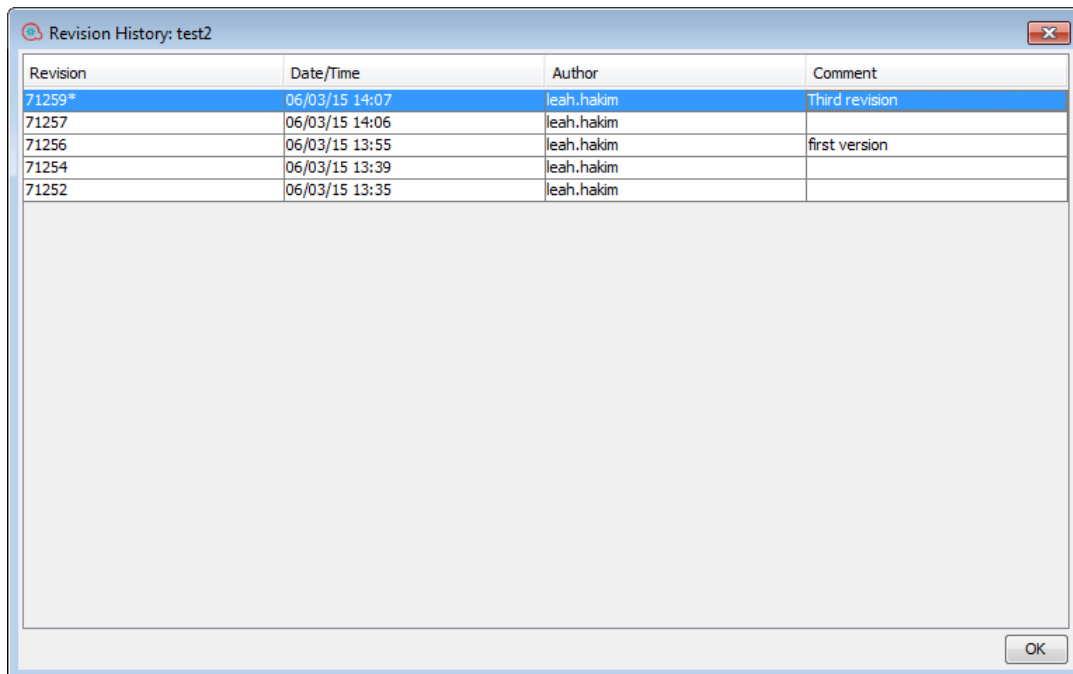
The **History** option shows you:

- A list of commits in which the selected project/item was affected
- All affected files for a particular commit

The **History** option also allows you to revert to a specific revision of the project/item. See "[Revert to a previous revision of a project/item](#)" below for details.

1. Right-click on a file/folder and select **SCM > History**.

The Revision History dialog box opens:



## Revert to a previous revision of a project/item

To revert to a previous revision:

1. Right-click on a file/folder and select **SCM > Revert**.

The Revision History dialog box opens.

2. Right-click on the commit revision you want to revert to.
3. Select **Revert to this version**.

**Note:** You cannot revert to a revision number earlier than the initial revision given when the project was first committed to SVN.

### Update to a tagged version or revision of a project/item

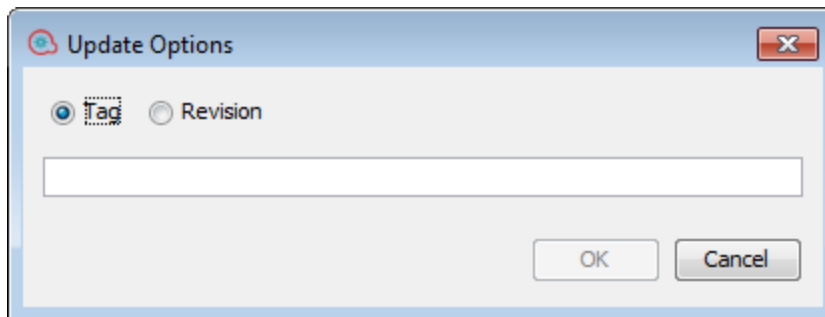
There may be cases for which you need to recreate a faulty Central content pack/project in order to troubleshoot a specific flow inside the content pack. If the content pack is not available (it has been lost or been overwritten by a new version), you can restore a specific version of the project/item using the **Update to...** operation.

**Note:** In SVN, you can perform a **Update to...** operation on individual folders and repositories.

You can use the **Update to...** option if you have already checked out the repository and you want to update your workspace to a specific revision from the SCM tag.

1. Select the content pack/project you want to update.
2. Right-click and select **SCM > Update To....**

The Update Options window opens:



3. Select **Tag** and type in the SCM tag you received from the Central administrator. By default, the SCM tag is in the format [Project Name]-[Project version].

or:

Select **Revision** and type in the revision number you received from the Central administrator.

4. Click **OK**.

## Move/cross-move a versioned item

You can move an item from one location to another within the same project or cross-move an item between two different projects.

To move an item:

1. In the Project pane, select the items that you want to move. You can select multiple items using the **Shift** or **Ctrl** key.
2. Drag and drop the items into another folder or another project.

*or:*

Cut the items using **Ctrl+X** and paste them at the new location using **Ctrl+V**.

The item is marked in the SCM Changes pane in the old location with the status **deleted** and in the new location with a status **moved**. You can see the original location in parentheses.

### Note:

- When moving a versioned item, the item must also be scheduled for moving in the SVN repository.
- When cross-moving an item (between a source project and a destination project), the item must be scheduled for moving, deletion or addition in the appropriate projects.
- Where relevant you can commit the result of the move/cross-move action to SVN.

## Clean up the working copy

In some cases, you will need to clean up the working copy in Studio's workspace. For example, if a Studio process crashes or if there is an IO error, and the working copy remains locked.

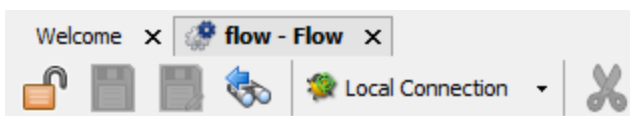
If you need to clean up the working copy, you will see an error message.

To clean up the workspace:

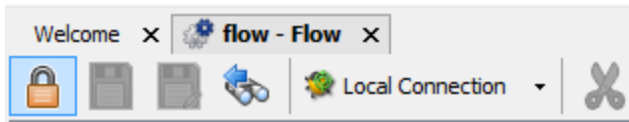
- From the **SCM** menu, select **Subversion > Cleanup**.

## Lock an Item

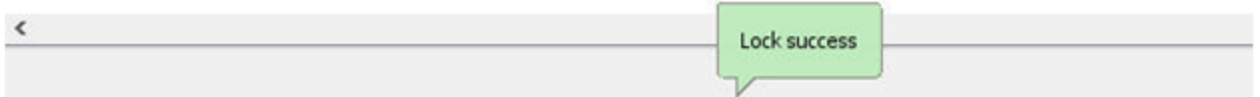
- In the top left corner of the items editor window, click the **Opened Lock** icon.



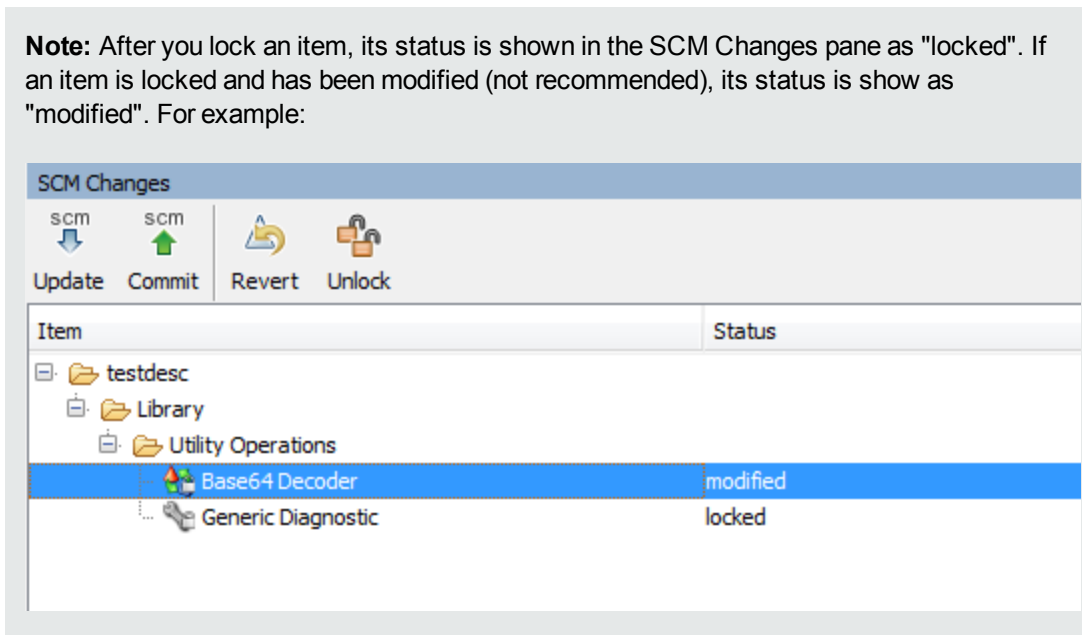
If the lock was successful, the icon changes to a closed lock and displays an SCM message displaying the full path to the locked file and with the user details.



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

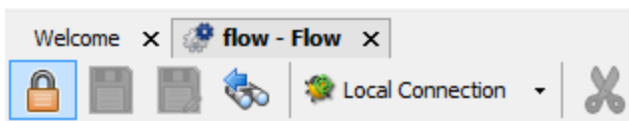


**Note:** After you lock an item, its status is shown in the SCM Changes pane as "locked". If an item is locked and has been modified (not recommended), its status is show as "modified". For example:

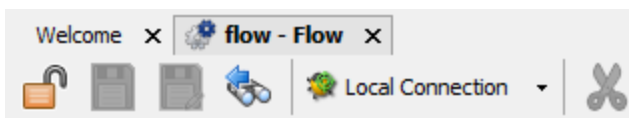


## Unlock an Item

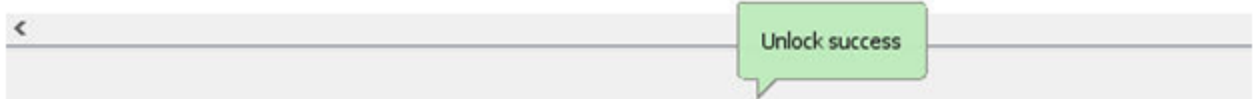
- In the top left corner of the items editor window, click the **Closed Lock** icon.



If the unlock was successful, the icon changes to an Open Lock and displays an SCM message displaying the full path to the unlocked file and with the user details.



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



## Set the Enforce Locking Policy

- From the SCM menu, select **Subversion > Enforce Locking Policy**.

## Reference Material: SCM Changes Pane

The SCM Changes pane 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 **SCM Changes** pane. This pane also shows a list of deleted projects (projects marked for deletion), if such projects exist.

**Note:** Projects that are outside of Studio workspace are not added/committed to SCM. They are shown as unversioned (in brown) in the Projects Pane, and are not shown in the SCM Changes Pane.

## Types of changes

You can see how different types of changes are shown:

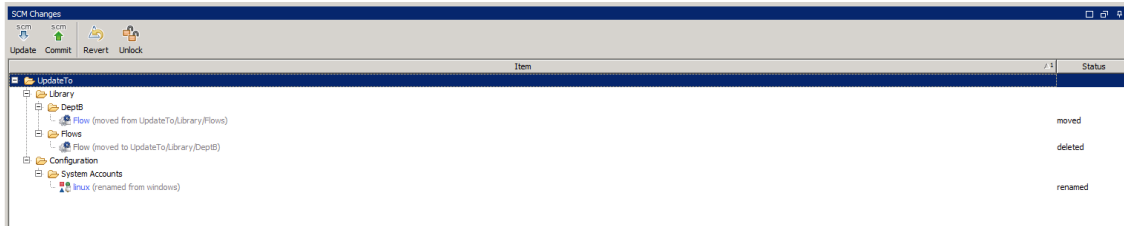


- Deleted flow, named **flow**, and colored gray.
- Added folder, named **deprecated**, colored green
- Unversioned project, named **Project2**, colored brown.
- One deleted project.

- 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, **windows** was renamed **linux**, and the folder **flow** was moved.



### SCM Changes toolbar

	Update all: Updates the entire Studio workspace.
	Commit all changes: Commits all the changes that show in the <b>SCM Changes</b> pane. Only available when there are changes.
	Revert all changes: Reverts all changes that show in the <b>SCM Changes</b> pane. 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 pane)
- Green: Added
- Grey: Deleted (not available in the projects tree)
- Blue: Modified

- Brown: Unversioned
- Zig-zag underscore: Includes errors.

**Note:** By default, SVN has a list of file patterns (shown below) that are ignored when non-versioned files are added to SCM:

```
*.o *.lo *.la *.al .libs *.so *.so.[0-9]* *.a *.pyc *.pyo
```

This means that a non-versioned project or folder that matches one of these patterns is shown as black (as if it is versioned) in the SCM Changes panel.

However, you can still add the project or folder to SCM and work with them as usual.

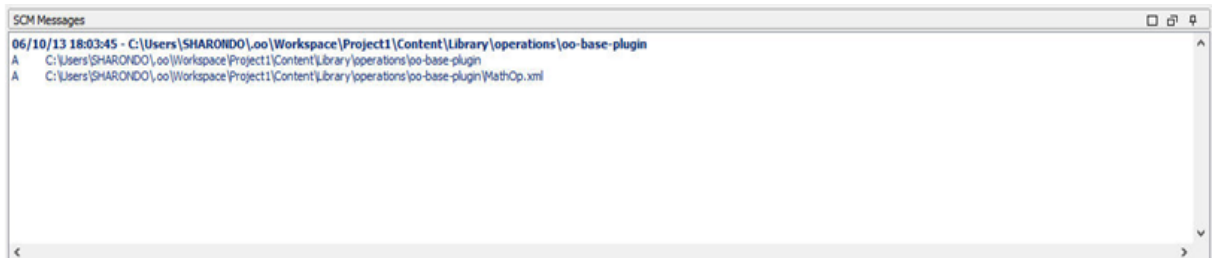
## SCM Message Pane

The SCM Message Pane displays the message results from the SCM actions. Every action results in a message and is color-coded. In addition, a pop-up message appears informing the user of the result of the SCM action. Clear the **SCM Message** pane by right clicking anywhere inside the pane and then clicking the **Clear All** button.

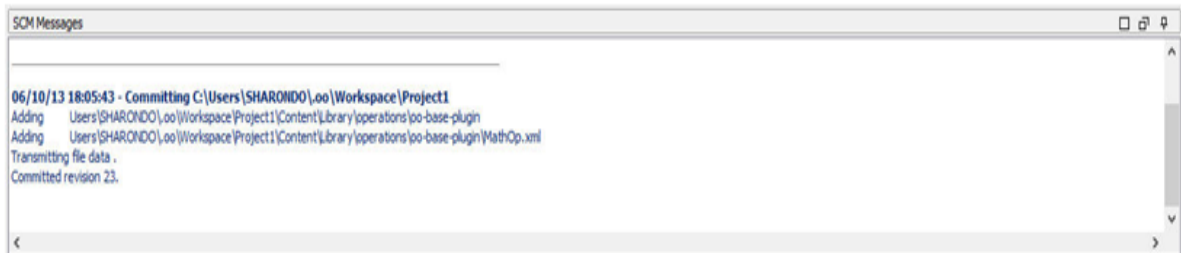
### Examples of SCM Messages

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.

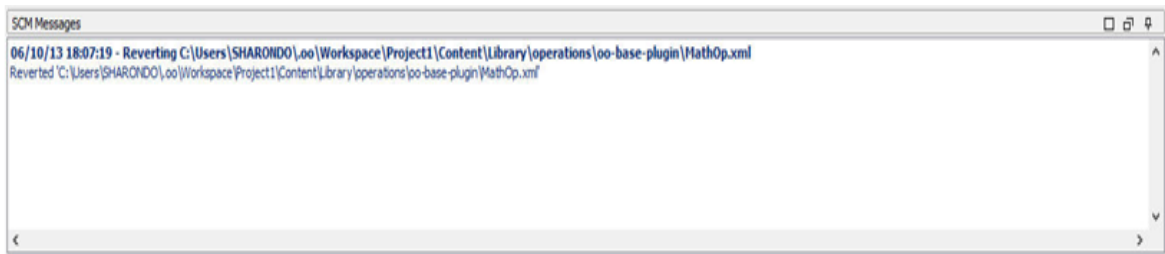


After you commit an item, the following message appears. This message shows that your previously added items were now successfully committed to the server.



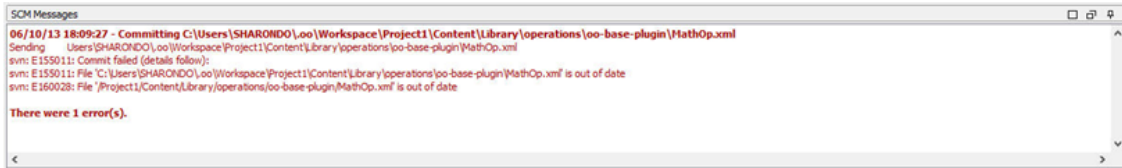
After making a change to **MathOp**, for example, adding a new input, revert the changes. This message shows that a change to MathOp.xml was reverted.





### Error messages

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



## Reference Material

### Revision History

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

The screenshot shows a window titled "Revision History: test2" containing a table with the following data:

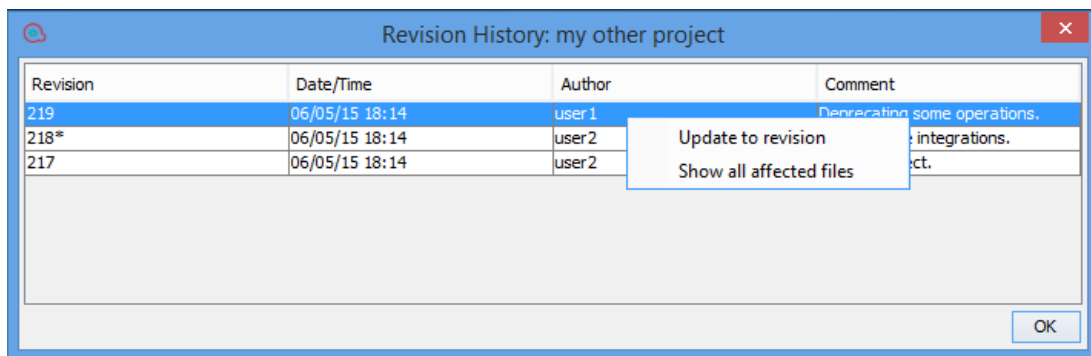
Revision	Date/Time	Author	Comment
71259*	06/03/15 14:07	leah.hakim	Third revision
71257	06/03/15 14:06	leah.hakim	
71256	06/03/15 13:55	leah.hakim	first version
71254	06/03/15 13:39	leah.hakim	
71252	06/03/15 13:35	leah.hakim	

An "OK" button is located at the bottom right of the window.

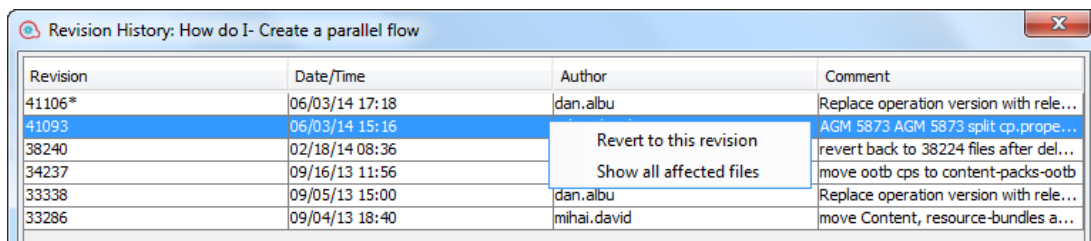
- **Revision:** The revision for the commit. The revision of the local copy is marked with an asterisk. After checkout, the asterisk always marks the latest revision, but after changes have been committed by another user, and before updating the asterisk marks a revision that is not the latest. The screen shot above shows that the local copy is at revision 1953, but there was a commit to the file in revision 402.
- **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 (\*).



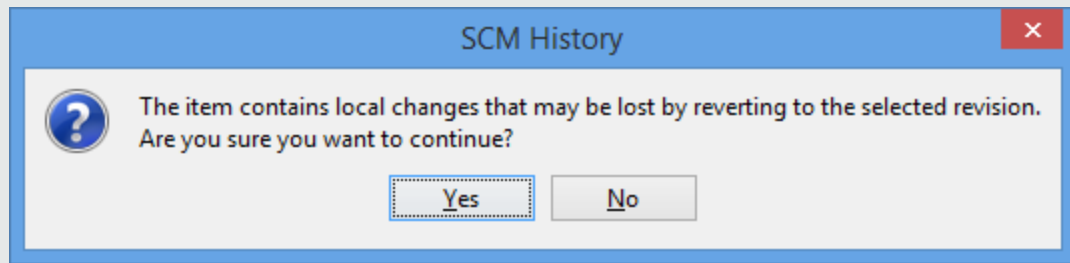
- **Revert to this revision:** Locally changes the object to the way it was in the selected revision. If for example a flow contains two steps in revision 333, and three steps in revision 337 then reverting it to revision 333 causes it to have two steps again. In order to make the changes visible to other authors, you must commit the flow. 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:** When running a **Revert to this revision** operation on a folder with changed items or on a changed flow/operation/configuration item, the changes can be lost and there could be tree conflicts that are resolved using the local version. In these cases, the following message

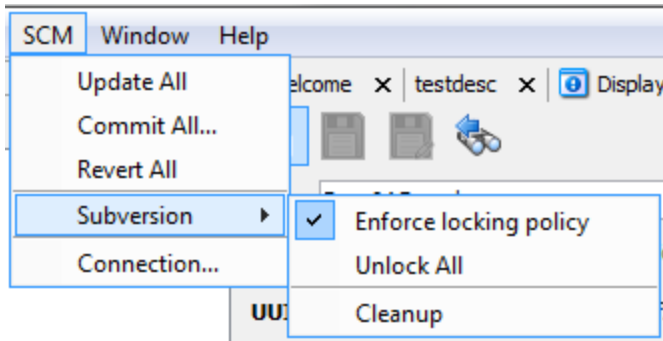
appears:



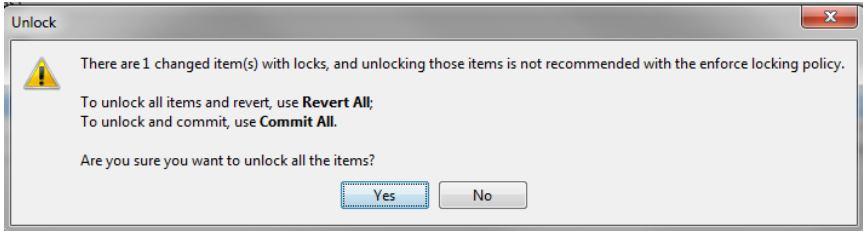
It is recommended to revert the changes first, and then to try to run **Revert to this revision** again.

## 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.

Option	Description
<b>Subversion</b> <b>&gt; Enforce Locking Policy</b>	This options prevents the flow author from making any changes to an item (flow or configuration item) unless the item is locked. This ensures that only one author can edit an item. See <a href="#">Enforce Locking Policy</a> above for more information.
<b>Subversion</b> <b>&gt; Unlock all</b>	Unlocks any locked items in the workspace. A warning message appears. For example: <div data-bbox="522 657 1380 886" style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  </div> Select <b>Yes</b> to unlock all the items or use the other options presented in the message.
<b>Subversion</b> <b>&gt; Cleanup</b>	Clean up the working copy in Studio's workspace.
<b>Connection</b>	Allows you to change the way authentication is performed with the source control server, to select the SCM type and to check out/detach a repository. <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 PasswordAuthentication:</b> Performs authentication using the supplied user name 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>

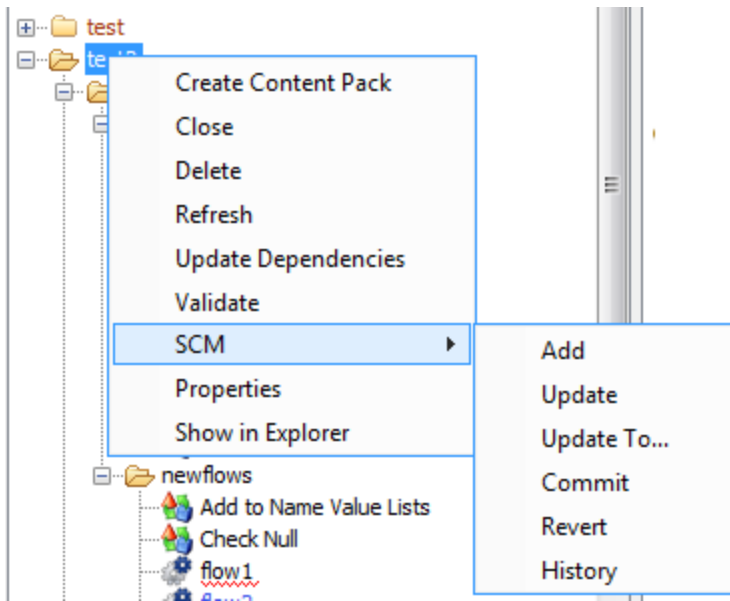
## Projects pane

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 pane

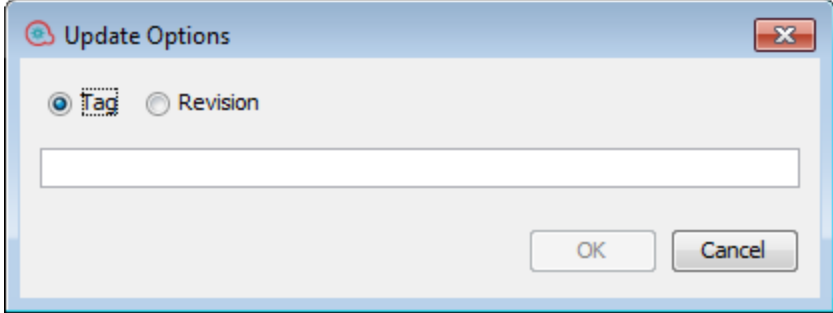
The context menu shows all actions that are available for the selected object, according to its state.

Following are the options available when you right-click on a flow:



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 you commit 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.

Option	Description
<b>Update to...</b>	<p>Updates the selected item to a specific version. When this option is selected, the Update Options window opens:</p>  <p>Enter the tag name (from the deployment page in Central) or the revision number (from the Revision History window) and click OK.</p>
<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 you add an item it is automatically marked as added, but when you revert changes on it only the addition mark is removed, but the item still exists in Studio and in the file system. You can re-add the item using <b>Add</b> in the menu, or delete it.</p>
<b>History</b>	<p>The <b>Revision History</b> window, displays the SCM history. See <a href="#">Revision History</a> for more information.</p>

## Deleting a Project

- Select the project that you want to delete, and click the **Delete** button. When you 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 SVN

- **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 Git Source Management System

Git is a Source Control Management (SCM) system that provides you with an alternative solution to source control, in parallel with the Subversion – SVN solution. Git is a distributed version-control system, with certain concepts that differ from Subversion.

The methodology of working with Git in Studio is very similar to the methodology of SVN.

The initial version of Git support for 10.50 targets the basic use cases and provides ease-of-use, similar to the Git support integrated to other IDE tools (such as IntelliJ IDEA or Eclipse).

### Git Terminology

#### **Add: Add a new file to a Git repository**

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

#### **Clone**

Clone is used to download sources from the Git repository to the working copy. If you want to access files from the source control server, clone is the first operation you should perform. When you clone a repository, a working copy is created, allowing you to edit, delete, or add contents. To clone, you need the source control server URL of the Git repository you want to clone.

#### **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, you must commit the changes. The changes are saved in the local Git repository. You can then push them to the remote Git repository.

#### **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 to the remote repository.

#### **Downstream/Upstream**

In terms of source control, you're downstream when you copy (clone, checkout, etc) from a repository. Information flowed "downstream" to you.

When you make changes, you usually want to send them back "upstream" so they make it into that repository so that everyone pulling from the same source is working with all the same changes. This is mostly a social issue of how everyone can coordinate their work rather than a technical requirement of source control. You want to get your changes into the main project so you're not tracking divergent lines of development.

#### **Ignore**

There are files that do not need to be version-controlled in Git. These files are handled by the Git Ignore mechanism that operates behind the scenes, and is automatically executed when adding a project. By default, the files to ignore are (relative to the project home): \*.idx, \*.tmp, \*.lock, and Content/.metadata.



## Merge

The **Merge** command incorporates changes from the commits (since the time their histories diverged from the current branch) into the current branch.

## Move

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

## Pull

Pull refers to when you are fetching in changes and merging them. For example, if someone has edited the remote file you're both working on, you'll want to pull in those changes to your local copy so that it is up to date.

## Push

Pushing refers to sending your committed changes from the local repository to a remote repository. For instance, if you change something locally, you'd want to then push those changes so that others may access them.

## Rebase

Rebase take all the changes that were committed on one branch and replays them on another branch.

## Stash

Stash saves your working directory and index to a safe place and restores your working directory and index to the most recent commit. After stashing, you can then work on other branches, make commits, etc. and when you're ready to get back to where you were, run a **git stash apply** (unstash) operation to restore your working directory.

## 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.

## Update: Update the working copy

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

**Note:** Some of the terminology in this glossary is taken from the github site <https://help.github.com/articles/github-glossary/>.

For more information on Git, go to one of these sites: <https://help.github.com/> or <http://git-scm.com/>.

## Getting Started with Git in Studio

To begin working with Git, you must connect to the remote Git repository and clone it. This way, you have a copy of the repository that you can work on, and then commit changes to the local repository and push your changes to the remote Git repository.

Have a look at the following movies to get a better understanding on how to work with Git in Studio.

Movie	What you will learn
<a href="#">Connecting to Git and cloning a Git repository</a>	In this movie, you will be introduced to some basic Git operations. You will learn how to clone a Git repository, commit changes to the local repository and push them to the remote repository.
<a href="#">Working with Git – Basic Operations</a>	In this movie, you can see how to use a different type of authentication (SSH) to clone the Git repository, learn how to stash local changes, revert to an earlier committed version and unstash.
<a href="#">Conflict Handling</a>	In this movie, you will see how to handle conflicts that may occur in Git when two authors make changes to the same Studio workspace and try to commit and push to Git.
<a href="#">Git branching functionality</a>	In this movie, you will learn how to work with different branches in Git.
<a href="#">Multi-authoring</a>	In this movie, you will learn about advanced multi-authoring capabilities in Git.

## Git Branch support

Branching is one of the strong points of Git, one that is a differentiator when compared to Subversion. One most popular branching model is explained on this blog: <http://nvie.com/posts/a-successful-git-branching-model/>.

Branch management is an important part of the Git workflow. Branches make it easier to track changes and collaborate with other people. You can manage branches in the Git repository directly from Studio. You can see more information on Git branching here:

<https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow/>

For details on creating a new branch, see "Create a new branch in the Git repository" below.

## Understanding the Git Repository Log

The **Git Repository Log** shows a graphic representation of the latest commits and updates that have been made to the remote Git repository. This functionality is similar to the logs provided in other Git tools.

The repository log shows the following information:

- Important pointers (such as master, HEAD, origin/master, branches, tags etc.), that allow you to identify the branch from which the commit was done type of commit and whether or not it was pushed to a remote branch
- A commit graph showing the relationship between the different commits/updates (based on their ancestry) and showing how merges between the commits/updates were done
- The commit message, as entered in the Commit window
- The author of the commit (name and email address)
- The date and time of the commit (in chronological order)

Each node in the graph represents a commit/update operation. Depending on the update strategy selected (see "[Git Update Strategies](#)" below), the path will branch off into a new branch.

## Git Update Strategies

When updating files from the remote Git repository, you can choose the update strategy you want to use. Depending on the update strategy selected, the path will branch off into a new branch.

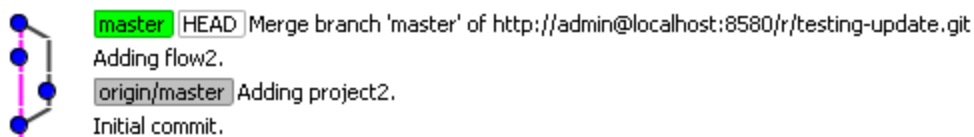
- **Update strategy: Branch Default** The default update strategy (which is always either merge or rebase) is configurable by the Git repository configuration using one of the following Git keys: **pull.rebase** and **branch.<name>.rebase**. For more information, see <http://git-scm.com/docs/git-config>.
- **Update strategy: Merge** When using the update strategy merge, Git preserves the original hierarchy of the local commits, and creates a special "merge" commit that merges the two diverged "branches" (the local branch and the remote branch) back together.

**Example:** Consider a simple example of a repository that has the following graph before updating.



In this case, the local repository and the remote repository have diverged. The local repository contains commits that the remote repository does not include (the commit with the message **Adding flow2**), and the remote repository contains commits that the local repository does not include (the commit with the message **Adding project2**).

After merging the two commits, a special merge commit **Merge branch 'master' of ....** is created:



If conflicts occur during the merge operation, the merge commit is not created automatically. The merging stops, and you must resolve the conflicts in order to complete the merge process.

In this state, all the remote changes are shown as local changes, visible in the SCM Changes pane. It is your responsibility to resolve the conflicts, review all the changes, and conclude the merging by running a **Commit All Changes** operation from the toolbar in the SCM Changes pane. See ["Resolve conflicts during an Update operation" on page 102](#) for details.

If there are multiple conflicts from multiple local commits, all the conflicts are resolved at once, at the same time.

- **Update strategy: Rebase** Rebasing is an alternative method of integrating changes that is sometimes preferable over merge. Using merge makes the repository history grow horizontally (as information of all the separate “branches” are kept) and it inserts new merge commits, making it grow vertically as well. This makes it difficult to get an overall picture of the changes made to a repository.

Rebasing changes all the local commits to appear as if they were based on the most recent changes from the remote repository, making the repository log graph appear as a straight line. Rebasing is an iterative operation: first it rebases the oldest (least recent) local commit on top of the latest from the remote repository, then it rebases the second commit on top of the previously rebased commit, and so on, until it rebases all the local commits.

If there are conflicts during the rebase of a certain commit, rebasing stops and you can select to resolve the conflicts and continue rebasing, or to skip the commit (discarding all its changes). In any event, you can choose to abort the rebase operation and return to the initial state, with all previously rebased commits disappearing.

An important implication of rebasing is that you may need to resolve conflicts multiple times (when rebasing different commits), possibly on the same items.

#### **Example:**

Both repositories (local and remote) are identical until the author commits **Introducing Utility Operations**. Then, the author of the local repository adds the operations **Base64 coders**, **UUID Generator**, **Remote Command Execution** and **Groovy Scripts**.

In the meantime, another author pushes his commits to the remote repository, adding **Search and Replace**, his own version of the **Remote Command Execution** and **Remote File Transfer** operations. The **Remote Command Execution** operation results in a conflict as it was added by both authors with different content.

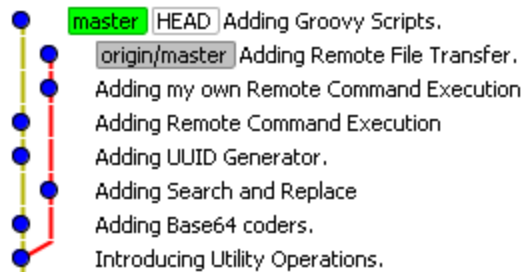
After running an **Update All** operation with rebase selected as the Update strategy, the following events occur:

- Studio displays the Conflicts dialog box with the following message: **“Merge of the current commit: 4d10a29 detected conflicts. Resolve them before continuing the rebase.”** The item in conflict is a flow named **Remote Command Execution**.

**Resolve manually** is chosen for the item.

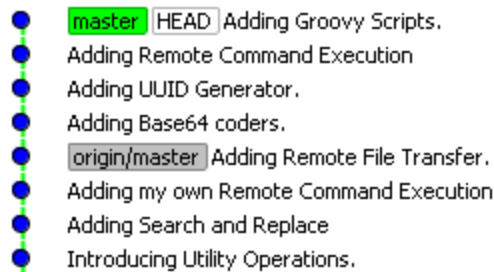
- The SCM Messages pane shows that two commits were successfully rebased (**Adding UUID Generator** and **Adding Base64 coders**), rebasing stopped due to conflicts, and the current commit is **Adding Remote Command Execution**.

At this stage, the repository log looks as follows:



The commits **Adding Base64 coders** and **Adding UUID Generator** appear twice: first, at their original location based on **Introducing Utility Operations**, and second, at the top of the list based on the latest commit from origin/master: **Adding Remote File Transfer**. Note that the current branch (“master”) is still intact. If the author chooses to abort rebasing, the workspace will go back to the master branch.

Let us assume that the user resolves the conflict and chooses to continue rebasing. At this point, then, the SCM Messages explains that it further rebased two commits (**Adding Remote Command Execution** and **Adding Groovy Scripts**), and rebasing completed successfully. The Repository log looks like this:



Notice that after rebasing, the graph the log is a straight line. After rebasing, the author can push the newly rebased commits to the remote repository.


**Note:** When there are conflicting changes on moved or renamed items (during Update), Studio automatically detects these and tries to resolve them automatically.

For example, user1 moves a flow and makes some changes to it, and user2 makes some non-conflicting changes on the same flow. Studio automatically detects this situation, and tries to merge the changes to the flow.

See "[Update files from the remote Git repository](#)" on page 100 for details on how to perform a Git Update operation.

## When is the Git Repository Log refreshed?

Studio automatically updates the Git Repository Log when:

- Studio starts up (and is connected to Git)
- After checking out a Git repository
- After committing files to the remote Git repository
- After updating files from the remote Git repository
- After detaching from the remote Git repository
- After you click  (a manual refresh in the Git Repository Log)

## Conflict Handling in Git

In Git, a change that you made is referred to as “ours”, while a change made by a different user is referred to as “theirs”.

When you try to update your local repository, a conflict may occur between a local copy of a file (added/edited/deleted by “us”) and the file in the remote Git repository (added/edited/deleted by “them”) in the following cases:

- A file was deleted by you (“us”) and was modified by another user (“them”).
- A file was deleted by another user (“them”) and was modified by you (“us”).
- An item was added by both you (“us”) and another user (“them”) with different contents.
- The same item was modified by both you (“us”) and another user (“them”), and it is not possible to automatically merge the changes.

### Different Types of Conflict

There are three situations in which an author may come across conflicts – all these can be encountered on an **Update** operation, but one of them can also be seen on an **Unstash** operation. In each situation, the type of conflict that occurred is shown in the Conflicts dialog.

- **Merge:** Conflicts during a merge operation. In an **Update** operation, a merge is also made if the update strategy selected in the Update window was **Merge**). For details on how to resolve this type of conflict, see ["Resolving Conflicts: When to use Accept Our/Theirs and when to use Prefer Ours/Theirs" below](#)
- **Rebase:** Conflicts during rebase. In an **Update** operation, a rebase is also made if the update strategy selected in the Update window was **Rebase**.

**Note:** In this situation, the meaning of “ours” and “theirs” are opposite to the Merge case. So, for example, if the author selects “Accept/prefer ours”, he is accepting/preferring the change that comes from the remote branch, and if he selects “Accept/prefer theirs”, he is accepting/preferring the change from the local branch.

- **Unstash:** Conflicts during unstash. These occur if before the Update operation, the author has uncommitted changes that are in conflict with items that are located on the remote repository.

These conflicts behave exactly like conflicts that occur during rebase (as described ["Rebase: Conflicts during rebase. In an Update operation, a rebase is also made if the update strategy selected in the Update window was Rebase. "](#)). For details, see ["Manually Unstash Changes" on page 105](#).

See ["Resolve conflicts during an Update operation" on page 102](#) for details on how to resolve conflicts.

### **Resolving Conflicts: When to use Accept Our/Theirs and when to use Prefer Ours/Theirs**

Studio provides two methods of resolving conflicts: **Accept** and **Prefer**.

The **Accept Ours/Theirs** resolution uses the entire file from us/them. There is no merging of files, even if there are changes in some of the files.

The **Prefer Ours/Theirs** resolution automatically merges as many changes as possible into the selected file. If merging is not possible, the system selects changes from the preferred side. All other changes that cannot be automatically resolved, can be manually merged later in Studio. See ["Resolve conflicts during an Update operation" on page 102](#) for an example of automatic and manual resolution of conflicts.

For example, two users, user1 and user2, made changes on the same flow.

- user1 changed a certain step and added a flow-level input named **input1**.
- user2 changed the same step and added a flow-level output named **output2**.

In this case, if user1 has to resolve the conflict, s/he has the following options:

- **Accept Ours** - user1's entire flow will be used: user1's version of the step, with **input1**, but without **output2**.

- **Accept Theirs** - user2's entire flow will be used: user2's version of the step, **output2**, but without **input1**.
- **Resolve Manually Preferring Ours** - the flow will contain user1's version of the step, and both **input1** and **output2**.
- **Resolve Manually Preferring Theirs** - the flow will contain user2's version of the step, and both **input1** and **output2**.

**Note:** If there are conflicting changes on moved and renamed items during an **Update** operation, Studio detects these and tries to resolve them automatically as shown in the following example:

1. User1 moves a flow and makes some changes to it.
2. User2 makes some non-conflicting changes on the same flow.

Studio automatically detects these changes and merges them automatically. There is no conflict and both users can see their changes to the flow.

## Authentication Options

When connecting to a Git repository, you can choose to connect to the Git server using one of two authentication schemes: Credentials or SSH Private Key.

- **Git Repository Credentials: Password authentication**
- **Private key file:** To use this scheme, you need to fill in the Private Key File and Private Key Passphrase. You can generate a **public key-private key** pair by running the following command from the git command line:

```
$ ssh-keygen -t <algorithm_type> -C "your_email@example.com"
```

where `algorithm_type` may be `rsa1` for protocol version 1 and `rsa` or `dsa` for protocol version 2.

**Note:** The private key must be in OpenSSH format. For public keys, the RSA and DSA algorithms are supported. A public key must have an extension of **.pub** only.

In addition, you can configure known host parameters in the `studio.properties` file:

```
git.ssh.known.hosts.policy=allow|strict|add
```

The policy for known hosts. If set to `add`, a connection to a host that is not in the **known\_hosts** file will be allowed and the host and connection will be added to the file. If set to `allow`, Studio allows SSH connections to any host. If set to `strict`, only allow connections to hosts specified inside the **known\_hosts** file are allowed.



`git.ssh.known.hosts.file`

The location of the **known\_hosts** file. By default, the value is `${user.home}/.ssh/known_hosts`.

You can select the authentication type in the SCM Connection dialog. For details, see ["Connect to and Clone a Git repository" below](#).

### Connecting to a Git repository using a proxy

All Git operations that are available in Studio and interact with the Git server can be done using `http/https` connection URLs through a proxy. The proxy can be without authentication or using basic authentication scheme.

See ["Set the Proxy for Debugging on a Remote Central" on page 342](#) for details.

## What do you want to do?

### Connect to and Clone a Git repository

In order to work with the Git source control system, you must first connect to the Git repository and clone it to your local file system. When you clone a repository, Studio performs the following steps:

- Copies the repository into your Studio workspace
- Creates remote-tracking branches for each branch in the cloned repository
- Creates and checks out an initial branch that is forked from the cloned repository's currently active branch.

You can then work locally, commit your changes and then push the changes to the Git remote branch.

#### Note:

- Upon cloning, the default branch is always checked out.
- If the Git server verification fails, an error message opens. You can choose to continue or abort the operation.

1. Select **SCM > Connection....**

The **SCM Connection** dialog box opens.


2. In the **Type** area, select the SCM type: **Git**.
3. In the **SCM Repository** area, in the **URL** field, type or paste the URL of the Git repository.

**Note:** The URL may include an **http**, **https**, **file** or **ssh** protocol.

When using **https**, If the server certificate is not trusted, a message appears asking whether you want to trust this certificate. You can also import the certificate automatically as described in [""Import Certificates Automatically with a Remote Debugger Connection" on page 343.](#)

4. **Authentication:** There are two types of authentication available: **Credentials** or **SSH Private Key**.

- **Credentials:** In the **Type** dropdown, select **Credentials** to use Windows credentials, and enter the user name and password in the User Name and Password fields.
- **SSH Private Key:** In the **Type** dropdown, select **SSH Private Key** to use SSH private key authentication, and enter/select the private key file and passphrase in the **Private Key File** and **Private Key Passphrase** fields.

**Note:** Click on the  icon to select a private key file from a location on the disk.

5. **User Information:** Enter the user name and email address in the **Full Name** and **Email Address** fields. Both these fields must be filled in as this information is attached to every Git commit operation. If you do not fill in these fields, Git will use the computer name as the full name, and the `<computer name>@<full computer name>` as the email address.

6. Click **Clone**.

A message in the following format appears in the SCM messages pane:

```
<date> <time> - Clone https://<full_path>/gitTestRepo.git
```

```
Cloning from https://<full_path>/gitTestRepo.git  
to C:\Users\<username>\.oo\Workspace was done successfully.
```

**Note:**

- In order for the **Clone** command to run correctly with a destination directory that is not empty, Studio creates a blank directory (under `C:\...\oo\Workspace\git`) and copies the entire Git repository into the new directory.
- The first time you try to connect to Git, if you have filled in the Windows credentials, Git will use these to configure and populate the Git library.

**Note:** There is no Lock functionality in Git. This is because there is no need to lock files in Git, as there is the Git Conflict management. Therefore, the Lock icon in the editor for items (flows,

operations, configuration items) is always disabled – the same behavior as when you are not connected to any SCM repository.

## Add files to the local Git repository

1. From the projects tree, select the project whose files you want to add to the repository.
2. Right-click and select **SCM > Add**.
3. A message appears in the SCM messages pane. For example:

```
Add  
C:\Users\<<username>\.oo\Workspace\Project\Content\Configuration\Scriptlets
```

**Note:** You cannot add projects that are located outside the current workspace.

## Move/cross-move a versioned item

You can move an item from one location to another within the same project or cross-move an item between two different projects.

To move an item:

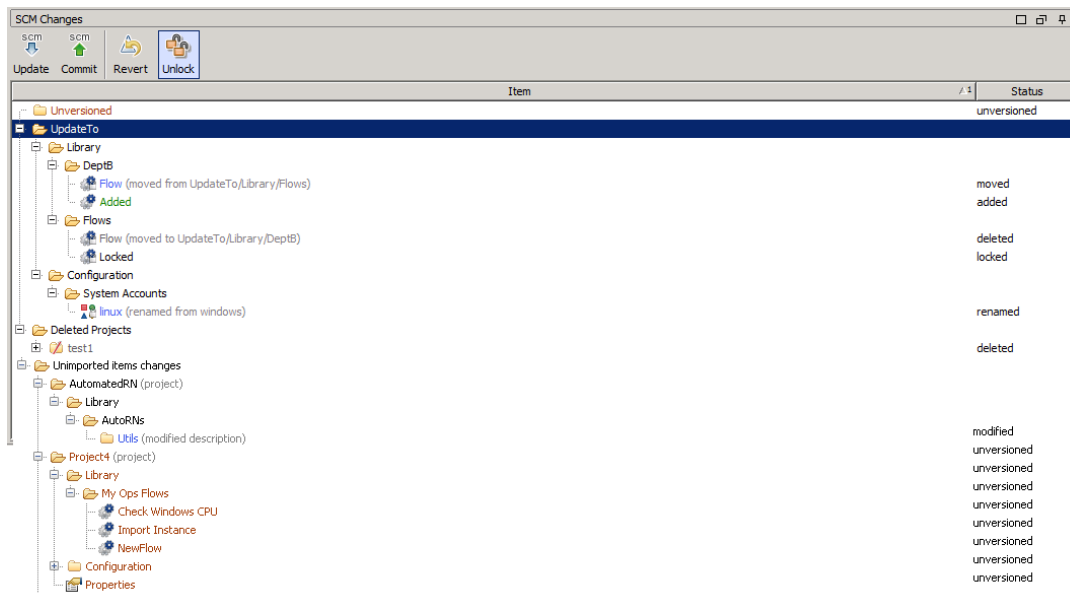
1. In the Project pane, select the items that you want to move. You can select multiple items using the **Shift** or **Ctrl** key.
2. Drag and drop the items into another folder or another project.

*or:*

Cut the items using **Ctrl+X** and paste them at the new location using **Ctrl+V**.

The item is marked in the SCM Changes pane in the old location with the status **deleted** and in the new location with a status **moved**. You can see the original location in parentheses. For

example:



**Note:**

- Where relevant you can commit the result of the move/cross-move action to Git.

## Update files from the remote Git repository

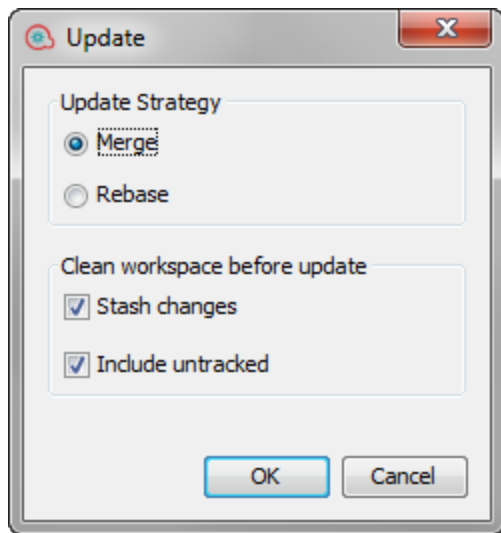
The **SCM Update** operation performs three Git operations, Stash, Pull and Unstash.

Stashing takes the current state of your local working workspace — that is, your modified tracked files and staged changes — and saves it on a stack of unfinished changes that you can reapply at any time.

If changes exist in your local workspace, and you have committed them, Studio tries to merge them automatically with the files in the remote Git repository. If there are conflicts, the Conflicts dialog opens and you can resolve the conflicts as described in ["Resolve conflicts during an Update operation" on page 102](#).

**Note:** You can also perform manual **Stash/Unstash** operations only using the **Stash** and **Unstash** options from the **SCM > Git** menu. For details, see ["Manually Stash Local Changes" on page 104](#) and ["Manually Unstash Changes" on page 105](#).

1. Right-click and select **SCM > Update**.
2. The following window opens:



3. Select the Git Update strategy you want to use:

- **Merge:** When using the update strategy Merge, Git preserves the original hierarchy of the local commits, and creates a special “merge” commit that merges the two diverged “branches” (the local branch and the remote branch) back together.
- **Rebase:** The Rebase strategy changes all the local commits to look like they were made based on the most recent changes from the remote repository. This shows in the repository log as a straight line, preventing the horizontal growth that is present when using Merge.

**Note:**

- You can change the default Update strategy using the **pull.rebase** and **branch.<name>.rebase** keys. For more information, see <http://git-scm.com/docs/git-config>.
- When performing a Rebase operation, there may be several conflicts for the same file.

4. Select the **Stash changes** check box to automatically stash local changes before updating.

5. Select the **Include untracked** check box to include the untracked items (the unversioned items marked in brown) in the stash.

**Note:** If you try to update from the remote Git repository and it is currently empty, Studio displays the following error message:

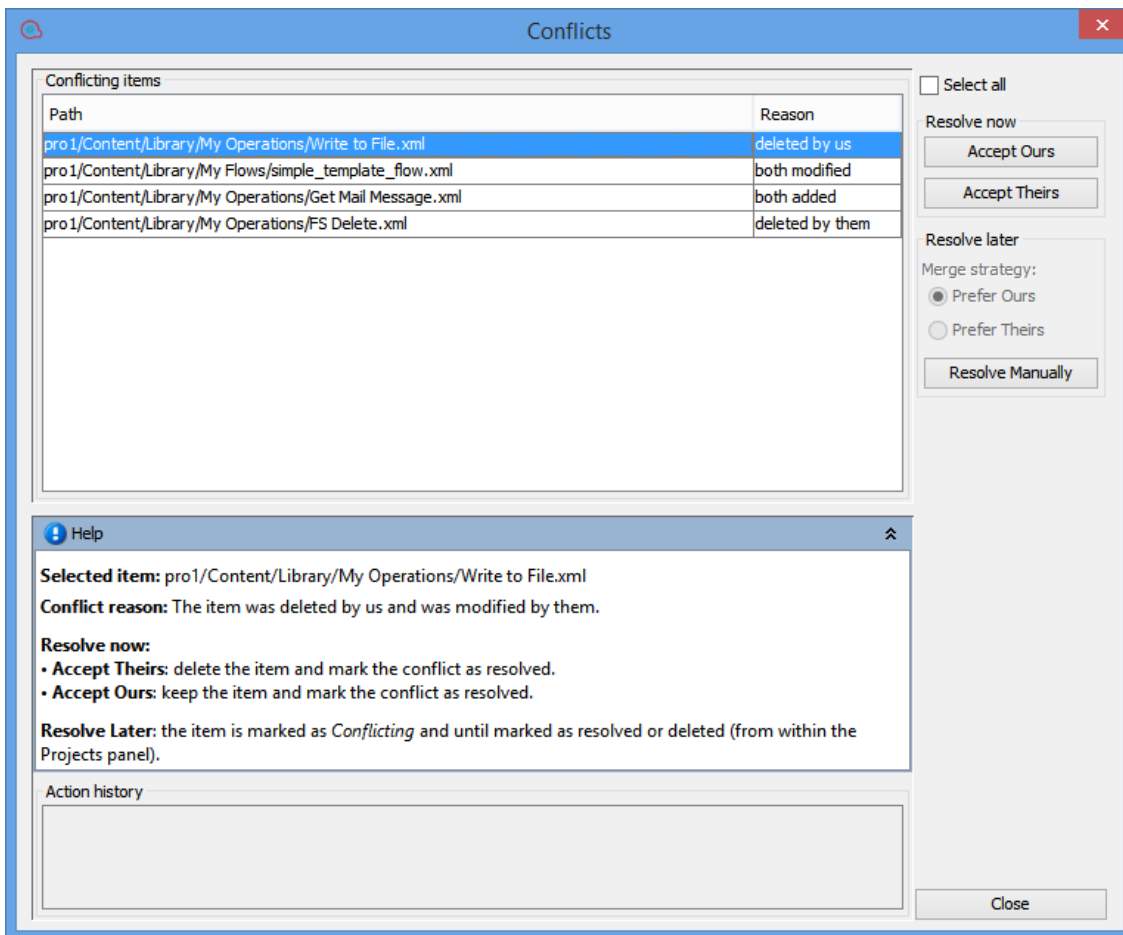
```
Cannot update the current branch as it does not exist in the remote repository.
```

## Resolve conflicts during an Update operation

Conflicts during an Update operation may appear in one of the following situations:

- **On merge:** If the author selects the Merge strategy on update and there are conflicting differences between the commits on the local branch and the commits coming from the remote branch.
- **On rebase:** If the author selects the Rebase strategy on update and there are conflicting differences between the commits on the local branch and the committed items coming from the remote branch.
- **On stash:** If the author has local changes that were not committed but have been stashed, and there are conflicting differences between these items and the committed items that come from remote branch. See "[Manually Stash Local Changes](#)" on page 104 for details.

You can see any conflicts that arise in the Conflicts dialog box. For example:



**Note:** There may be several conflicts in the same item. You must resolve all the conflicts in order to mark the item as resolved.

For each conflict, you can choose to resolve the conflict now or later.

### Resolve Now

- From the right pane, select **Accept Ours** to apply the changes made by "us" (the ones that are committed on our local repository).

or:

Select **Accept Theirs** to apply the changes made by "them" (the ones received by pulling from the remote repository).

### Resolve Later (Manually)

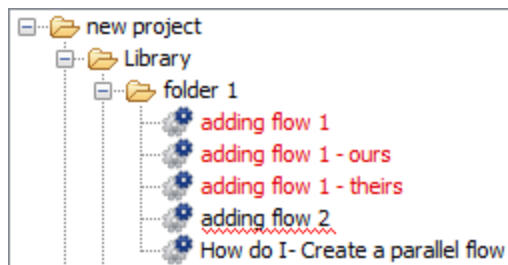
If you choose to resolve a conflict at a later stage, Studio prepares the item for manual conflict resolution. All non-conflicting changes are automatically merged into the item, and conflicting changes are replaced based on your selected preference (ours or theirs).

The original file is marked as **Conflicting** until marked as resolved or deleted (from within the **Projects** pane).

If the conflict type is not "deleted by us" or "deleted by them", Studio automatically creates two new items named `<filename> - ours` and `<filename> - theirs` shown in red:

- `<filename> - ours`: The original version of the item (from the local repository)
- `<filename> - theirs`: The original version of the item (from the remote Git repository)

For example:



These files are read-only and cannot be modified. After the conflict is resolved, Studio automatically deletes the "ours" and "theirs" files.

#### Note:

- When there are unresolved merge conflicts, there may be some operations that you will not be able to perform on the conflicting items, for example, committing them to the Git repository. In addition, you cannot move or rename conflicting items or the folders containing them.
- Conflicting files are shown in the Problems pane.

- The new file, created by merging conflicts, automatically or manually, may be in an invalid state.

To manually resolve the conflict:

1. From the right pane, select **Prefer Ours** to apply the changes made by "us" (the ones that are committed on our local repository).

*or:*

Select **Prefer Theirs** to apply the changes made by "them" (the ones received by pulling from the remote repository).

2. Select **Resolve Manually**.
3. Right-click on a conflicting item from the project tree, and select **SCM > Mark Conflict as Resolved**, **SCM > Resolve Conflict Accepting Ours** or **SCM > Resolve Conflict Accepting Theirs**.
4. Commit and push all the changes to the remote repository.

## Manually Stash Local Changes

Stash enables you to create a "patch" of your local changes saved inside your local workspace, that can be accessed later, at any time.

When you creates a stash, all the changes in the workspace are "stashed", and the local changes are *reverted*, i.e., removed from the workspace.

**Note:** If you try to stash changes while connected to an uninitialized, empty repository, Studio displays the following error message: "Before stashing, you must initialize your repository by a first commit".

**Note:** The **Stash** operations always include all changes made in the workspace.

You can also choose whether or not to include the untracked files.

**Note:** If you have unimported projects in your workspace, when you perform a **Stash** operation, these projects are included. After stashing, these files are stored in memory and not in the workspace.


However, the empty folders still remain. At any point, you can unstash your changes and the stashed files will return to their original location.

To stash local changes:

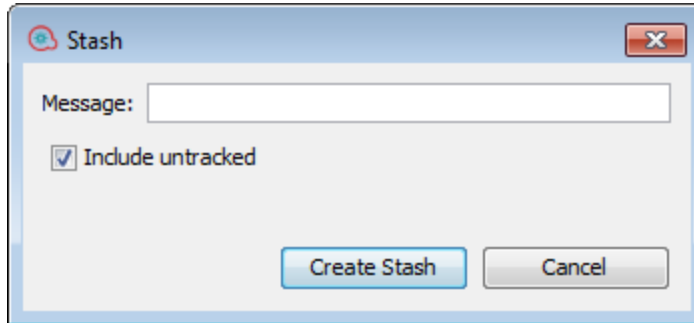


1. From the SCM menu, select **Git > Stash**.

or:

Click  in the SCM Change Pane toolbar.

The Stash dialog box opens:



2. In the Message field, type a message to identify the Stash operation. For example, `Changes to the Integration Operations`.
3. Select the **Include untracked** check box to include the untracked files in the stash.
4. Click **Create Stash**.

When the Stash operation has completed, you can see the operations that have been performed in the **SCM Messages** pane. For example:

```
04/14/15 15:13:30 - Stash
Stashing changes in the workspace including the untracked files.
Removed project(s): oo-virtualization-project
Stashed paths:
added: oo-virtualization-project/.gitignore
added: oo-virtualization-project/Content/Library.properties
added: oo-virtualization-project/Content/Library/Accelerator Packs.properties
added: oo-virtualization-project/Content/Library/Accelerator
Packs/Virtualization.properties
added: oo-virtualization-project/Content/Library/Accelerator
Packs/Virtualization/Get All Snapshots.xml
added: oo-virtualization-project/contentpack.properties
added: oo-virtualization-project/pom.xml
```

## Manually Unstash Changes

The Unstash operation enables you to apply previously stashed changes. This operation is equivalent to a **git stash apply** operation. The Unstash dialog box shows a list of stashes along with the details of the currently selected stash.

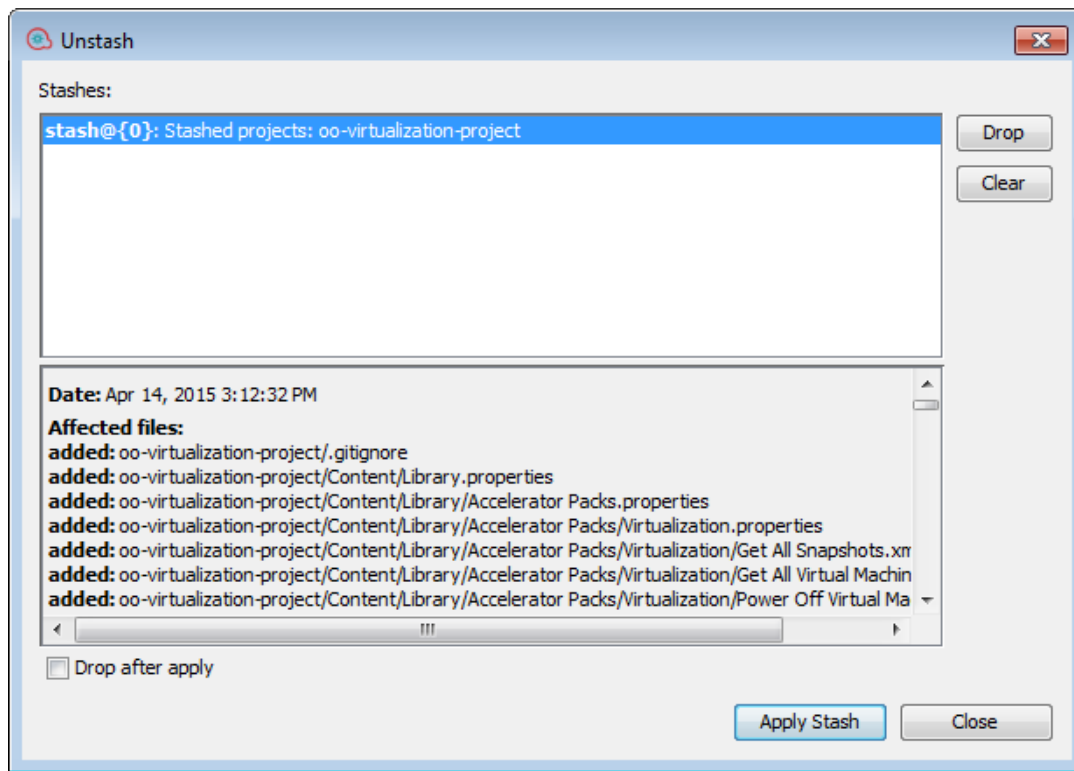
**Note:** Take care when running an Unstash operation as it may result in conflicts.

Conflicts can occur during an Unstash operation if the author has uncommitted changes that are in conflict with items that are located on the remote repository.

To unstash changes:

1. Select the stash commit you want to unstash.
2. From the SCM menu, select **Git > Unstash**.

The Unstash dialog box opens:



3. Select **Drop** to delete the currently selected stash.
4. Select **Clear** to delete all the stashes.
5. Select **Apply Stash** to implement the currently selected stash(es). If **Drop after apply** is selected, the **Apply Stash** operation will also delete the currently selected stash.

**Note:** If the author has uncommitted changes that are in conflict with items that are located on the remote repository, a conflict may occur here.

## Commit and push files to the Git repository

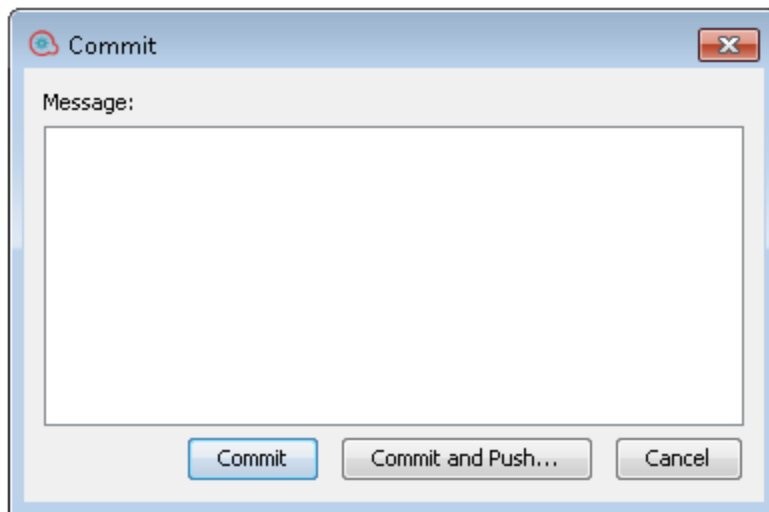
After making changes in Studio, you must commit the files to the local Git repository in order to record the changes, and then push them to the remote Git repository on the Git server.

You can choose to commit changes, and then push them later, or to commit and push them in the same step.

**Best practice:** In order to ensure that you have the latest files, before committing it is highly recommended to update the files (pull) from the remote Git repository using the command **SCM > Update**.

1. From the projects tree, select the folder whose files you want to commit to the local repository.
2. Right-click and select **SCM > Commit**.

The Commit dialog box opens:



3. Click **Commit** to commit the changes to the local repository.

*or:*

Click **Commit and Push...** to commit the changes to the local Git repository then push (copy) them to the remote Git repository.

**Note:** Before pushing, your entire SCM workspace must be up-to-date. The push operation pushes all committed files in the workspace to the remote Git repository. If there are uncommitted changes, they are not pushed to the remote repository.

You can also push changes to the remote repository using the **SCM > Git > Push..** option from the main menu. See ["Push files to the Git repository" on the next page](#) for details.

The following message appears in the SCM messages pane:

```
Pushed 1 commit(s) to origin/master
```

**Note:** Git does not store empty folders in the repository. However, in Studio, you can have empty folders inside projects, as Studio automatically creates them as required..

## Push files to the Git repository

After making changes in Studio, you must commit the files to the local Git repository in order to record the changes, and then push them to the remote Git repository on the Git server.

You can choose to commit changes, and then push them later, or to commit and push them in the same step. This procedure describes how to push changes to the remote Git repository.

**Note:** When you push your changes to the remote server, *all* your commits are pushed.

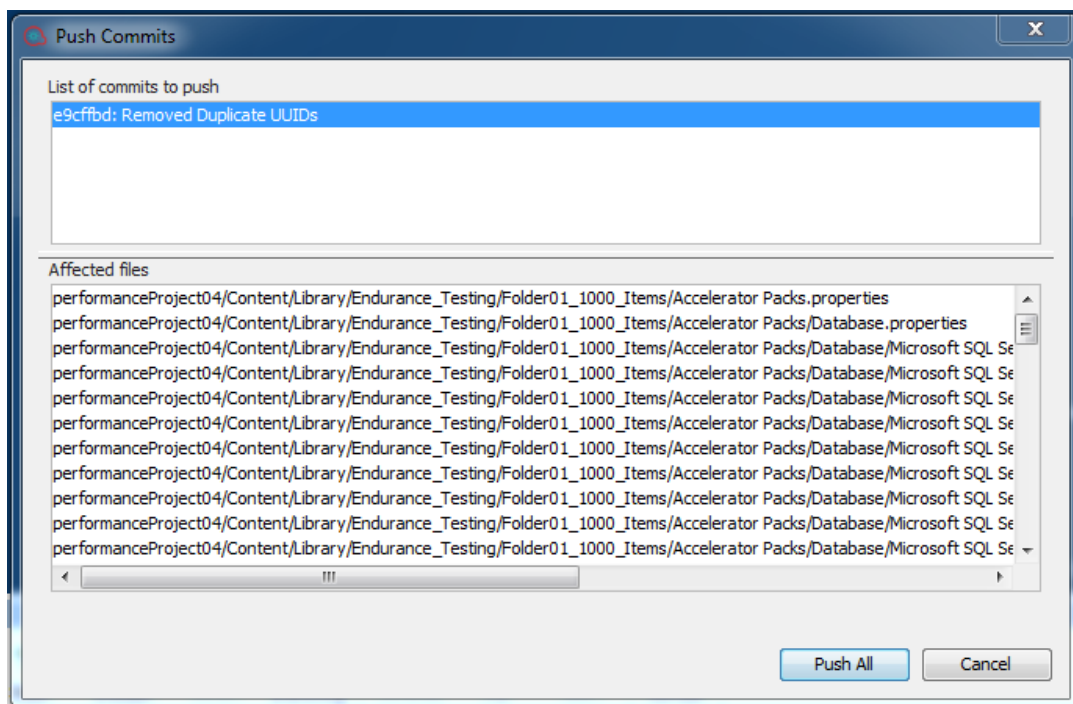
**Best practice:** In order to ensure that you have the latest files, before committing it is highly recommended to update the files (pull) from the remote Git repository using the command **SCM > Update**.

1. From the main menu, select **SCM > Git > Push....**

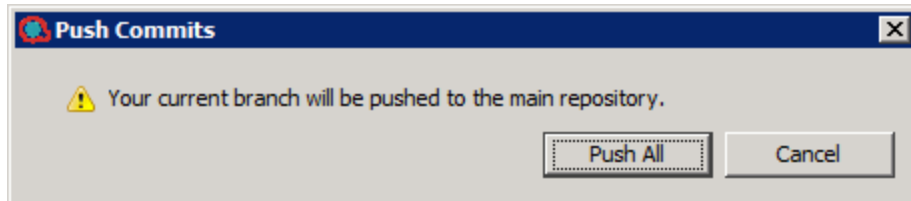
or:

From the SCM Changes pane, click  .

The Push Commits dialog box opens showing the list of commits to be pushed (along with the Commit ID) and an aggregated list of affected files for each commit:



If the branch is not pushed on the remote repository, the Push window looks as follows:



2. From the **List of commits to push**, select a commit to see the list of affected files in the **Affected Files** section.

**Note:**

- Before pushing, your entire SCM workspace must be up-to-date. This means that there must not be any files in the workspace that have not yet been committed or updated.
- If the remote Git repository is empty and you try to push to it without a local commit, Studio displays the following error message:

Before pushing, you must initialize your repository by a first commit.

The list of affected files is shown in the **Affected Files** section.

3. Click **Push All** to push all the commit operations to the remote repository.

A confirmation message appears in the SCM Messages pane. For example:

```
04/14/15 13:19:19 - Push  
Push successful  
Pushed 3 commit(s) to origin/master
```

## Abort Merging or Rebasing

There may be cases in which you do not want to accept changes from the branch you are merging/rebasing with. Aborting is common in the case of rebase.

When rebasing, you go through each commit from the branch you are rebasing and add it sequentially to the branch you are rebasing in. If you find a problem after several such commits, you can abort the rebase and then restart it.

In the case of merging, you may have a conflict in your workspace that you did not manage to resolve as you intended. In this case you can choose to abort the operation.

1. From the SCM menu, select **Git > Abort Rebasing or Merging**.

or:



From the SCM Changes toolbar, select

A confirmation message appears in SCM Messages pane.

## View a history of Git operations in the repository

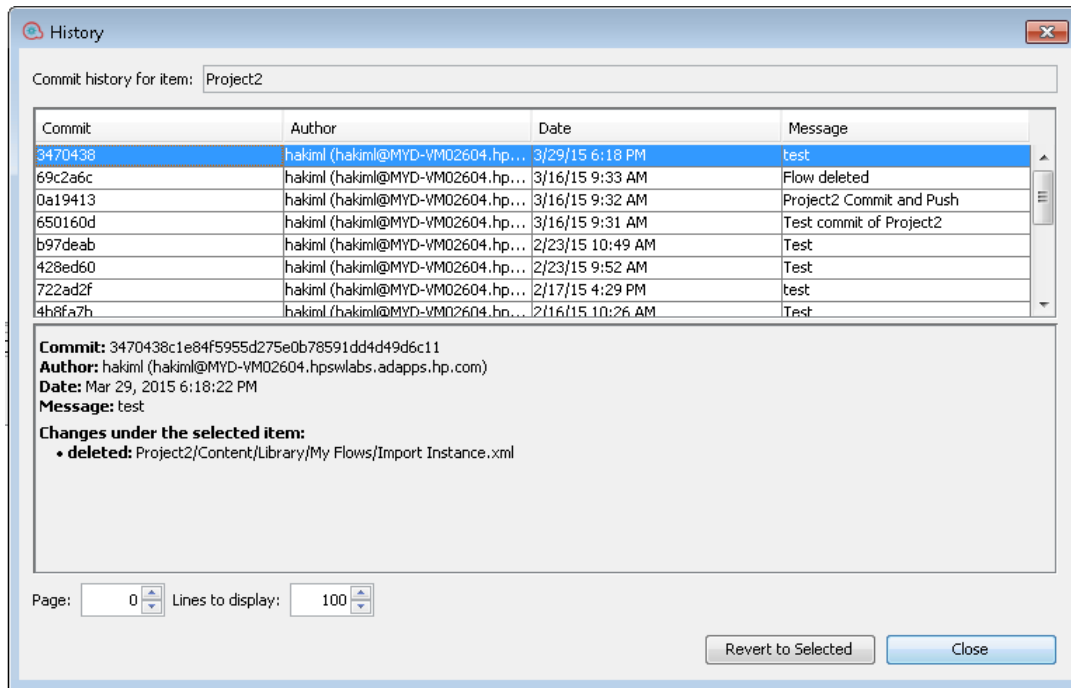
The **History** option shows you:

- A list of commits in which the selected project/item was affected
- All affected files for a particular commit

The **History** option also allows you to revert to a specific revision of the project/item. See ["Revert to a previous revision of a repository" on the next page](#) for details.

1. Right-click on a file/folder and select **SCM > History**.

The Revision History dialog box opens:



## Revert to a previous revision of a repository

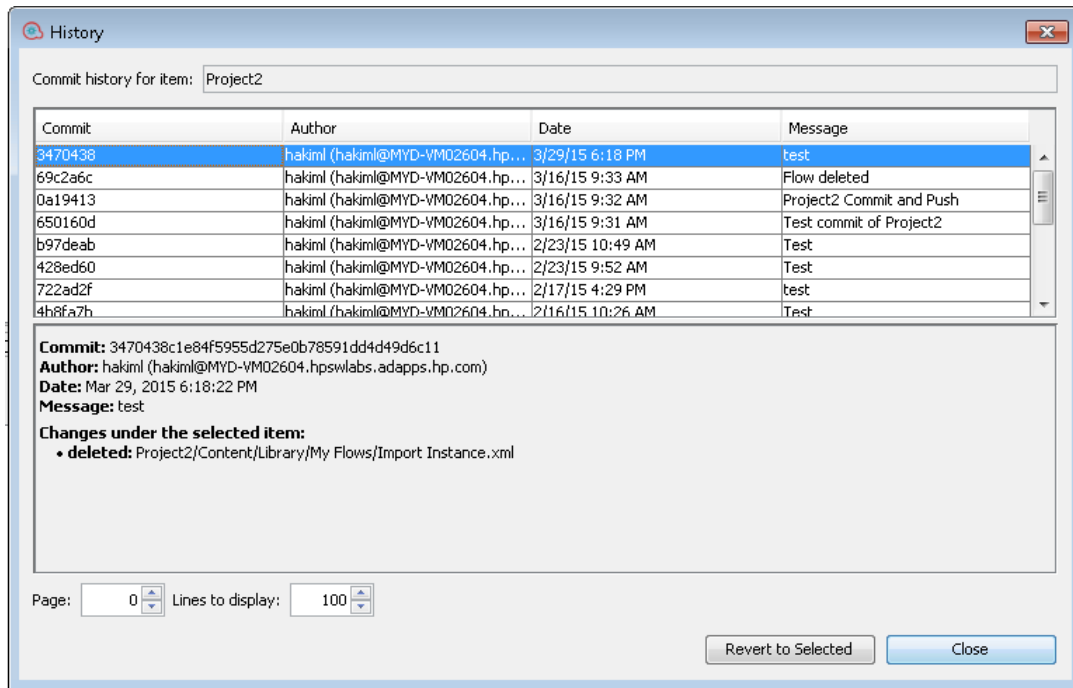
There may be cases for which you need to recreate a faulty Central content pack in order to troubleshoot a specific flow inside the content pack. If the content pack is not available (it has been lost or been overwritten by a new version), you can restore a specific version of the content pack using the **History** option.

**Note:** In Git, you can perform a **Revert** operation only on repositories.

To revert to a previous revision:

1. Right-click on a file/folder and select **SCM > History**.

The Revision History dialog box opens:



2. Select the commit revision you want to revert to.
3. Click **Revert to Selected**.

## Revert to a older version of a file/folder from the Git repository

Reverting removes the changes that the author made to the items (and in the case of folders, to all items they contain recursively), and returns the original state of the reverted items to the state they were in, the last time they were updated.

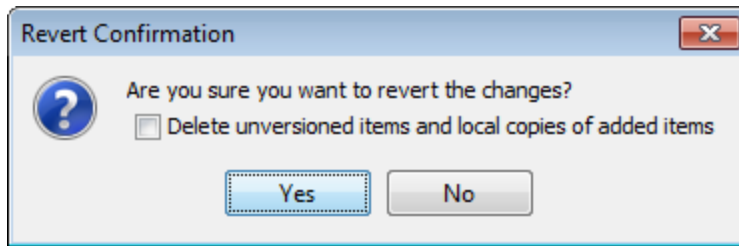
1. From the projects tree, select the requested folder/file(s).
2. Right-click and select **SCM > Revert**

or:

From the SCM Changes toolbar, select **Revert All Changes** 

The following message appears:





3. Select the check box if you want to delete all unversioned items and local copies of added items.
4. Select **Yes** to revert to the previous version.

### Reset to a tagged commit of a project/item

There may be cases for which you need to recreate a faulty Central content pack/project in order to troubleshoot a specific flow inside the content pack. If the content pack is not available (it has been lost or been overwritten by a new version), you can restore a specific version of the project/item using the **Reset to tag...** operation.

You can use the **Reset to tag...** option if you have already checked out the repository and you want to update your workspace to a specific revision from the SCM tag.

**Important!** The entire workspace reverts to the state it was in at the time it was tagged.

1. From the SCM menu, select **Git > Reset to tag...**
2. Type in the SCM tag that you received from the Central administrator. By default, the SCM tag is `[Project Name]-[Project version]`.

#### Note:

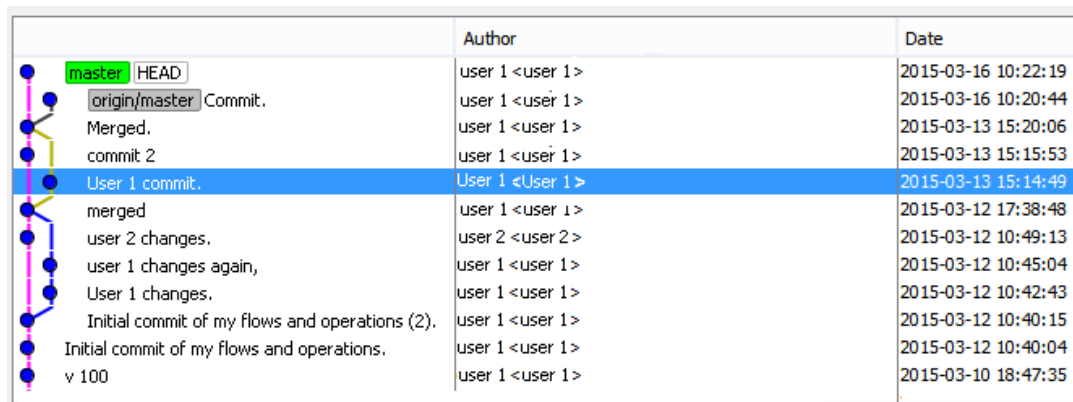
- For Git, the SCM tag must not include blank spaces.
- If you try to reset to a tag that is not on the current branch, an error message appears.

### View the SCM repository log

To view the SCM repository log:

1. Select the **SCM Repository Log** tab at the bottom of the screen.

The repository log opens. For example:



The image shows a Git repository log with columns for commit message, Author, and Date. The log includes several commits, with a merge conflict resolution highlighted in blue. The conflict resolution is shown as two different paths in the repository log.

	Author	Date
master HEAD	user 1 <user 1 >	2015-03-16 10:22:19
origin/master Commit.	user 1 <user 1 >	2015-03-16 10:20:44
Merged.	user 1 <user 1 >	2015-03-13 15:20:06
commit 2	user 1 <user 1 >	2015-03-13 15:15:53
User 1 commit.	User 1 <User 1 >	2015-03-13 15:14:49
merged	user 1 <user 1 >	2015-03-12 17:38:48
user 2 changes.	user 2 <user 2 >	2015-03-12 10:49:13
user 1 changes again,	user 1 <user 1 >	2015-03-12 10:45:04
User 1 changes.	user 1 <user 1 >	2015-03-12 10:42:43
Initial commit of my flows and operations (2).	user 1 <user 1 >	2015-03-12 10:40:15
Initial commit of my flows and operations.	user 1 <user 1 >	2015-03-12 10:40:04
v 100	user 1 <user 1 >	2015-03-10 18:47:35

2. If the local and the remote repositories have different versions that have been merged, the split and merge are shown as two different paths in the repository log (as shown in the example above).

## Add unversioned files to a Git repository

You can add unversioned files to the local Git repository at any stage. The files can be an entire project, a specific folder (either under “Library” or a configuration item folder), a flow, an operation or a configuration item.

### Note:

- You cannot add files from projects located outside the current workspace.
- When adding a file from an unversioned project, the entire project is added to the repository.

Added files appear in green in the Projects pane and the SCM Changes pane.

1. Right-click and select **SCM > Add**.
2. A confirmation message appears in SCM Messages pane.

## Rename versioned files

When renaming versioned files, the files are also scheduled for renaming in the Git repository.

Renamed items appear in blue in the SCM Changes pane.

**Note:** If the file is versioned but it has the status **Added**, the color of the file will be green even after a Rename action, both in the Projects Pane and in the SCM Changes pane.

If a Git error occurs during the operation, you can see it in the SCM Messages pane.

1. Right-click and select **Rename**.
2. Type a new name for the file and press Enter.

A confirmation message appears in SCM Messages pane.

### Delete versioned files

When deleting versioned files, the files are also scheduled for deletion in the Git repository.

Deleted items appear in gray in the SCM Changes pane. After you delete a file, you can still commit it to the repository from the SCM Changes pane. This is useful when working in multi-author environments.

If a Git error occurs during the operation, you can see it in the SCM Messages pane.

1. Right-click on the file(s) and select **Delete**.

The file is deleted from the local repository and a confirmation message appears in SCM Messages pane.

2. You can now commit the deleted file(s) from the SCM Changes pane to remove it from the remote Git repository.

### Detach from the Git repository

After you have finished working in the Git repository, it is a best practice to detach from the remote Git repository.

**Note:** Before detaching, make sure that you have committed and pushed all your local changes to the remote Git repository.

If a Git error occurs during the operation, you can see it in the SCM Messages pane.

1. Select **SCM > Settings**
2. In the **SCM Repository** area, click **Detach**.

An information message appears recommending that you to commit and push all your changes before detaching.

3. Click OK.

Your local repository is detached from the remote Git repository.

### Create a new branch in the Git repository

Creating a new (local) branch enables you to have a branch that initially points to the same commit as the current HEAD pointer. This means that the new branch will have the projects that Studio included when creating the branch. The newly created branch is immediately checked out and Studio connects automatically to the new branch). Any further commits are made to the current branch and you can choose later to push the branch to the remote repository.

**Note:** When connected to an uninitialized repository, the local branch will be created only after an initial commit.

If a Git error occurs during the operation, you can see it in the SCM Messages pane.

1. From the **SCM** menu, select **Git > Create a new branch...**

The Create New Branch dialog opens.

2. Type the name of the new branch in the field and click **OK**.

**Note:** A Git repository name must not include spaces and must follow standard Git naming conventions.

The new branch is created in the local Git repository and a message is shown in the SCM Messages pane.

3. To make the branch available to other authors working with the same remote repository, you can push the current branch to the remote repository using the **Push** command.

### Check out a branch in the Git repository

Checking out a branch allows you to connect to an existing branch. Connecting to an existing branch means that your workspace (along with the projects in it) will be changed to reflect the state of the newly checked out branch.

**Note:**

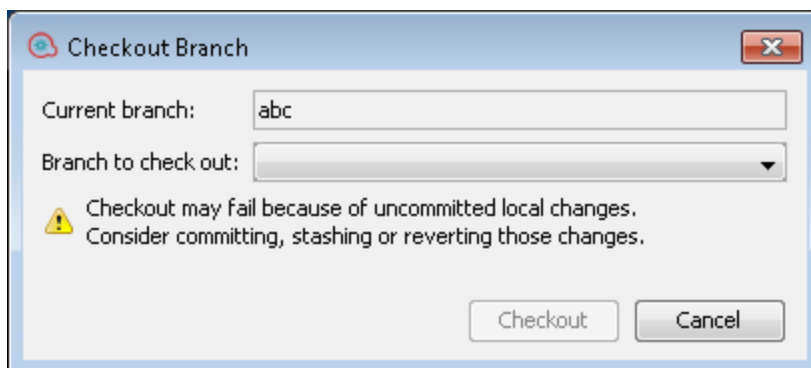
- Any project that exists in the newly checked out branch but does not exist in the current branch will be removed. The project data is not lost, however, as you can still check out that branch at a later stage).
- Any project that exists in both branches will be changed and refreshed to reflect its state in the newly checked out branch.
- Any project that exists only in the newly checked out branch will be copied to the disk, but Studio will not automatically import it.
- Checking out from a branch that has local changes is not recommended.

If the same files that are modified locally are different in the branch to be checked out, the checkout will fail, keeping the current branch and the local changes.

In addition, local changes are lost when the branch is checked out.

1. From the SCM menu, select **Git > Check out the existing branch.....**

The Checkout Branch dialog opens:

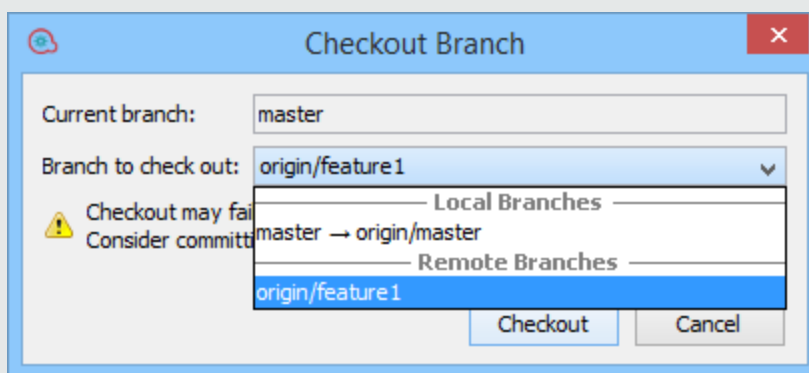


2. From the **Branch to check out** dropdown, select the local branch you want to work on.
3. Click **Checkout**.

The branch is checked out and ready for you to work on. You can see it the list of branches with a prefix of the remote repository's name. For example, `<master>-newbranch`.

**Note:**

- If a branch has not yet been pushed to the remote repository, it is listed with its name only under the Local Branches heading.
- If you check out a remote branch that was never checked out locally (such as in the example below), Studio creates a local branch with the same name and configures its upstream to the remote branch.



The slash sign means the branch is on the remote repository. For example **origin/** refers to a branch named origin on the remote repository.

The right arrow means that your local repository has previously been pushed to the remote repository and has a remote version. For example **master →origin/master** refers to a local branch **master** that has previously been pushed to the branch named **origin** on the remote repository.

There may be more remote branches that are not visible as the information has not yet been updated from the remote repository. In order to update this information, you need to perform a Fetch operation.

After check out, the following message is displayed in the SCM Messages pane:

```
07/07/15 14:29:28 - Branching  
Successfully checked out branch feature1 with upstream  
origin/feature1.
```

## View unimported items in the current workspace

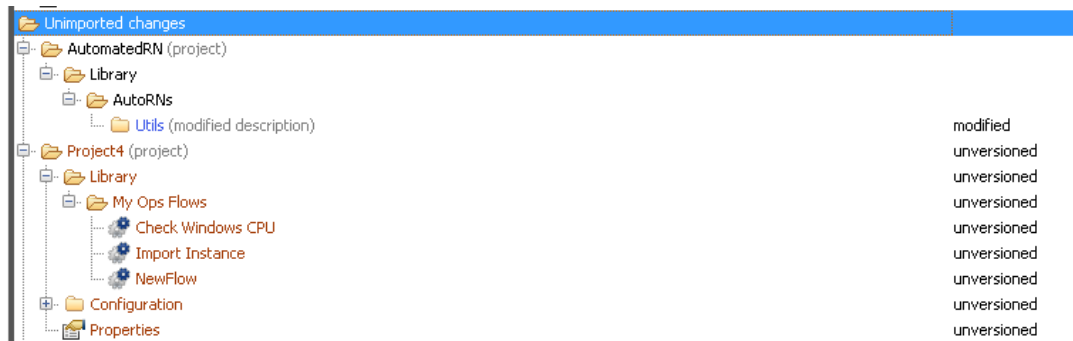
In the SCM Changes pane, you can see items (files, folders or projects) in the current workspace that have been changed and have not been imported into Studio. The SCM Changes pane displays changes between the local copy of the repository (the workspace) and the SCM repository. If a file is not changed, it is not displayed.

This information is important when you have conflicts, or when merging/rebasing and you need to understand all the changes that were made.

Unimported items may have one of the following states:

- **unversioned:** The item is new in the workspace, and has not yet been pushed to the Git repository.
  - **modified:** The item is already in the Git repository and has been changed.
1. Select the **SCM Changes** tab.
  2. In the Item tree, select the **Unimported changes** node.

A list of modified and unversioned files opens. For example:



From any item in the Unimported items changes node, you can perform the following operations:

- **Show in Explorer:** Right-click on an item and select **Show in Explorer**. Windows Explorer is opened at the item's directory, with the item highlighted.

- **Import Project** (from a project node only): Right-click on a project node and select **Import Project**. Studio imports the selected project, and you can see it in the workspace tree.

All other modified items that are not under the Studio Project structure can not be handled by Studio. You will need an external tool to handle them.

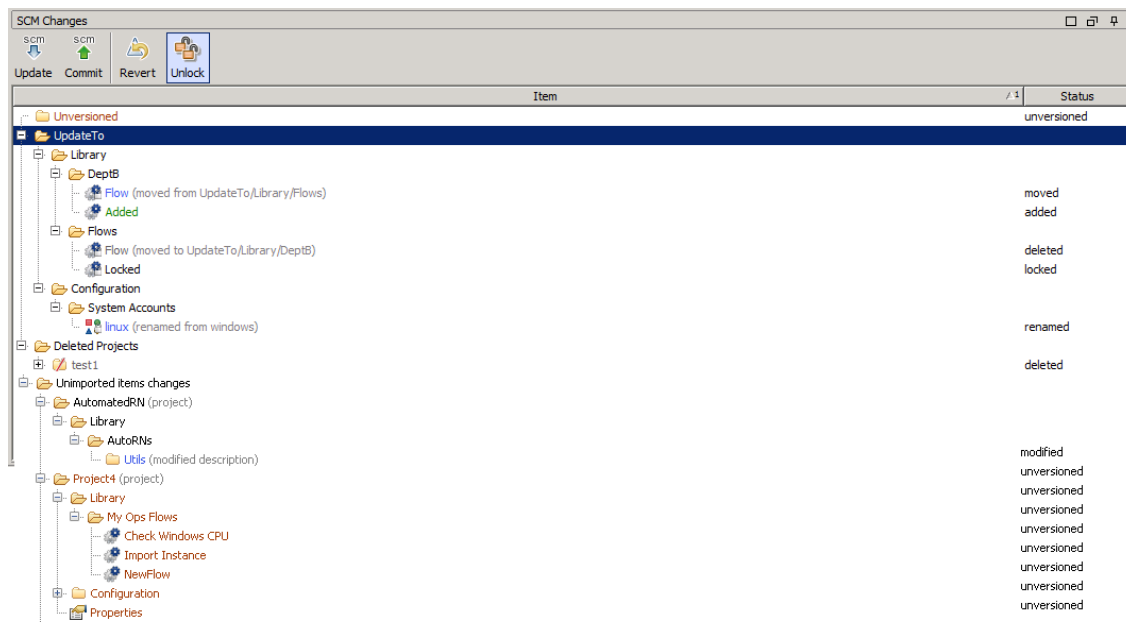
## Reference Material: SCM Changes Pane

The SCM Changes pane 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 **SCM Changes** pane. This pane also shows a list of deleted projects (projects marked for deletion), if such projects exist.

**Note:** Projects that are outside of Studio workspace are not be added/committed to SCM. They are shown as unversioned (in brown) in the Projects Pane, and are not shown in the SCM Changes pane.

### Types of changes

You can see the different types of changes in the Status column.



The types of changes are also reflected in the color coding. This may be:

Status	Color
added	green
renamed	blue

Status	Color
modified	blue
conflicted	red
moved (from/to)	blue
deleted	grey
unversioned	brown

In addition, you can see a text message in parentheses after the item name that describes the status of the item. For example, **moved from <old location>**, **moved to <new location>**, **renamed from <old name>**. You can also perform all standard SCM actions from these entries.

In this example, you can see:

- A deleted project, named **test1**, and colored gray.
- An added flow, named **Added**, colored green.
- An unversioned folder, named **Unversioned**, colored brown.
- A list of unimported items that have been changed.
- Several changed items, shown in blue:
  - A modified flow, **adding flow 2**.
  - A renamed folder, **new folder a**.
  - A moved flow, **flow 1234567**.
  - A folder whose description has been changed, **Utils**.

**Note:** Renamed and moved items have a special label, shown in parentheses, with the location where the item was moved or renamed from.

## State

The **State** area shows the state and the name of the current Git branch.

By default, the current Git branch is "master".













The state can be one of the following:







- **Ready:** the default state
- **Merging:** when you are in the middle of a merge operation and there are still unresolved conflicts



- **Merging - all conflicts resolved:** when you are in the middle of a merge operation and all conflicts have been resolved
- **Interactive rebase:** when you are in the middle of a rebase operation and there were conflicts

## SCM Changes toolbar

GUI item	Description
	<b>Update all:</b> Updates the entire Studio workspace.
	<b>Commit all changes:</b> Commits all the changes in the <b>SCM Changes</b> pane. Only available when there are changes.
	<b>Push changes:</b> Pushes all the changes in the <b>SCM Changes</b> pane to the main Git repository.
	<b>Stash:</b> Takes the current state of your local working directory — that is, your modified tracked files and staged changes — and saves it on a stack of unfinished changes that you can reapply at any time.
	<b>Unstash:</b> Takes the stack of unfinished changes from the stash and reapplies them on the local working directory.
	<b>Revert all changes:</b> Reverts all changes that show in the <b>SCM Changes</b> pane.
	<b>Branching:</b> Presents the Git branching operations. Click the icon to see the additional options, <b>Create</b> , <b>Checkout</b> , <b>Merge</b> and <b>Rebase</b> .
	<b>Create:</b> Creates a new Git branch.
	<b>Checkout:</b> Connects to an existing branch. After checking out, your workspace (along with the projects in it) will be changed to reflect the state of the newly checked out branch.
	<b>Merge:</b> Git preserves the original hierarchy of the local commits, and creates a special “merge” commit that merges the two diverged “branches” (the local branch and the remote branch) back together.
	<b>Continue rebasing:</b> Rewrites the project history by creating new commits for each commit in the original branch.
	<b>Skip commit in rebasing:</b> Rewrites the project history without creating new commits.

	<b>Abort:</b> Aborts the merging (for a merge operation) or stops the project history rewriting (for a rebase operation).
	<b>Refresh</b> Refreshes all the projects/items in the SCM changes pane.
<b>State:</b>	<p>Shows the name of the current repository (e.g., master) and the rebasing stage.</p> <p>When connected to an empty repository, the text (<b>&lt;empty&gt;</b>) appears. For example:</p> <div data-bbox="423 604 706 680" style="background-color: #f0f0f0; padding: 5px;"><p>State:  Ready (&lt;empty&gt;)</p></div> <p>During rebasing, the text <b>Interactive rebase</b> appears. For example:</p> <div data-bbox="423 768 834 827" style="background-color: #f0f0f0; padding: 5px;"><p>State:  Interactive rebase (17b9ff7)</p></div> <p>When merging, the text <b>Merging</b> appears. For example:</p> <div data-bbox="423 915 722 974" style="background-color: #f0f0f0; padding: 5px;"><p>State:  Merging (master)</p></div> <p>When merging after all conflicts have been resolved, the text <b>Merging - all conflicts resolved</b> appears. For example:</p> <div data-bbox="423 1087 933 1146" style="background-color: #f0f0f0; padding: 5px;"><p>State:  Merging - all conflicts resolved (master)</p></div> <div data-bbox="423 1178 1252 1318" style="background-color: #f0f0f0; padding: 10px;"><p><b>Note:</b> To complete the merge process, you must perform a final commit to get the files into a state ready for committing and pushing to the main Git repository.</p></div>

## SCM Message Pane

The SCM Message Pane displays the message results from the Git actions. Every action results in a message and is color-coded. In addition, a pop-up message appears informing the user of the result of the Git action. Clear the **SCM Message** pane by right clicking anywhere inside the pane and then clicking the **Clear All** button.

The type of messages that are shown depends on many factors, such as the update strategy, stashing, possible conflicts, or whether errors occurred when executing certain commands.

### Examples of Git Messages

In the following example, the current workspace is updated. This process includes stashing, updating and adding files.

```
SCM Messages
05/06/15 13:39:54 - Update
Updating workspace started
Stash
Stashing changes in the workspace including the untracked files.
Removed project(s): test2
Stashed paths:
  modified: AutomatedRN/Content/Library/AutoRNs/Generate Release Notes(9x).xml
  modified: AutomatedRN/Content/Library/AutoRNs/Generate Release Notes.xml
  modified: AutomatedRN/contentpack.properties
  modified: AutomatedRN/pom.xml
  added: test2/.gitignore
  added: test2/Content/Library.properties
  added: test2/contentpack.properties
  added: test2/pom.xml
```

### Error messages

In certain cases, errors appear. These are displayed in red. In this example,

```
05/07/15 10:55:16 - Push
http://admin@address.labs.company.com:8180/r/test01.git: cannot open git-receive-pack
There were 1 error(s).
```

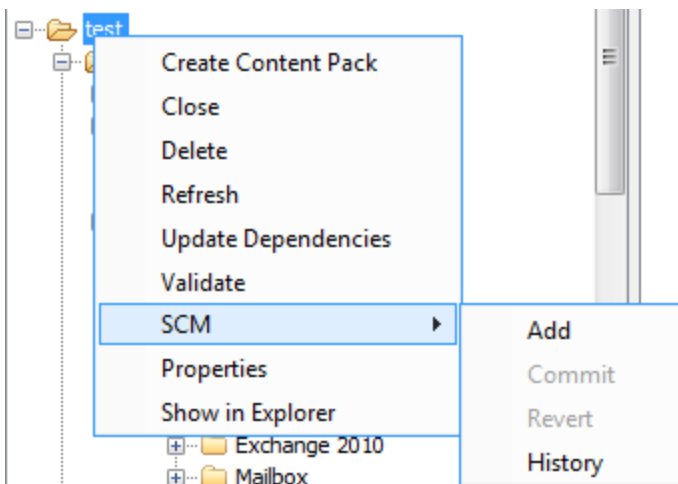
### Projects pane

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.

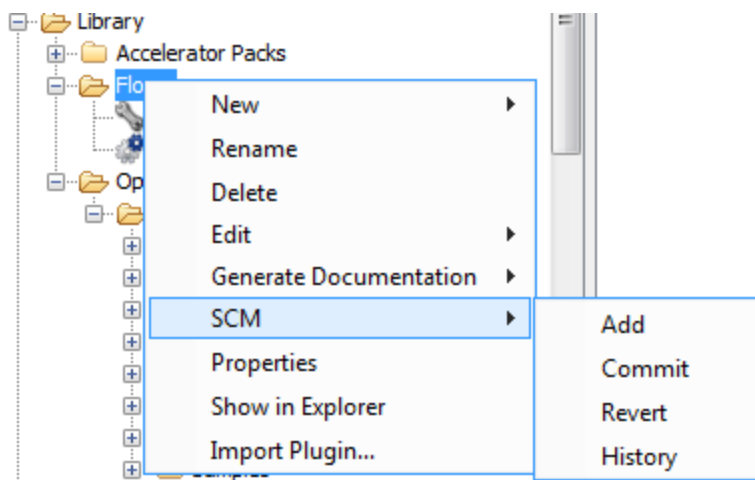
#### Context Menu from the Projects pane

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 **Add** and **History** are available:



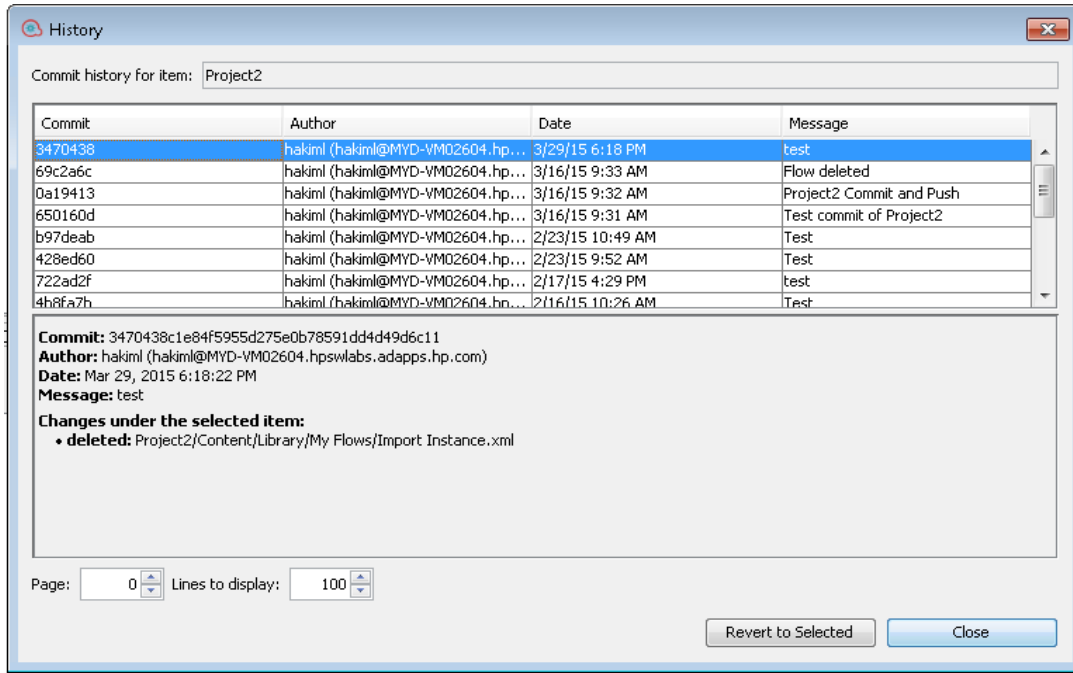
If the object was to be changed by an author, for example by adding a step, then it would have local changes. In this case, **Commit** and **Revert** are available.



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 you commit a folder, all of its child items are also committed. After you commit, you can add a comment for the commit.
<b>Revert</b>	Reverts all local changes in the selected items. Changes in flows, operations and configuration Items are reverted. Items that were deleted are restored.  <b>Important:</b> When you add an item it is automatically marked as added, but when you revert changes on it only the addition mark is removed, but the item still exists in Studio and in the file system. You can re-add the item using <b>Add</b> in the menu, or delete it.
<b>History</b>	The <b>Revision History</b> window, displays the SCM history. See " <a href="#">View a history of Git operations in the repository</a> " on page 110.

## History dialog box

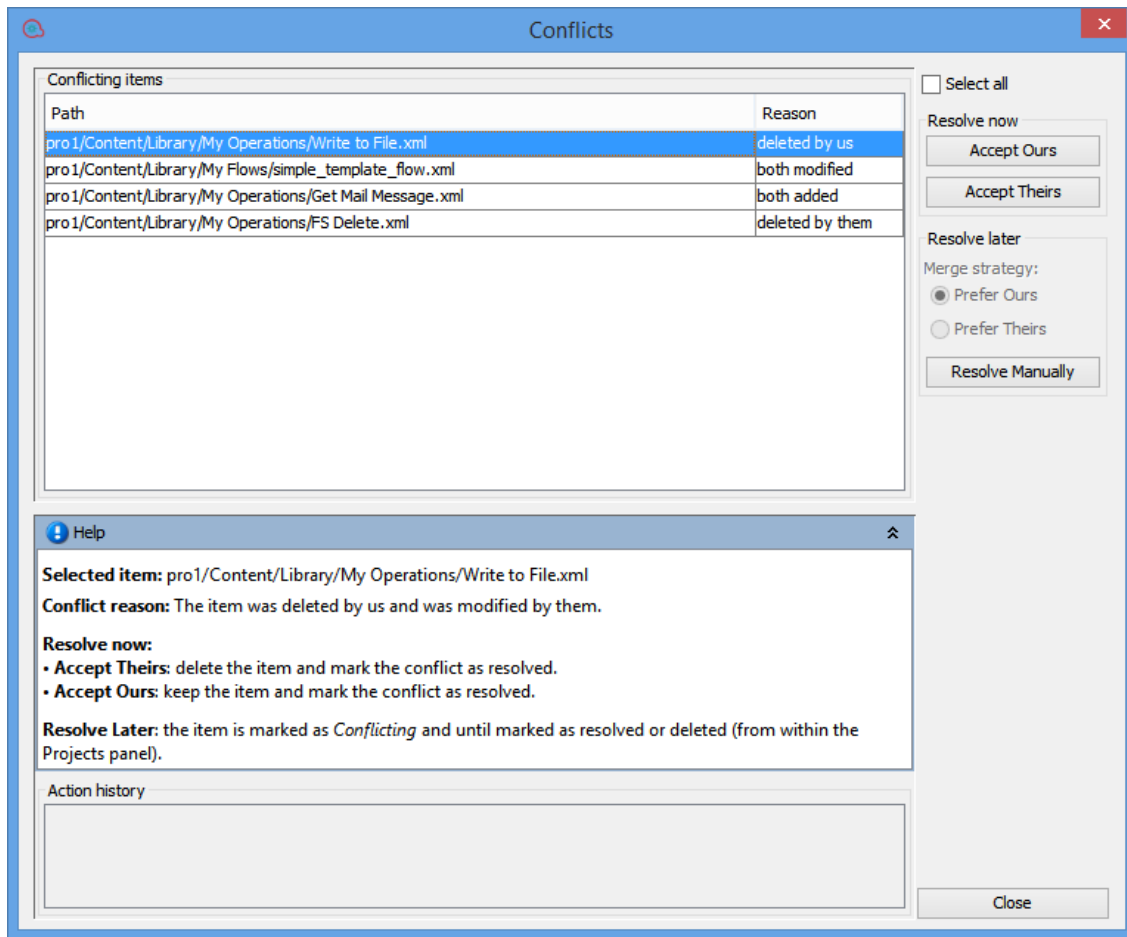
The History dialog box is used to revert to a previous revision of a project/item.



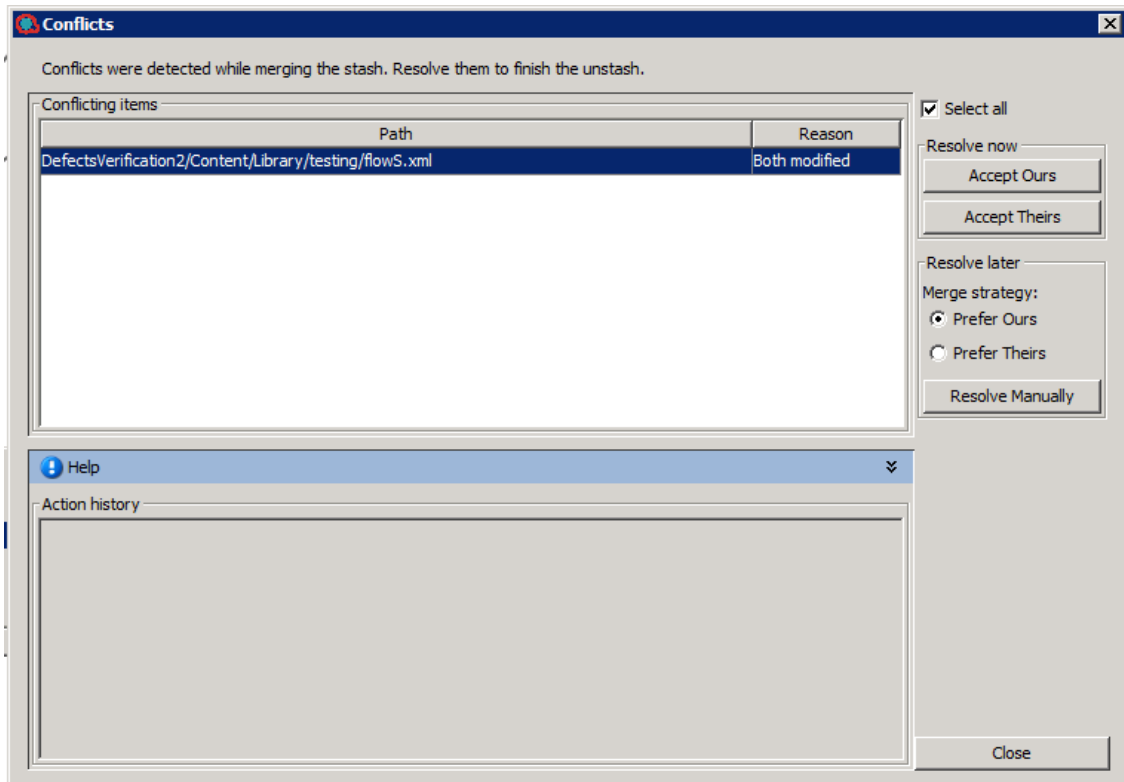
Option	Description
<b>Commit history for item</b>	Name of the project/item.
<b>Commit</b>	ID of the Commit operation.
<b>Author</b>	User name/id of the user who committed the changes.
<b>Date</b>	Date on which the Commit operation was done.
<b>Message</b>	Commit message written when the Commit operation was done.
<b>Changes under the selected item:</b>	Changes made and list of items changed for this commit.
<b>Page</b>	If the revision history is longer than one page, you can change the number of pages, and number of lines that are displayed on each page.
<b>Lines to display</b>	Enter the number of lines to be displayed on each page.
<b>Revert to selected</b>	Click <b>Revert to selected</b> to load the select commit of the project/item.

## Conflicts dialog box

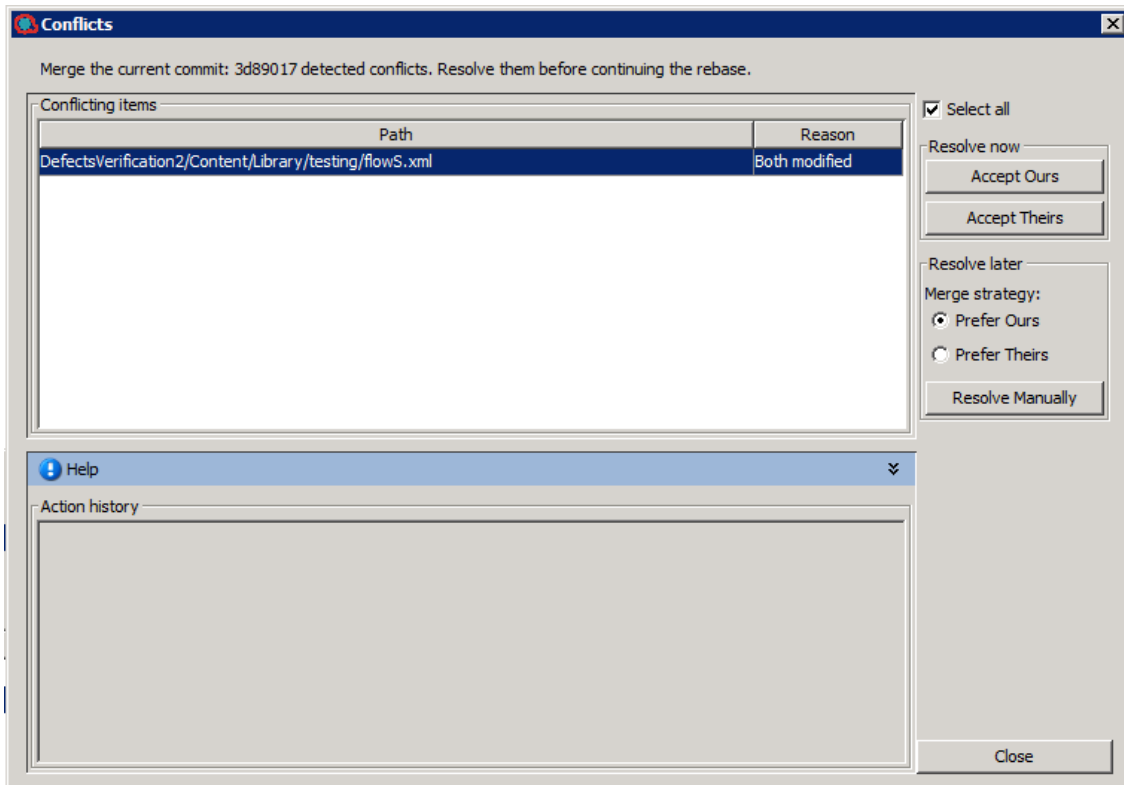
### Merge Example



## Stash Example



### Rebase Example



Option	Description
<b>Conflicting Items</b>	A list of the items in the project that caused the conflict and their paths. For each item, a reason for the conflict is shown.
<b>Select All</b>	Selects all the conflicting items from the list.
<b><u>Resolve Now:</u></b>	The conflicts are resolved immediately.
<b>Accept Ours</b>	Applies the changes made by "us" immediately.
<b>Accept Theirs</b>	Applies the changes made by "them" immediately.
<b><u>Resolve Later:</u></b>	The conflicts are resolved at a later stage.
<b>Resolve Manually:</b>	The original file is marked as <b>Conflicting</b> until marked as resolved or deleted (from within the Projects pane).
<b>Prefer Ours</b>	Changes made by us will be applied for the conflicts in the original file.
<b>Prefer Theirs</b>	Changes made by them will be applied for the conflicts in the original file.
<b>Help</b>	Shows the Help pane that provides additional details on the selected item.
<b>Action History</b>	Shows a log of all the executed operations. For example, the items that were resolved, how they were resolved, the items that will be resolved manually.  You can scroll through the log using the scroll bar.

### Resolve Later

If you choose to resolve a conflict at a later stage, Studio prepares the file for manual conflict resolution. All non-conflicting changes are automatically merged into the file, and conflicting changes are replaced based on your selected preference (ours or theirs).

The original file is marked as **Conflicting** until marked as resolved or deleted (from within the Projects pane).

In addition, Studio automatically creates two new files named <filename> - ours and <filename> - theirs representing the original versions from us and from them, respectively.

After the conflict is resolved, Studio automatically deletes the "ours" and "theirs" files.

### Resolve Manually

To manually resolve the conflict:

1. From the right pane, select **Prefer Ours** to apply the changes made by "us".



## 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



This section describes how to import content packs to Studio.

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.

**Note:** When you install Studio, you can select to import existing content packs in the Installation Wizard . 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.

Once you have imported a content pack, the files are available in the **Dependencies** 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 347.

You can 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 site and navigate to the Operations Orchestration Community page <https://hpln.hp.com/group/operations-orchestration>, and then navigate to the list of content packs.
2. Download the Base content pack 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 **Dependencies** pane, click the **Import** button .

**Note:** You can import a content pack from either the **Tree Viewer Merged View** tab in the **Dependencies** pane.

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 **Dependencies** pane, click the **Import** button  or from the File menu, select **Import Content Pack**.
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 **Dependencies** pane.

**Security Note:** Studio does not verify digital signatures. Therefore, it is recommended to manually verify the digital signature manually using well known tools, before importing it into Studio.

## 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 cannot be open in Studio at the same time.

## Managing Content Packs and Dependencies in a Project

Once a content pack is imported, you can use its 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, you can 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.

## Dependencies

A dependency is a project or content pack that includes a reference to an item contained in the current project. The item may be one or more of the following:

- Operation soft copies that reference operations inside other content packs/projects
- An operation group alias that comes from another content pack/project
- An operation/flow/step scriptlet that comes from another content pack/project
- Operation/flow outputs filters that come from other content packs/projects

- Operation response rule evaluators that come from other content packs/projects
- Operation response rule filters that come from other content packs/projects
- Operations, flows or step inputs that have an Otherwise assignment from system accounts, selection lists or domain terms inside other content packs/projects.
- Operations, flows or step inputs evaluators and domain terms coming from input validation format and input record under respectively that are inside other content pack/projects.
- Flow steps referencing operations/flows from other content packs/projects
- Step results filters from other content packs/projects
- Transition role aliases from other content packs/projects

Dependencies are stored in the project's pom.xml file.

### Dependency Conflicts

In general, the version of a declared dependency should match the version of dependencies that are automatically detected by Studio.

However, there are three types of conflicts that may occur between dependencies:

- **New conflict** – appears when the dependency is not declared in the project and Studio finds that this dependency is used. When resolving such a conflict, the dependency will be added with the version that exists in the current workspace.
- **Updated conflict** – appears when the detected version is different from the declared version. The conflict will only occur if the detected version is outside the range of the declared version. For example, if the declared version is a range 1.2 - 1.6 and the detected version is 1.3 there will not be a conflict as 1.3 is within the 1.2 - 1.6 range. If the declared version is a fixed version 1.5, and the detected version is a different fixed version 1.6, there will be a conflict. When **Apply Change** is checked for such a row in the Resolve Dependency Conflicts dialog box, the dependency will be updated in the list of dependencies of that project with the version taken from the Detected Version column.
- **Might not be used** – appears when the dependency is declared in the project but Studio does not detect that it is used. For example, you may have a project ProjectB that has been added as a dependency in your project ProjectA, but is not used in any of the flows, operations and configuration items from ProjectA. Or, you may have a dynamic reference to a system property inside ProjectB referenced from a flow inside ProjectA. In both these cases, ProjectB dependency will be tagged as a ***Might not be used*** conflict.

In this case, the dependency will be removed from the dependency list in the current project if **Apply Change** is checked.

### Updating Dependencies in a Project

If your project includes dependencies, the first step you must perform is to manually update all the dependencies as described in "Update dependencies for a project" below. You can then validate dependencies, add, edit or delete dependencies in the Dependency Editor.

## What do you want to do?

### Copy content pack objects to the project


1. Right-click the object to be copied in the **Dependencies** 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 **Dependencies** 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 Packs** tab in the **Dependencies** pane.
2. Select the content pack and click the **Delete** button .
3. Click **Yes** in the confirmation dialog box.

### Close a content pack

If you close a content pack, it is visible in the **Dependencies** 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 content 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 Packs** tab in the **Dependencies** pane.
2. Select the content pack that you want to close.
3. Right-click the content pack and select **Close**.

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

### Display the properties of an imported content pack

Right-click a content pack in the **Dependencies** 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 **Dependencies** pane. The Properties window for the object opens, in read-only mode.

### Update dependencies for a project

There are three methods that you can use to update dependencies:

1. Right-click on the project name, and from the menu, select **Update Dependencies**.

*or:*

2. In the Projects pane, right-click on **Properties**.
3. From the menu, select **Update Dependencies**.

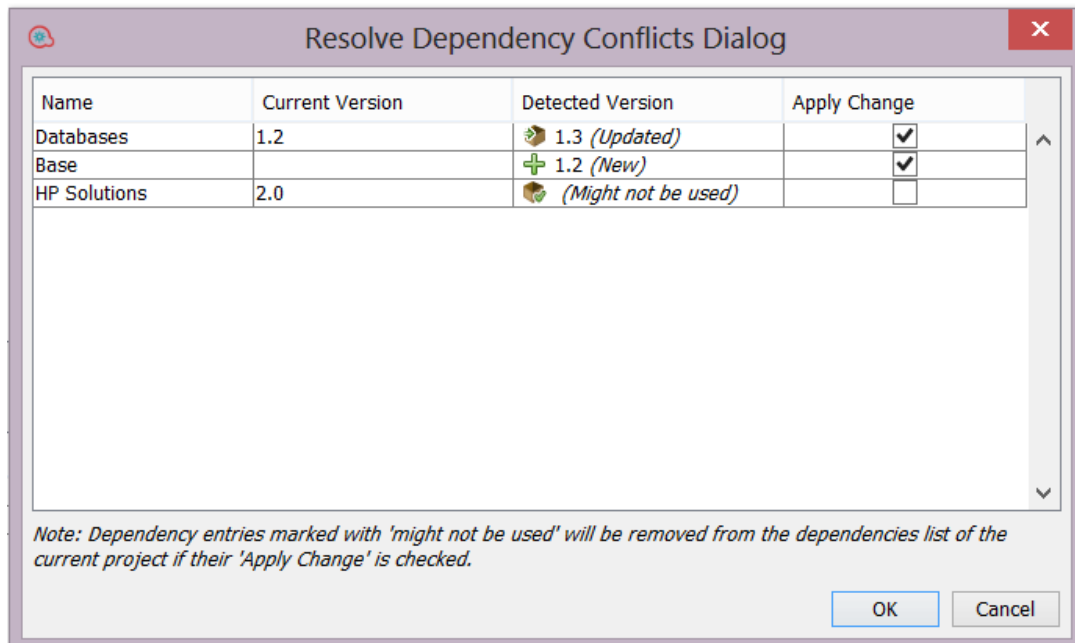
*or:*

From the Dependencies Editor toolbar, click .

If Studio finds a new dependency, it adds a new line corresponding to that dependency. If the dependency already exists, Studio updates it.

**Note:** In the case of an update, the current version is kept as long as it does not conflict with the dependency detected by Studio.

4. If there is a conflict between the current version and the version detected by Studio, the Resolve Dependencies Conflict dialog box opens:



5. Check **Apply Change** for each row, in order to resolve the conflicts.

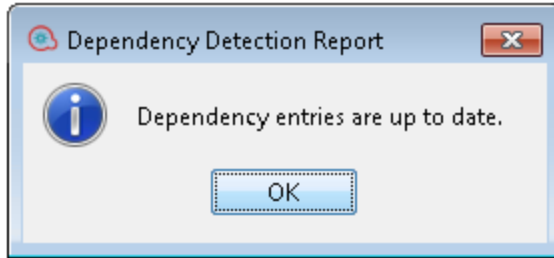
**Note:** If a dependency is marked as *{Might not be used}*, this means that the dependency is declared in the project but Studio cannot identify whether or not it is used. For example, you may have an input in a flow that has a dynamic reference to a system property in another project. In this case, Studio cannot identify whether or not the system property is used in the flow.

For these dependencies, **Apply Change** is unchecked by default. After checking **Apply Change** and clicking OK, the dependency will be removed from the dependency list in the current project.

6. Click OK.

After clicking OK, Studio applies (updates, adds, removes) the selected dependencies (as marked in the **Apply Change** column).

If there are no conflicts, or after all the conflicts have been resolved, the following message appears:



## Validate dependencies for a project

This procedure shows how to validate dependencies manually. However, there are cases in which the dependency validation is triggered automatically:

- When the dependencies of a project are changed and saved
- When a project is updated from SCM either as individual project or as part of an **Update All** operation
- After a project is imported
- When a project is opened

**Note:** The validation process checks the range of dependent items of a project and also checks that the version of dependent projects match the project version.

To manually validate dependencies:

1. In the **Projects** pane, right-click on **Properties**.
2. From the menu, select **Validate**.

If there are any dependency warnings or errors in the project, you can see them in the **Problems** pane. For example:

Type	Source Type	Name	Description	Location
Warning	Dependencies	test_Responses	Missing 'new_eval_Ref:1.0.0-SNAPSHOT' dependency for project/content pack 'test_Responses'	/test_Responses
Warning	Dependencies	test_Responses	Missing 'Base:1.2.1' dependency for project/content pack 'test_Responses'	/test_Responses
Warning	Dependencies	test1	'test2:2.0' dependency for project/content pack 'test1' not present in workspace	/test1
Warning	Dependencies	test1	Possibly unused 'test2:2.0' dependency for project/content pack 'test1'	/test1
Warning	Dependencies	test1	Missing 'Base:1.2.1' dependency for project/content pack 'test1'	/test1
Error	System Account	sa	The system account name 'sa' is a duplicate of /new_eval_Ref/Configuration/System Accounts/sa	/test_Responses/Configuration/System Acco...
Error	System Account	sa	The system account name 'sa' is a duplicate of /test_Responses/Configuration/System Accounts/sa	/new_eval_Ref/Configuration/System Acco...
Error	SystemProperty	sp1	The system property name 'sp1' is a duplicate of /meserasul/Configuration/System Properties/sp1, /...	/test_Responses/Configuration/System Prop...
Error	SystemProperty	sp1	The system property name 'sp1' is a duplicate of /meserasul/Configuration/System Properties/sp1, /...	/new_eval_Ref/Configuration/System Prop...
Error	SystemProperty	sp1	The system property name 'sp1' is a duplicate of /test_Responses/Configuration/System Properties/s... /meserasul/Configuration/System Propertie...	/meserasul/Configuration/System Propertie...

After the validation, the problems pane is updated with the new status of the dependencies.

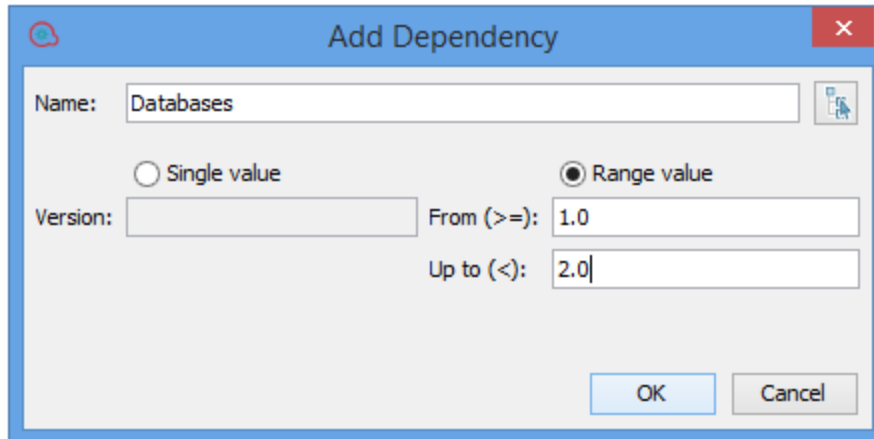
## Add a new dependency for a project


In addition to the dependencies that are automatically added to the project, you can add new dependencies, even if they do not exist in the workspace.

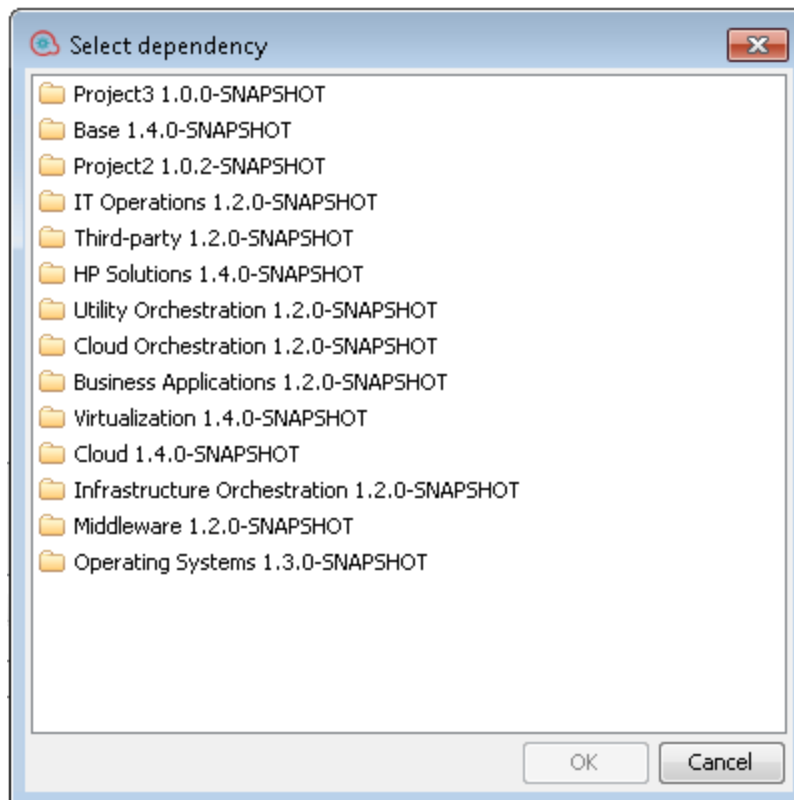



1. Click  in the Dependencies Editor toolbar.

The Add Dependencies dialog box opens.



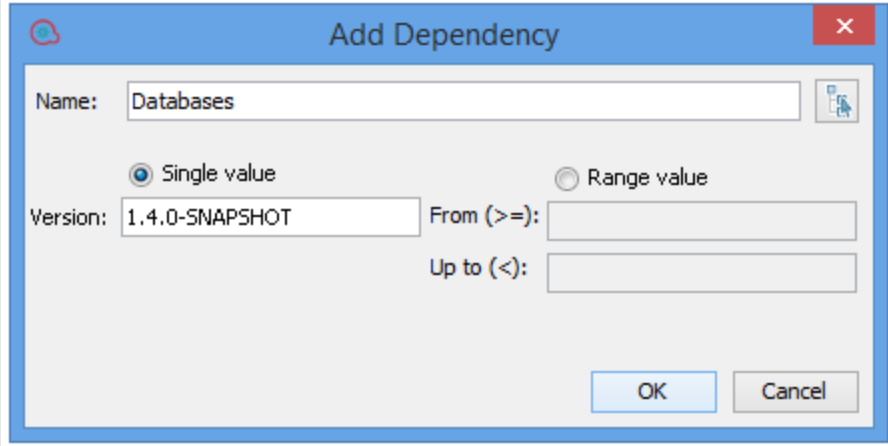
2. In the **Name** field, type the name of the dependency you want to add or click  and select the dependency from the list of current projects and content packs that have been imported into Studio:



For example, if your project uses information from the Middleware content pack, click  and select type **Middleware 1.2.0-SNAPSHOT**.

3. In the **Version** field, type the version of the dependency you want to add. You can use a numeric value (up to 99999999) or a snapshot version, e.g., 1.4.0-SNAPSHOT .

**Note:** If you selected from the list of dependencies , the **Version** field is automatically filled with the version number. For example:



The screenshot shows a dialog box titled "Add Dependency". It has a blue header with a close button (X) on the right. Below the header, there is a "Name:" label followed by a text input field containing "Databases" and a small dependency icon on the right. Below that, there are two radio buttons: "Single value" (which is selected) and "Range value". Under "Single value", there is a "Version:" label followed by a text input field containing "1.4.0-SNAPSHOT". To the right of this field are two more input fields: "From (>=):" and "Up to (<):", both of which are currently empty. At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

In addition, if there is no official current version, the snapshot version is used.

4. To define a range of versions to be supported in the dependency, select **Range value**, and type the range in the **From** and **To** fields. For example, if you are currently using Base version 1.4.0, but you want your project to be backward compatible with version 1.2.0, type 1.2.0 in the **From** field, and 1.4.0 in the **To** field.


**Note:** The **From** field includes the value entered, as well as values higher than it, while the **To** field does not include the value entered. For example, "from 1.2.0 to 1.4.0" means higher than and including 1.2.0, but lower than 1.4.0.

You can also specify all values lower than... by entering a value only in the **To** field, or all values greater than.... by entering a value only in the **From** field.

A range is marked by *(range)* in the Dependencies Editor.

**Note:** In the inline editor, you can automatically define a range by typing the range in the **From** field. For example 1.2.0 - 1.4.0.


## Edit a dependency

1. In the Dependencies Editor toolbar, select the dependency you want to edit.
2. Click .
3. Edit the dependency definitions as required.

## Delete a dependency


You can delete dependencies from the project.

**Note:** The dependency is only removed from the project, not from the file system.

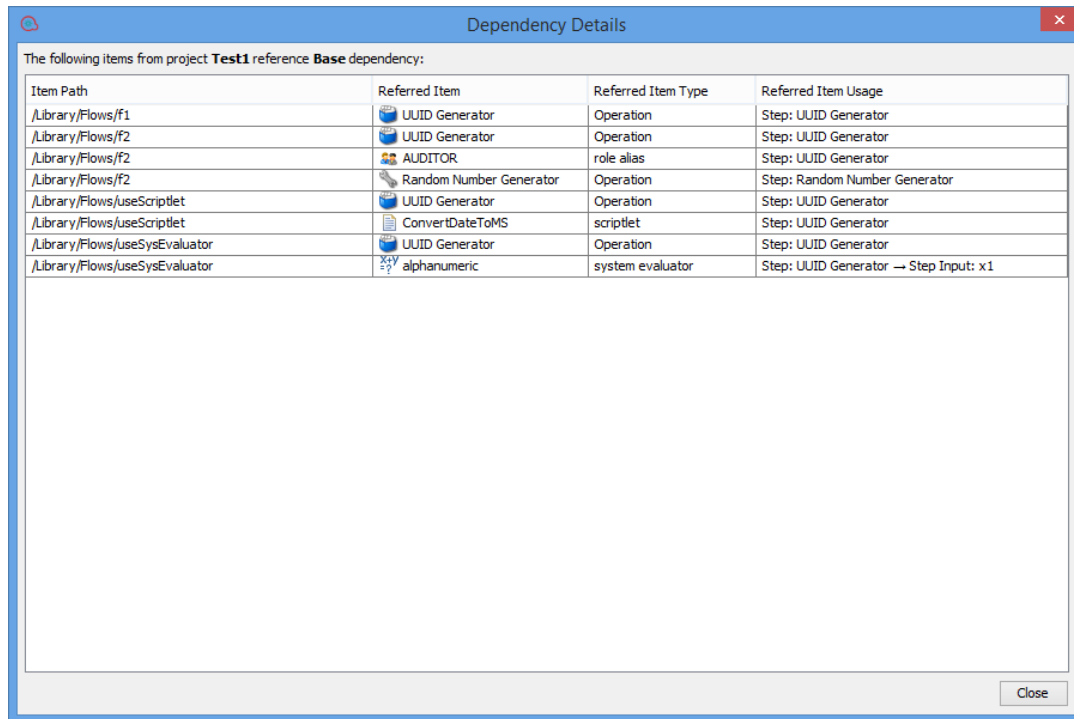
1. In the Dependencies Editor toolbar, select the dependency you want to delete.
2. Click .

## Show dependency details

From the Problems Pane, you can take a closer look at the dependencies that caused warning messages, and then update the dependencies as required.

1. In the Problems Pane, select the error/warning that you are interested in.
2. From the toolbar, click .

The Dependency Details dialog box opens showing a list of items in the project that reference the selected dependency. For example:



4. Double-click on an item to load it in Studio.

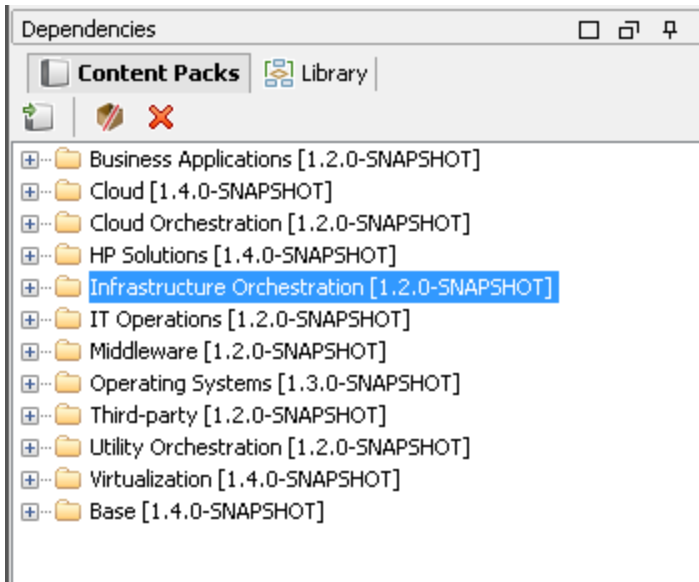
## Reference Material





### Dependencies pane

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

The **Dependencies** pane contains two tabs:

- **Content Packs** - Displays multiple trees, for multiple content packs. From this view, it is possible to close, delete, or import a content pack.
- **Library** - Displays a single tree, with all of the content merged together under a general **Library** folder and a general **Configurations** folder. From this view, it is possible to import a content pack.



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 project and select **Properties**, the Properties sheet opens.

In the Properties sheet for a project, you can update all the properties. However, for a content pack, the properties are read-only.

You can show the same information from the Dependencies pane. 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.

Name:	Base [1.4.0-SNAPSHOT]
Uuid:	75b8b3d6-d260-43af-acf9-8a142b4feadf
Version:	1.4.0-SNAPSHOT
Publisher:	Hewlett-Packard
Description:	Base Content Pack contains flows and operations to be used in every automation use case. Inside you can find technologies such as HTTP Client, Email, Remote Command, XML\JSON processing and
Dependencies:	Base Content Pack does not depend on any other Content Pack.

GUI item	Description
<b>Name</b>	The name of the content pack is taken from the project name. This field is read-only.
<b>Uuid</b>	A unique identifier for the content pack.
<b>Version</b>	The content pack version.
<b>Publisher</b>	Enter the publisher of the content pack. This information will appear in the content pack's <b>Properties</b> sheet.
<b>Description</b>	Enter a description of the content pack. This information will appear in the content pack's <b>Properties</b> sheet.

## Dependencies Editor


At the bottom of the Content Pack/Project Properties page, the Dependencies Editor is shown.




In the Dependencies Editor, you can see all the installed content packs, their versions and their dependencies for the current project. You can edit these values as required.

Dependencies:








Name	Version
Base	1.4.0-SNAPSHOT
Middleware	1.2.0-SNAPSHOT
Infrastructure Orchestration	1.2.0-SNAPSHOT

GUI item	Description
<b>Name</b>	The content pack or project name
<b>Version</b>	<p>The content pack version. You can edit this version with one of the following:</p> <ul style="list-style-type: none"> <li>• <b>A fixed version number:</b> For example, 1.2.0. This means that you can only use the exact version of the content pack.</li> <li>• <b>Interval:</b> For example, 1.2-. This means that you can use any version greater than or equal to 1.2.</li> </ul> <p>For example, -1.2. This means any version strictly less than 1.2.</p> <ul style="list-style-type: none"> <li>• <b>Interval with range:</b> For example, 1.2-3.0.1. This means that you can use any version from 1.2 up to 3.0.1 (up to but excluding 3.0.1).</li> </ul>
<b>Add</b> 	Open the Add Dependency dialog to add a new dependency for the current project.

<b>Delete</b> 	Delete the selected dependency from the current project.
<b>Edit</b> 	Open the Edit Dependency dialog to change the definition of the selected dependency.
<b>Update Dependencies</b> 	Show and resolve any dependency conflicts in the current project.

## Problems Pane toolbar



GUI item	Description
	Where relevant, open the flow related to the problem.
	Update dependencies in the current project.
	Add a dependency for the current project.
	Delete a dependency from the current project.
	Show details of dependencies in the current project.

## Dependencies in the Problems Pane

If any issues occur when you validate dependencies, you can see them in the **Problems** pane.

There are three types of dependency issues that may occur. For each issue, a specific warning message is shown in the Problems pane:

- **Missing Dependency:** The project contains links to items from another content pack/project but the content pack/project containing the items are not declared in the project's dependencies list. To see
- **Probably unused dependency:** The declared dependencies of a project contains a dependency that cannot be found by the Studio dependency detection rules. However, these rules do not detect dynamic references for system properties, so the declared dependency could exist for a system property dynamic reference, or may not exist at all.
- **Not in workspace:** The project links to a project or content pack that is not currently imported in Studio workspace.

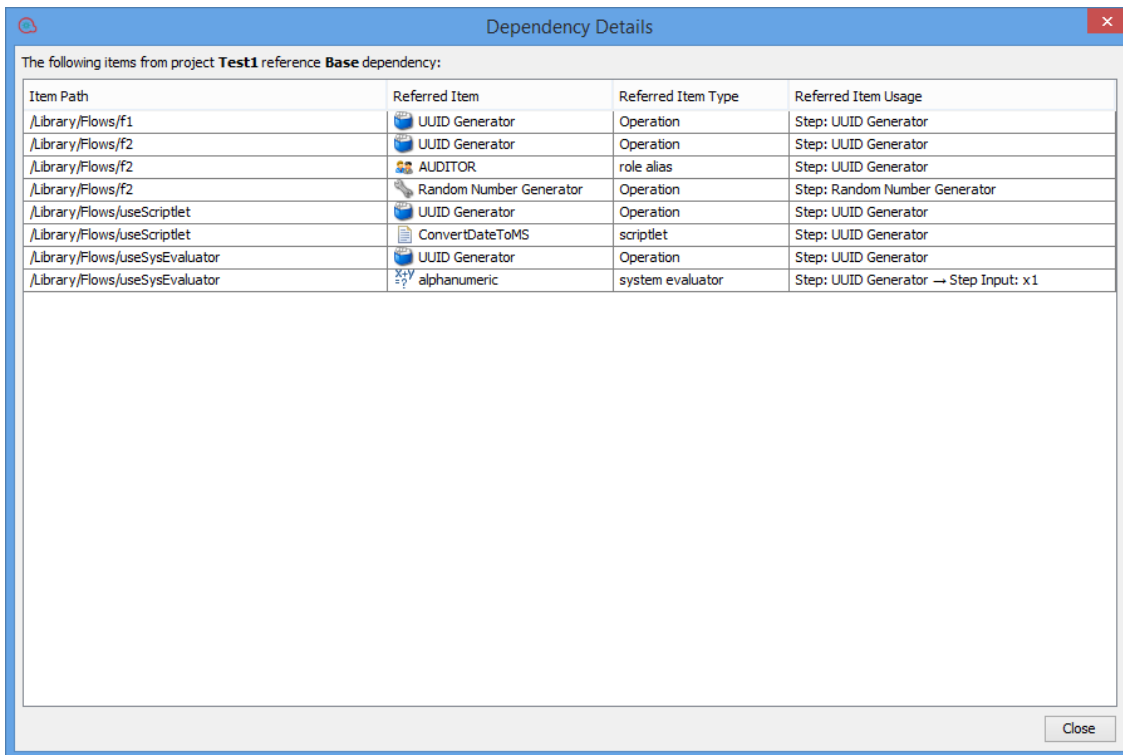
The following screen shows examples of these three types :

Type	Source Type	Name	Description	Location
Warning	Dependencies	test_Responses	Missing 'new_eval_Ref:1.0.0-SNAPSHOT' dependency for project/content pack 'test_Responses'	/test_Responses
Warning	Dependencies	test_Responses	Missing 'Base:1.2.1' dependency for project/content pack 'test_Responses'	/test_Responses
Warning	Dependencies	test1	Missing 'Base:1.2.1' dependency for project/content pack 'test1'	/test1
Warning	Dependencies	test1	Possibly unused 'test2:2.0' dependency for project/content pack 'test1'	/test1
Warning	Dependencies	test1	Missing 'Base:1.2.1' dependency for project/content pack 'test1'	/test1
Error	System Account	sa	The system account name 'sa' is a duplicate of /new_eval_Ref/Configuration/System Accounts/sa...	/test_Responses/Configuration/System Acco...
Error	System Account	sa	The system account name 'sa' is a duplicate of /test_Responses/Configuration/System Accounts/sa...	/new_eval_Ref/Configuration/System Acco...
Error	SystemProperty	sp1	The system property name 'sp1' is a duplicate of /meseriasul/Configuration/System Properties/sp1, /...	/test_Responses/Configuration/System Prop...
Error	SystemProperty	sp1	The system property name 'sp1' is a duplicate of /meseriasul/Configuration/System Properties/sp1, /...	/new_eval_Ref/Configuration/System Proper...
Error	SystemProperty	sp1	The system property name 'sp1' is a duplicate of /test_Responses/Configuration/System Properties/s...	/meseriasul/Configuration/System Propertie...

GUI item	Description
<b>Type</b>	<p>Type of issue: Warning or Error. You cannot debug a flow with errors, but you can edit it.</p> <div style="border: 1px solid gray; padding: 10px; margin: 10px 0;"> <p><b>Best practices:</b></p> <ul style="list-style-type: none"> <li>• It is recommended to fix all issues, warnings and errors.</li> <li>• In multi-author environments, it is important that no warnings are produced when running Validate Dependencies as modifications in the workspace by one author can be also used by another author. Therefore, it is important to have the necessary projects/content packs imported with the correct versions.</li> </ul> </div>
<b>Source Type</b>	The section in which the problem occurs. This may be a flow error, an operation error, a step error, a configuration item error or a dependencies warning.
<b>Name</b>	The name of the section.
<b>Description</b>	A description of the dependency issue.
<b>Location</b>	The location of the item.



## Dependency Details dialog box



GUI item	Description
<b>Item Path</b>	The path of the item in the project that references the dependency.
<b>Referred Item</b>	The name of the item in the project that references the dependency.
<b>Referred Item Type</b>	The type of the item in the project that references the dependency.
<b>Referred Item Usage</b>	The name of the section in the flow in which the item is used.

## Managing Configuration 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.

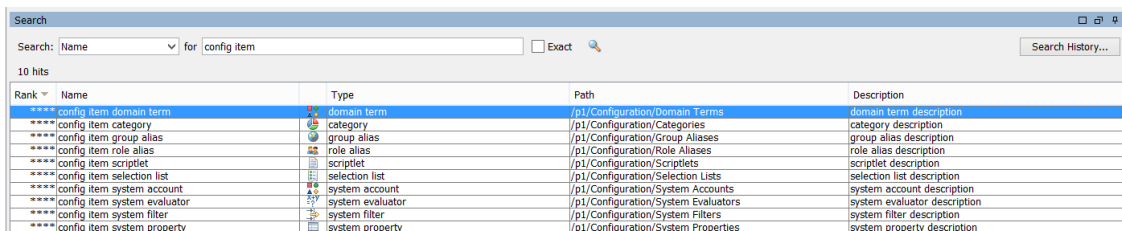
**Important!** If you need to delete, create, or rename a configuration item inside your project, make sure to do this from within Studio, and not by deleting, creating, or renaming the item in the file system.

## Working with Configuration Items

### Searching for Configuration Items

You can search for configuration items by UUID, name, and description fields. The `<all fields>` type search and `<with Lucene query>` type search using name, description and/or UUID as fields inside the search query, will also search through the configuration items that belong to the currently opened projects and content packs.

Using the **Search** pane, you can perform a full-text search throughout the list of open projects and open content packs.



The screenshot shows a search window with the following table of results:

Rank	Name	Type	Path	Description
****	config item domain term	domain term	/p1/Configuration/Domain Terms	domain term description
****	config item category	category	/p1/Configuration/Categories	category description
****	config item group alias	group alias	/p1/Configuration/Group Aliases	group alias description
****	config item role alias	role alias	/p1/Configuration/Role Aliases	role alias description
****	config item scriptlet	scriptlet	/p1/Configuration/Scriptlets	scriptlet description
****	config item selection list	selection list	/p1/Configuration/Selection Lists	selection list description
****	config item system account	system account	/p1/Configuration/System Accounts	system account description
****	config item system evaluator	system evaluator	/p1/Configuration/System Evaluators	system evaluator description
****	config item system filter	system filter	/p1/Configuration/System Filters	system filter description
****	config item system property	system property	/p1/Configuration/System Properties	system property description

See "Search for an operation or configuration item" in "[Finding a Flow or Operation](#)" on page 369 for details on how to search for a configuration item.

## Working with Configuration Item Folders

You can create folders in all the configuration items under the existing configuration item's folder structure. You can perform the following operations on the configuration item folders:

- Create folders in each configuration item section
- Copy and paste configuration Items folders from one project to another

- Move configuration items folders from one project to another
- Move items between folders and the top level section
- Delete configuration items folders.
- Rename configuration items folders
- Set descriptions for each configuration items

## What do you want to do?

### Search for a configuration item using the Navigate to item... option

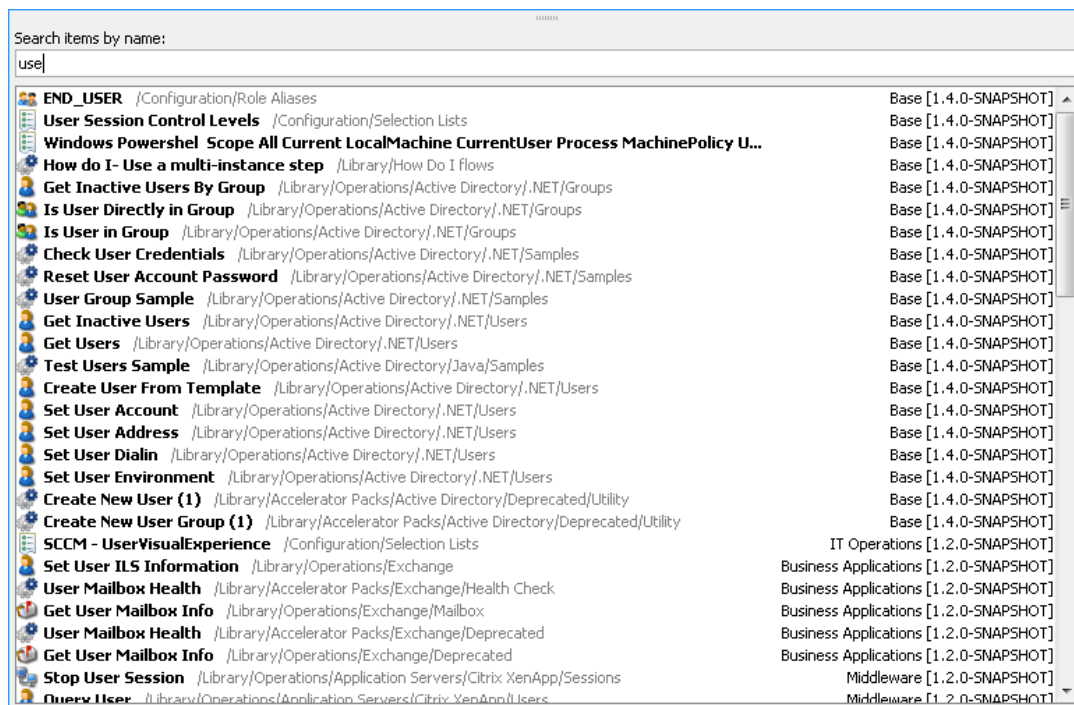
You can use the **Navigate to item..** option to easily find and open a configuration item in your workspace.

1. Select **Tools > Navigate to item...** .

The **Search items by name:** window opens.

2. Type in part of the configuration item name.

The window is immediately populated with search results. For example, if you search for a item whose name contains the string "use", you may see the following results:



3. Use the Up and Down arrow keys to move in the list, then press Enter to open the selected

item. Alternatively, you can open an item with a double-click.

4. You can create steps in the current Flow Editor by dragging and dropping operations or flows from the list.

### Create a configuration item folder

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click on a configuration item folder, and then click **New > Folder**.
3. Enter a name for the new folder and click **OK**.

You can now add, copy or move configuration items into this folder.

### Rename a configuration item/folder

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click on a configuration item folder, and then select **Rename**.
3. Type in a new name and press Enter.

### Copy and paste configuration items from one project to another

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click on a configuration item folder, and select **Edit > Copy**.
3. Select the location for the configuration items in the desired folder and select **Edit > Paste**.

### Move configuration item folders and items

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click on a configuration item folder and select **Edit > Cut**.
3. Select the location for the folder in the second project and select **Edit > Paste**.

The entire folder and its configuration items are moved to the new location.

### Delete a configuration item folder

1. In the **Projects** pane, expand the **Configuration** folder.
2. Right-click on a configuration item folder, and then select **Delete**.

**Note:** Deleting the folder and its configuration items will break any dependencies that they had with other items.

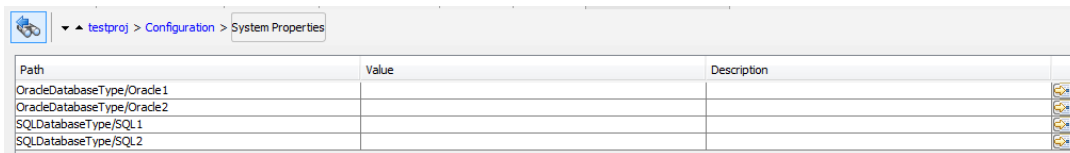
## Set a configuration item folder description

1. Right-click on a configuration item folder and select **Properties**.
2. In the **Description** box, type a description for the configuration item folder.
3. Click **Save**.

## View all the configuration items and their folder structure


1. In the **Projects** pane, right-click on the configuration item folder.
2. Select **Properties**.

A list of all the configuration items along with their folder structure opens. For example, for System Properties the following list opens:



The screenshot shows a dialog box titled 'System Properties' with a breadcrumb path 'testproj > Configuration > System Properties'. Below the title bar is a table with three columns: 'Path', 'Value', and 'Description'. The table contains four rows of configuration items. On the right side of the table, there are four small icons: a magnifying glass, a double arrow, a trash can, and a refresh icon.

Path	Value	Description
OracleDatabaseType/Oracle1		
OracleDatabaseType/Oracle2		
SQLDatabaseType/SQL1		
SQLDatabaseType/SQL2		

**Note:** From this page, you can see the folder structure in read-only mode and cannot edit the names inline. However, you can edit the individual configuration items by double-clicking on a row, or by selecting the  icon on the right of each row.

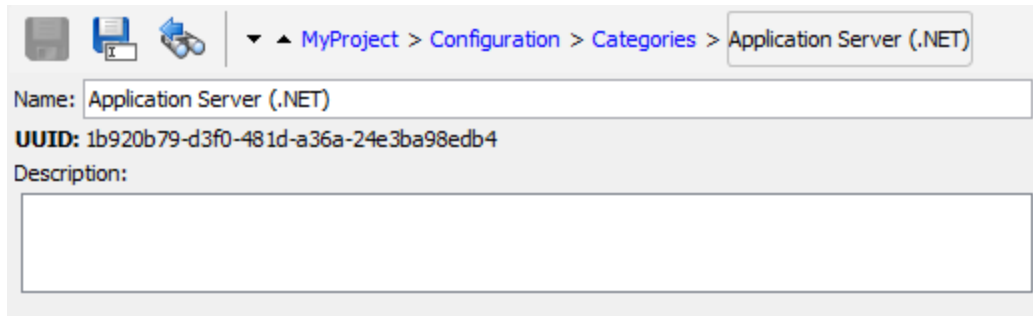
## Configuring Categories

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.



## What do you want to do?

### Create a category

1. In the **Projects** pane, expand the **Configuration** folder.
2. From the **Categories** folder, right-click and select **New > Category**.
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**.

### Copy a category

1. In the **Projects** pane, expand the **Configuration** and **Category** folders.
2. Right-click on the category that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **Category** folder of the project you want to copy to.

5. Right-click and select **Edit > Paste**.

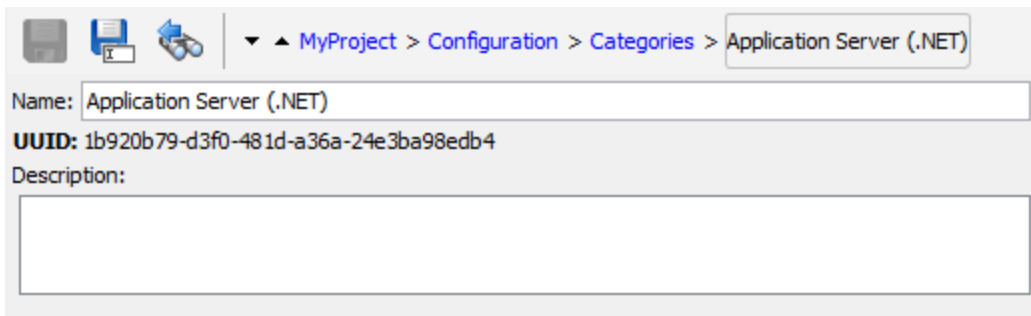
### Delete a category

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 378](#).

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



The screenshot shows the 'Categories editor' window. At the top, there are icons for file operations and a breadcrumb path: 'MyProject > Configuration > Categories > Application Server (.NET)'. Below this, there is a 'Name' field with the text 'Application Server (.NET)'. Underneath the name field is the 'UUID' field with the value '1b920b79-d3f0-481d-a36a-24e3ba98edb4'. Below the UUID is a 'Description' label followed by an empty text area.

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 the configuration editor for a domain term named "Severity". The breadcrumb path is "MyProject1 > Configuration > Domain Terms > Severity". The "Name" field contains "Severity" and the "UUID" is "46f698c7-f7d1-4f68-91d9-13e47a02819a". The "Description" field contains "Information, Warning, Error, Critical." Below the description are "Add" and "Remove" buttons. At the bottom 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. From the **Domain Terms** folder, select **New > Domain Term**.
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.

## Copy a domain term

1. In the **Projects** pane, expand the **Configuration** and **Domain Terms** folders.
2. Right-click on the domain term that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **Domain Terms** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

## 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 378](#).

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*.

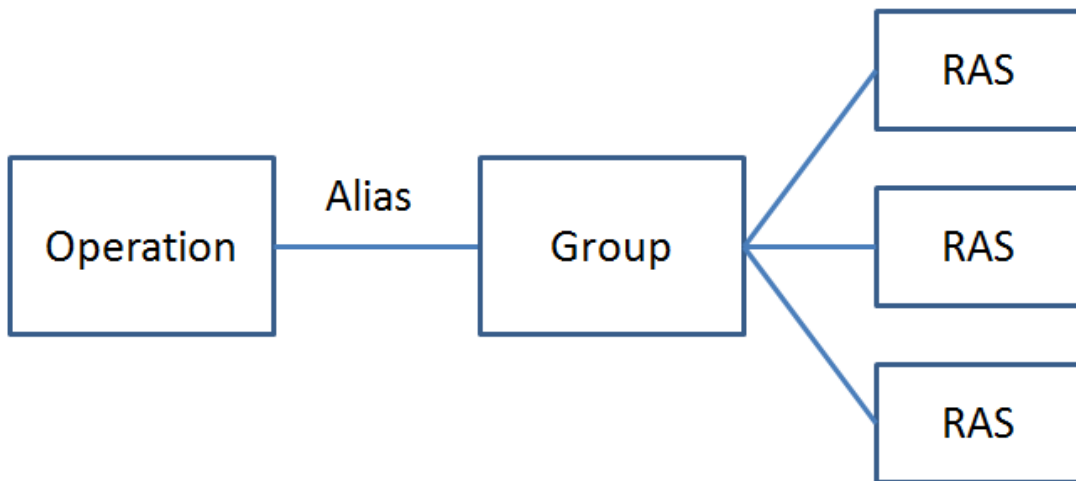
## Group Aliases

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.

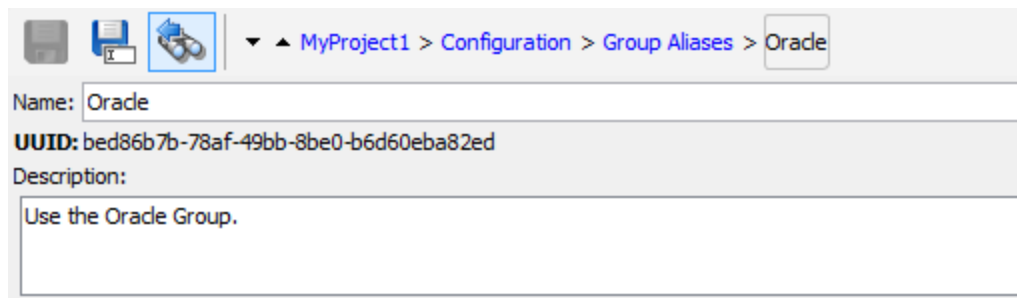
As a fallback, if the group alias is identical to the group name, it is mapped automatically to that group.

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.



## RAS Group Overrides

There are two methods of overriding RAS group details in HP OO:

- Static – as described in ["Group Aliases" on the previous page](#)
- Dynamic - HP OO assigns a value for the input at runtime. This can be done in one of the following ways:
  - The flow author can set the RAS using the **Override Group** field in the Operation fields in Studio's Input inspector. In the **Override Group** field, enter a different group, if you need to override the current group with a different one. This can be a static or a dynamic value. For example, `${overrideJRAS}` where `overrideJRAS` is the name of a variable. During runtime, the variable can automatically be assigned a value using one of the methods described in ["RAS Group Overrides" above](#).

Name: Local Ping

Location: /oo-base-project/Library/Operations/Network/Local Ping

UUID: c460b8c8-e1f7-4321-a72c-6134da73811a

Assign Categories:

Inputs | Outputs | Responses | Description | Scriptlet | Advanced

Operation fields

Group Id: com.hp.oo

Artifact Id: oo-base-legacy-plugin

Version: 1.2.0-SNAPSHOT

Action Name: com.iconclude.content.actions.cmd.ping.LocalPing

Group Alias: /oo-base-project/Configuration/Group Aliases/RAS\_Operator\_Path

Override Group: `${overrideJRAS}`

Input	Required	Type	Assign From	Otherwise
targetHost	<input checked="" type="checkbox"/>	👉	targetHost	Prompt user from the text
packetCount	<input type="checkbox"/>	👉	packetCount	Prompt user from the text
packetSize	<input type="checkbox"/>	👉	packetSize	Prompt user from the text
timeout	<input type="checkbox"/>	👉	timeout	Prompt user from the text
ipVersion	<input type="checkbox"/>	👉	ipVersion	Prompt user from the selection list: IP ...

- From the Scheduler in Central, add an input field that will automatically be assigned during runtime. See "Scheduling Flow Runs" in the *Central User Guide*.
- Add an input field to the flow in Studio. See ["Creating Input" on page 211](#)

### Examples of RAS Group Override Use

- **Example 1:** Defining a dynamic run environment as a variable. This environment may be a test environment or a runtime production environment. This variable can be used as a flow input, step input or set context in Studio or as an input from the Scheduler.
- **Example 2:** Defining a SAP client with RAS. The client can be called from a Central flow as a SAP 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. From the **Group Aliases** folder, right-click and select **New > Group Alias**.
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**.

### Rename a group alias

1. In the **Projects** pane, expand the **Configuration** and **Group Aliases** folders, and double-click the domain term to open its editor.
2. Right-click on the group alias and select **Rename**.
3. Type a new name for the group alias, and press Enter.

### Copy a group alias

1. In the **Projects** pane, expand the **Configuration** and **Group Aliases** folders.
2. Right-click on the group alias that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **Group Aliases** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

### Delete a group alias

1. In the **Projects** pane, expand the **Configuration** and **Group Aliases** folders.


2. Right-click on the group alias to be deleted and select **Delete**.

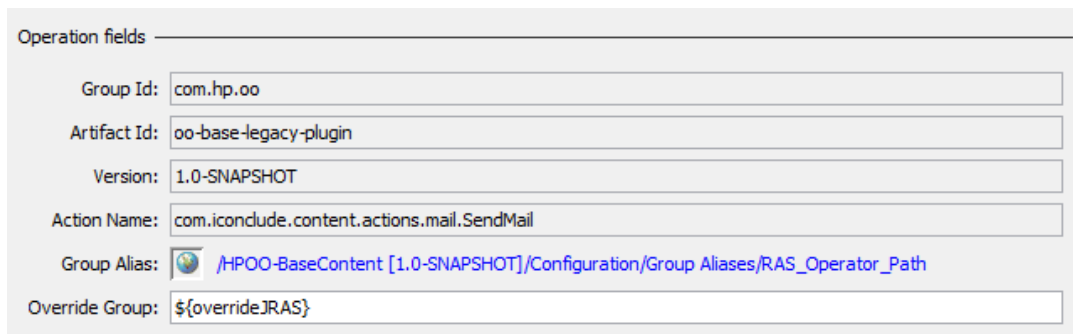
## Map the group alias to a group

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 360](#).
2. In the **Inputs** tab, in the **Operation Fields** section, notice the **Group Alias**  button.




Operation fields

Group Id:

Artifact Id:

Version:

Action Name:


Group Alias:  [/HPOO-BaseContent \[1.0-SNAPSHOT\]/Configuration/Group Aliases/RAS\\_Operator\\_Path](/HPOO-BaseContent [1.0-SNAPSHOT]/Configuration/Group Aliases/RAS_Operator_Path)

Override Group:

3. Go to **Configuration > Group Alias**, select the required group alias, drag it to the appropriate operation and drop it on the **Group Alias** icon (the globe based icon).
4. (Optional) In the **Override Group** box, enter a different group, if you need to override the current group with a different one. This can be a static or a dynamic value. For example, `${overrideJRAS}` where `overrideJRAS` is the name of a variable. During runtime, the variable can automatically be assigned a value using one of the methods described in ["RAS Group Overrides" on page 156](#).
5. Save the operation.

## Remove the assignment of a group alias from an operation

After a group alias has been assigned to an operation, it is possible to remove this assignment.

1. Open an operation to which a group alias was assigned.
2. In the **Inputs** tab, in the **Operation Fields** section, right-click the link next to the **Group Alias**  button.


Operation fields

Group Id:

Artifact Id:

Version:

Action Name:




Group Alias:  [/HPOO-BaseContent \[1.0-SNAPSHOT\]/Configuration/Group Aliases/RAS\\_Operator\\_Path](#)

Override Group:

- From the right-click menu, select **Clear**.

## Reference Material

### Group Alias editor

   ▼ ▲ MyProject1 > Configuration > Group Aliases > Oracle

Name:

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

Description:

GUI item	Description
<b>Name</b>	The name of the group alias.
<b>Description</b>	(Optional) Description of the group alias.

## Configuring Role Aliases

Roles and permissions are assigned to users in Central. However, you can plan for the roles that will be used, by creating role aliases in Studio. For example, you may want to attach a role alias to a gated transition.

When the content pack is deployed to Central, role aliases such as ADMINISTRATOR, EVERYBODY, PROMOTER, SYSTEM\_ADMIN, and END\_USER are mapped to the corresponding roles in Central.

**Note:** Note that some of the role aliases in the Base content pack (AUDITOR, LEVEL\_ONE, LEVEL\_TWO, and LEVEL\_THREE) do not have a corresponding role in Central. These role aliases are considered deprecated.

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

## What do you want to do?

### Create a role alias

1. In the **Projects** pane, expand the **Configuration** folder.
2. From the **Role Aliases** folder, right-click and select **New > Role Alias**.
3. In the dialog box that appears, type a name for the new role alias, and then click **OK**.
4. In the **Description** box, type a description of the new role alias.
5. Click **Save**.

### Rename a role alias

1. In the **Projects** pane, expand the **Configuration** and **Role Aliases** folders.
2. Right-click on the role alias to be renamed and select **Rename**.
3. Type a new name for the role alias, and press Enter.

### Copy a role alias

1. In the **Projects** pane, expand the **Configuration** and **Role Aliases** folders.
2. Right-click on the role alias that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **Role Aliases** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

### Delete a role alias

1. In the **Projects** pane, expand the **Configuration** and **Role Aliases** folders.
2. Right-click on the role alias to be deleted and select **Delete**.



## Reference Material

### Role Alias editor

The screenshot shows the Role Alias editor interface. At the top, there is a title bar with 'ADMINISTRATOR' and a close button. Below the title bar, there are icons for save, refresh, and help, followed by a breadcrumb navigation path: 'Base [2013-06-RC2] > Configuration > Role Aliases > ADMINISTRATOR'. The main form contains the following fields:

- Name:** ADMINISTRATOR
- UUID:** a06bed09-9983-42af-a8ee-c34b30fd3913
- Description:** Represents Operations Orchestration administrators.

GUI item	Description
<b>Name</b>	The name of the role alias.
<b>Description</b>	(Optional) Description of the role alias.

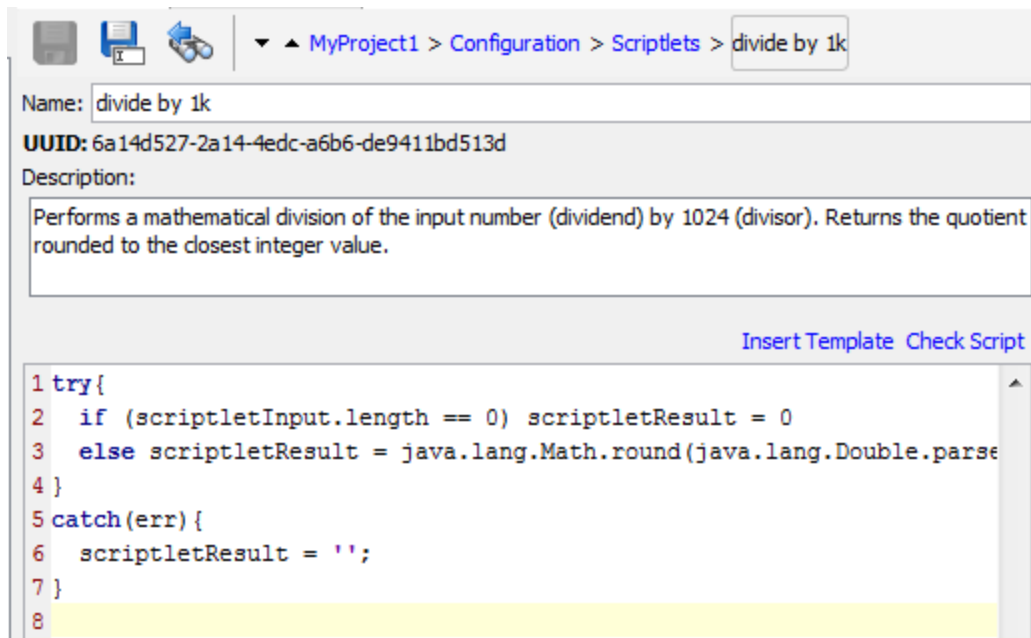
## Configuring Scriptlets

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.

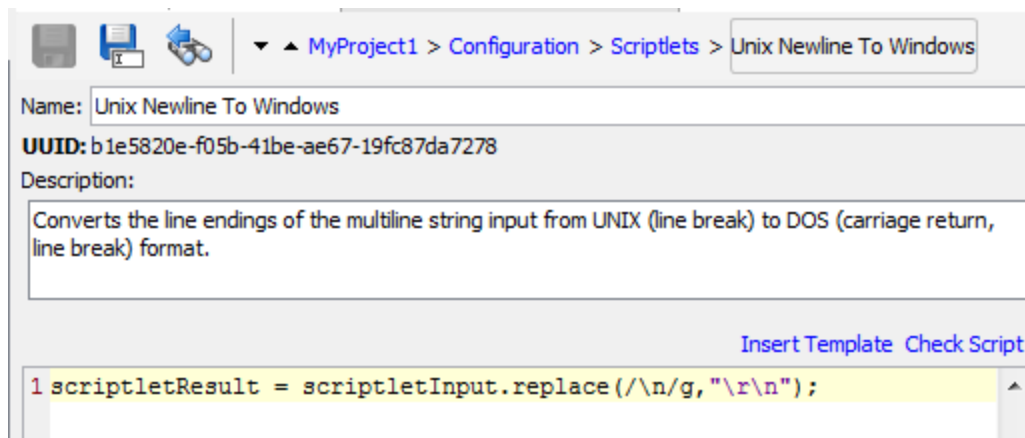
For more information about using scriptlets, see ["Using Scriptlets in a Flow" on page 308](#).

## What do you want to do?

### Create a system scriptlet

1. In the **Projects** pane, expand the **Configuration** folder.
2. From the **Scriptlets** folder, right-click and select **New > Scriptlets**.
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.

**Note:** You can reference a variable that is stored in a folder using the format `${variablename}` and a system property using the format `${path/system_property}`.

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 265](#)
8. Click **Save**.


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

### Save an existing scriptlet as a system scriptlet

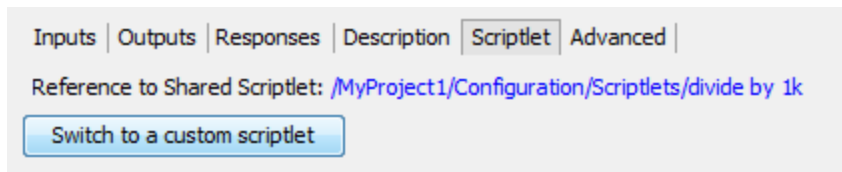
1. Under the **Scriptlet** tab in the **Properties** sheet or Step Inspector, open the scriptlet that you want to save as a system scriptlet.
2. In the **Projects** pane, expand the **Configuration** and **Scriptlets** folders.
3. Drag the **Scriptlet**  icon from the **Scriptlet** tab of the **Properties** 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.



### 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**.

### Copy a scriptlet

1. In the **Projects** pane, expand the **Configuration** and **Scriptlets** folders.
2. Right-click on the scriptlet that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **Scriptlets** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

### Rename a scriptlet

1. In the **Projects** pane, expand the **Configuration** and **Scriptlets** folders.
2. Right-click on the scriptlet to be renamed and select **Rename**.
3. Type a new name for the scriptlet, and press Enter.

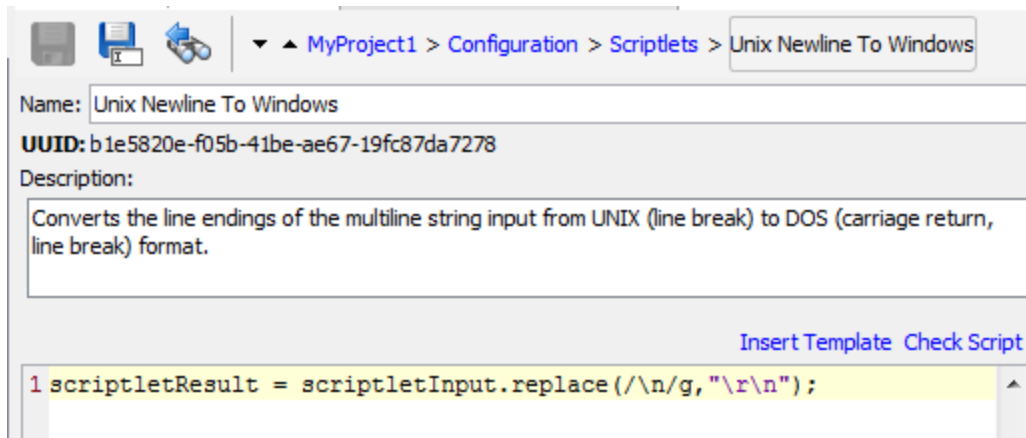
### Delete a system scriptlet

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 378](#).

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



GUI item	Description
<b>Name</b>	The name of the scriptlet.
<b>Description</b>	(Optional) Description of the purpose of the scriptlet.
<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.

### Status Bar

- Displays the line column according to the location of the cursor.
- Displays the status of the **Insert** key, when the Insert mode characters are added. In **Overwrite** mode they are overwritten. Toggle between the two modes using the Insert key.

### Keyword Completion

- While typing a keyword, press **Ctrl** and the **Space bar**, to display a drop-down list of options. Use the up and down arrow keys to navigate through the list and then select the appropriate word. After selecting an item the list disappears and the cursor is placed after the word.

```
1  
2 var sysvalue_password = scriptletContext.get("var_sa_studio_central_default");  
3 scriptletContext.put("field1", sysvalue_password);  
4 v  
5 volatile  
  var  
  void
```

## Jump to Line

- Double-click on the line column item in the status bar or press **Ctrl + G** key. You can only jump to lines inside that have a number marker to the left of the scriptlet window.

```
1  
2 var sysvalue_password = scriptletContext.get("var_sa_studio_central_default");  
3 scriptletContext.put("field1", sysvalue_password);  
4  
5  
6
```

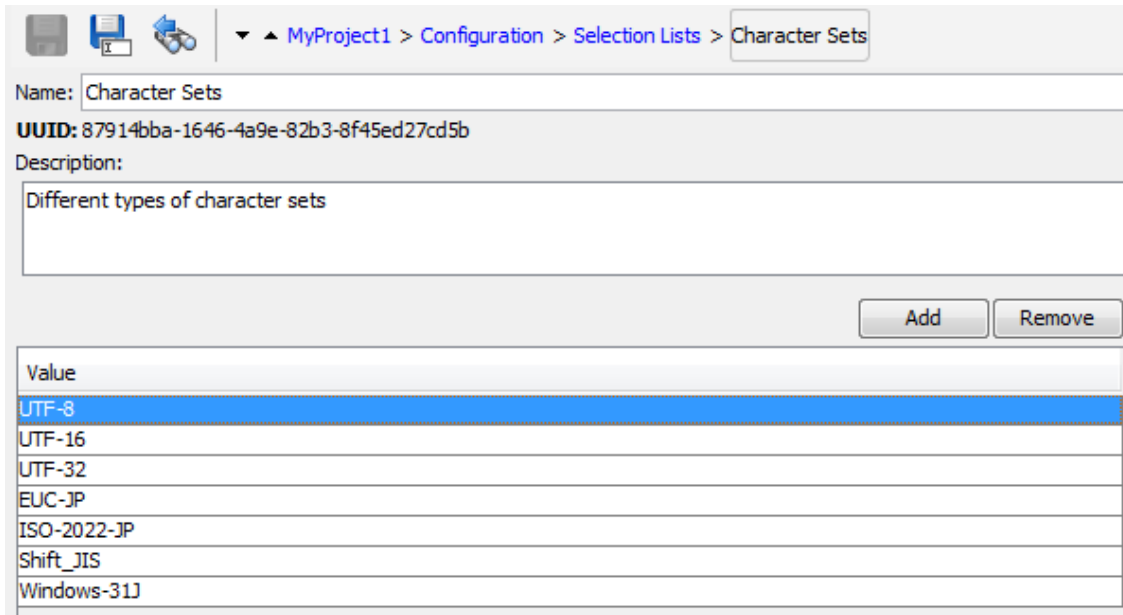
Go to line number:  
  
(Between 1 and 6. Press ENTER to continue)

## Configuring Selection Lists

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

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.



## What do you want to do?

### Create a selection list

1. In the **Projects** pane, expand the **Configuration** folder.
2. From the **Selection Lists** folder, right-click and select **New > Selection List**.
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 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. 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**.

### **Copy a selection list**

1. In the **Projects** pane, expand the **Configuration** and **Selection Lists** folders.
2. Right-click on the selection list that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **Selection Lists** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

### **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 378](#).

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 222.

**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 configuration interface for a system account named "John Citizen". The breadcrumb path is "MyProject1 > Configuration > System Accounts > John Citizen". The "Name" field contains "John Citizen". The "UUID" is "6fd1f200-d435-441f-9efa-925c67c7d9ea". The "Description" field is empty. The "Credentials" section has a "User Name" field containing "<domain>\John\_Citizen" and a "Password" field with masked characters. An "Assign Password" button is visible next to the password field.

## What do you want to do?

### Create a system account

1. In the **Projects** pane, expand the **Configuration** folder.
2. From the **System Accounts** folder, right-click and select **New > System Account**.
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**.

## Copy a system account

1. In the **Projects** pane, expand the **Configuration** and **System Accounts** folders.
2. Right-click on the system account that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **System Accounts** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

**Note:** You can also move/copy system accounts, or a folder that contains system accounts, from one project to another by dragging and dropping.

## 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

The screenshot shows the 'System Accounts editor' for a system account named 'John Citizen'. The breadcrumb navigation is 'MyProject1 > Configuration > System Accounts > John Citizen'. The form includes a 'Name' field with 'John Citizen', a 'UUID' field with '6fd1f200-d435-441f-9efa-925c67c7d9ea', and a 'Description' field. Under the 'Credentials' section, there is a 'User Name' field with '<domain>\John\_Citizen' and a 'Password' field with masked characters. An 'Assign Password' button is located to the right of the password field.

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 Evaluators

System evaluators are string formats that flow authors can use to validate inputs for any data sources except system accounts.

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.

See ["Evaluating Input Data" on page 236](#).

System evaluators can use any of 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 313](#)
- Scriptlets – for more information, see ["Using Scriptlets in a Flow" on page 308](#)

System evaluators are stored in the **Configuration\System Evaluators** folder.

The screenshot shows a configuration window for a system evaluator. The fields are as follows:

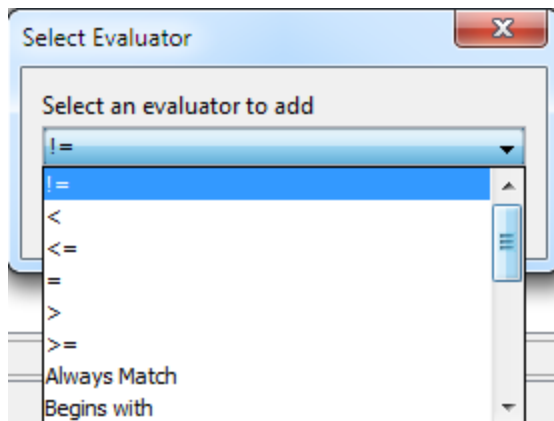
- Name:** MyEvaluator
- UUID:** 167a68dd-82d3-4e91-add4-ca48cdbe4a7d
- Description:** Checks that the input value is less than 1
- Evaluator Type:** < Compare the input against a string or number
- Compare To:** 1
- Test Filter Input:** Pattern Matches (1 occurrence)

At the bottom right of the Test Filter Input section, there are icons for file operations (copy, paste, delete) and buttons for 'Clear' and 'Quick Command'.

## What do you want to do?

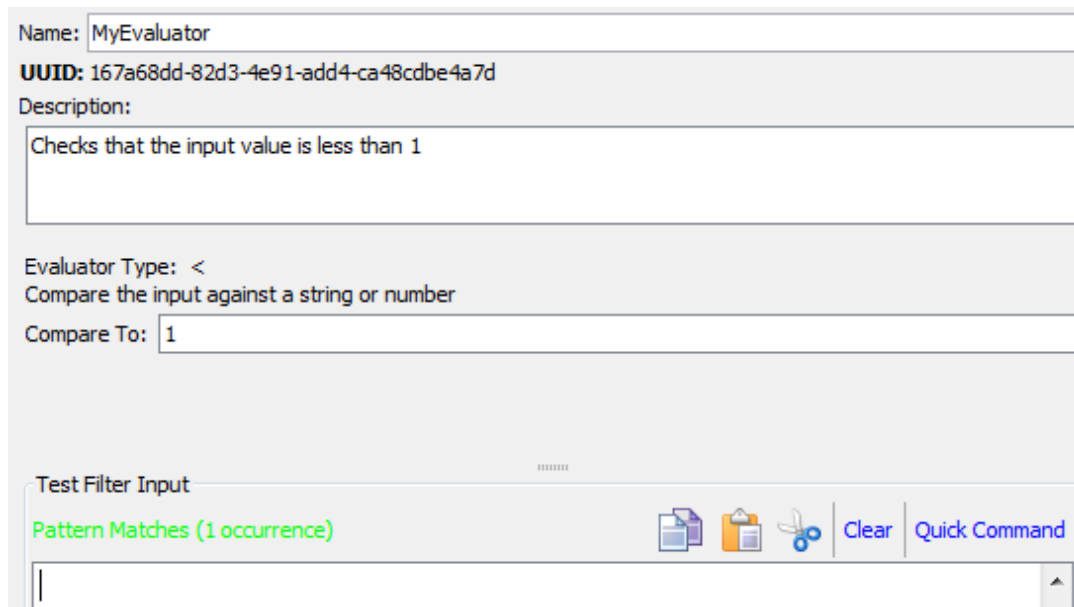
### Create a system evaluator

1. In the **Projects** pane, expand the **Configuration** folder.
2. From the **System Evaluators** folder, right-click and select **New > System Evaluator**.
3. In the Select Evaluator dialog box, select an evaluator type, and then click **OK**.



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

The Evaluator Editor opens. The appearance of the Evaluator Editor varies, depending on the type of evaluator you selected.



5. In the **Description** box, describe the purpose of the evaluator.
6. Enter the text, string, expression value, or scriptlet with which the evaluator should test the input.
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 265](#).

8. Click **Save**.

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

### **Edit a system evaluator**

1. In the **Projects** pane, expand the **Configuration** and **System Evaluators** folders.
2. Double-click the system evaluator that you want to edit.
3. Modify the text, string, expression value, or scriptlet with which the evaluator should test the input.
4. Click **Save**.

### **Copy a system evaluator**

1. In the **Projects** pane, expand the **Configuration** and **System Evaluators** folders.
2. Right-click on the system evaluator that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **System Evaluators** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

### **Delete a system evaluator**

Before you delete a system evaluator, 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 378](#).

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

## Reference Material

### Evaluator Editor – Standard

The appearance of the Evaluator Editor varies, depending on the type of evaluator you selected. If you selected a simple operator in the Select Evaluator dialog box, such as such as =, !=, Begins with, Contains, Match All Words, and Match At Least One Word, the Evaluator Editor appears as follows:

GUI item	Description
<b>Name</b>	Displays the name of the system evaluator
<b>Description</b>	Enter a description of the evaluator
<b>Compare To</b>	Type the text, string, expression value, or scriptlet with which the evaluator should test the input.
<b>Test Filter Input</b>	This is where you place data in order to test that the filter works as expected. detailed information about the GUI items in this section, see <a href="#">"Filtering Output and Results" on page 265.</a>

### Evaluator Editor – Regular Expression

The appearance of the Evaluator Editor varies, depending on the type of evaluator you selected. If you selected **Regular Expression** in the Select Evaluator dialog box, the Evaluator Editor appears as follows:



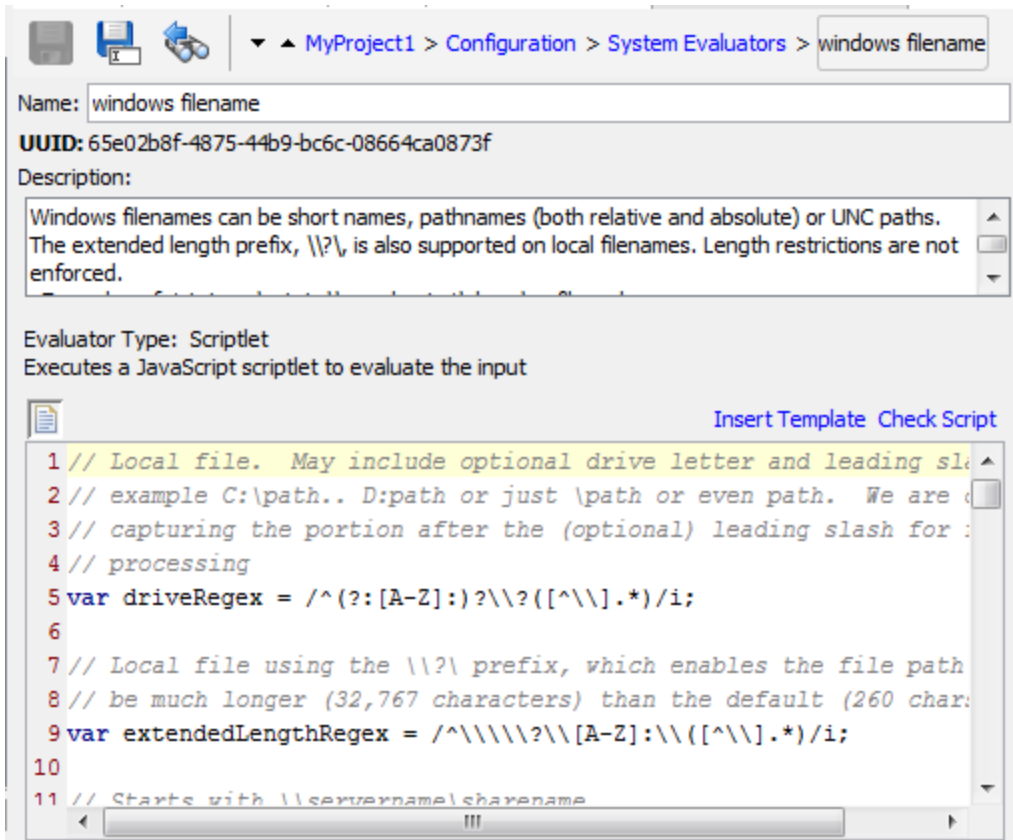
The screenshot shows the configuration editor for a system evaluator named "ip address". The breadcrumb navigation is "MyProject1 > Configuration > System Evaluators > ip address". The "Name" field contains "ip address". The "UUID" is "479bac92-8bac-4ebf-b71d-b300c2f9d8ba". The "Description" field contains the text: "Matches any value that contains a dotted decimal string with 4 groups (IPv4 Address). Examples: 127.0.0.1, 0.0.0.0, 255.255.255.01". The "Evaluator Type" is "Regular Expression". Below this, it says "Matches the input string against a regular expression. See the help documentation for a detailed description of regular expression types". The "Regular expression type" is set to "Java Style". The "Expression value" field contains the regular expression: "((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?),){3}(25[0-5]|2[0-4][0-9]|[01]?". The "Match the whole input" checkbox is checked. The "Multi-line" and "Ignore case" checkboxes are unchecked.

For information about creating regular expressions, see ["Using Regular Expressions in a Flow" on page 313](#).


GUI item	Description
<b>Name</b>	Displays the name of the system evaluator
<b>Description</b>	Enter a description of the evaluator
<b>Expression Type</b>	Select <b>Java Style</b> as the type of regular expression. Do not use the other styles; they have been deprecated.
<b>Expression Value</b>	Type the regular expression.
<b>Match the whole input</b>	Select to apply the evaluator to the entire input.
<b>Multi-line</b>	Select to enable multiple lines in an expression.
<b>Ignore Case</b>	Select to make the regular expression not case-sensitive.

### Evaluator Editor – Scriptlet

The appearance of the Evaluator Editor varies, depending on the type of evaluator you selected. If you selected **Scriptlet** in the Select Evaluator dialog box, the Evaluator Editor appears as follows:



For information about creating scriptlets, see ["Using Scriptlets in a Flow"](#) on page 308.

GUI item	Description
<b>Name</b>	Displays the name of the system evaluator.
<b>Description</b>	Enter a description of the evaluator.
<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.

## 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.

Name: TableFilter  
UUID: ba2eacc5-f1f3-4c0b-ad25-4d1e97d9a5a3  
Description:

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: [dropdown]  
Select Row: 0 Select Col: 3  
Select Width: 1 Select Height: 1

If checked, interpret the first row as column headers

Test Filter

Test Filter Input

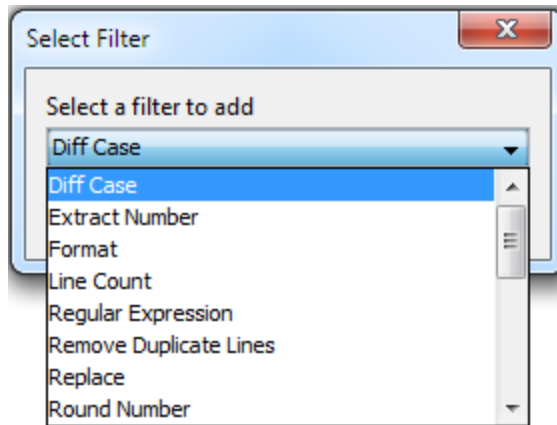
Clear Quick Command

## What do you want to do?

### Create a system filter

1. In the **Projects** pane, expand the **Configuration** folder.

2. From the **System Filters** folder, right-click and select **New > System Filter**.
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.

Name: TableFilter  
UUID: ba2eacc5-f1f3-4c0b-ad25-4d1e97d9a5a3  
Description:

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: Ascending  
Select Row: 0 Select Col: 3  
Select Width: 1 Select Height: 1

If checked, interpret the first row as column headers

Test Filter

Test Filter Input

Clear Quick Command

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 265](#).
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 265](#).
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.

**Troubleshooting:** If you are using a localized Windows command line, after using Quick Command, you might get some strange characters in the results, because of the encoding of the command line.

To avoid this:

1. Close Studio.
2. Open `<Install_folder>/studio/Studio.I4j.ini`.
3. Add the following:

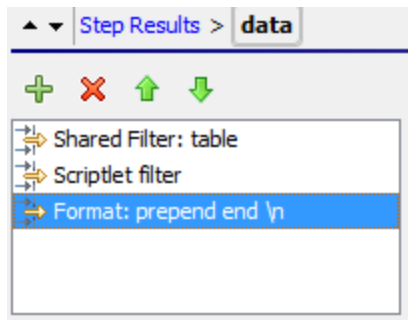
```
-Ddharma.windows.cmd.unicode=true
```

This will ensure that the characters read from the command line are interpreted in unicode.

Note that using this parameter might cause performance problems if you have multiple runs of flows containing operations that use the command line. Remove it from the **Studio.I4j.ini** file after you finish testing the filters.

## 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.

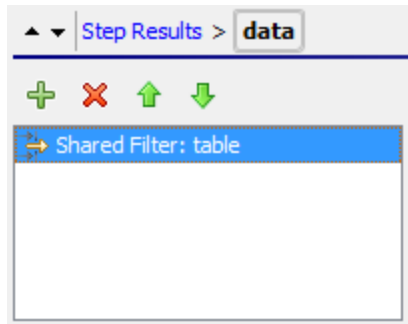


4. To rename the new system filter, right-click it, click **Rename**, and change its name.

## 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.



### 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**.

### Copy a system filter

1. In the **Projects** pane, expand the **Configuration** and **System Filters** folders.
2. Right-click on the system filter that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **System Filters** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

### Delete a system filter

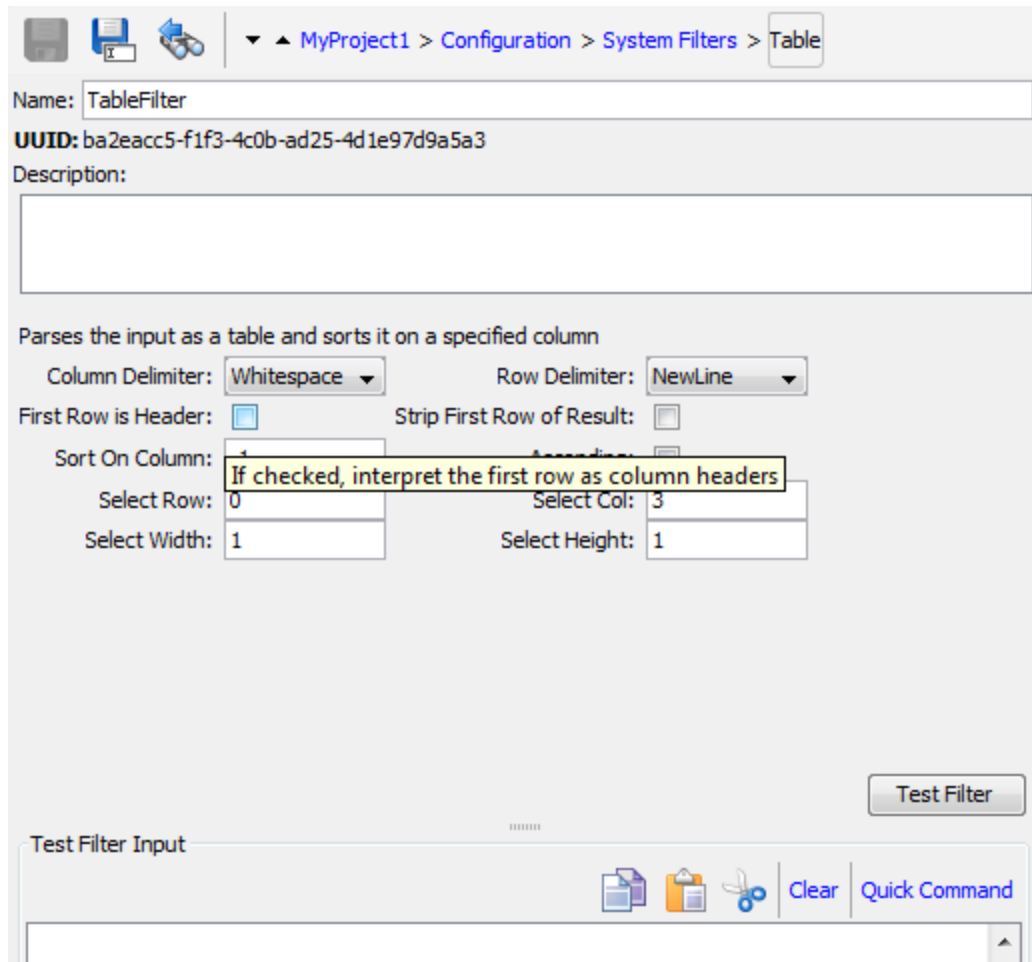
Before you delete a system filter, it is recommended to use the **What Uses This** function to check that other items do not depend on it. For more information, see ["Finding Out How Flows and Operations are Used" on page 378](#).

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 265.



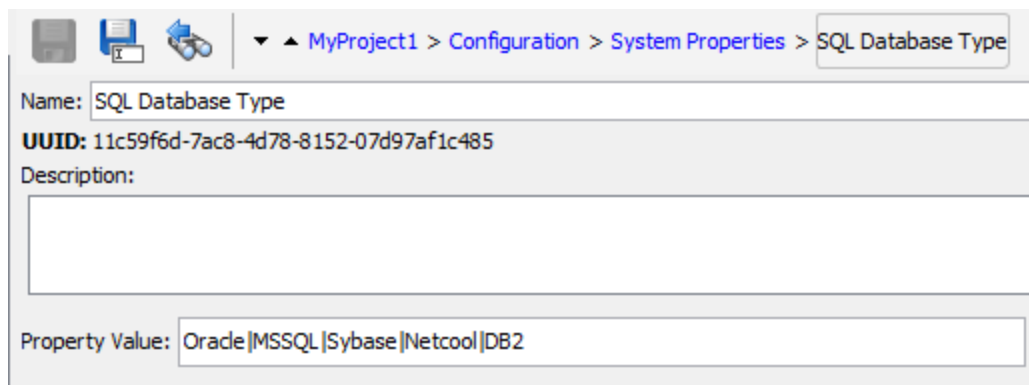
GUI item	Description
Name	Displays the name of the system filter
Description	Enter a description of the system filter

## 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.



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.

**Note:** When dynamically referencing a system property, you must use the complete path. For example, if there is a system property named **domainName** under the following folder structure **staging/domain**, use the string **\$(staging/domain)** to reference it:

Name: domainName Input Type: Single Value

Assign from Variable: <not assigned>

Otherwise:

Use Constant  Prompt User

Constant Value:  
\${staging/domain}

## What do you want to do?

### Create a system property

1. In the **Projects** pane, expand the **Configuration** folder.
2. From the **System Properties** folder, right-click and select **New > System Property**.
3. Enter a name for the system property and click **OK**.

The System Properties Editor opens.

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

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 308](#).

## 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.

## Rename a system property

1. In the **Projects** pane, expand the **Configuration** and **System Properties** folders.
2. Right-click on the system property to be renamed and select **Rename**.
3. Type a new name for the system property, and press Enter.

## Copy a system property

1. In the **Projects** pane, expand the **Configuration** and **System Properties** folders.
2. Right-click on the system property that you want to copy.
3. Select **Edit > Copy**.
4. Move to the **System Properties** folder of the project you want to copy to.
5. Right-click and select **Edit > Paste**.

## 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 378](#).

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

The screenshot shows the System Properties Editor interface. At the top, there is a breadcrumb navigation path: MyProject1 > Configuration > System Properties > SQL Database Type. Below this, the 'Name' field contains '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'.

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 292](#).

## 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 ["Authoring Best Practices" on page 26](#).

### 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 192](#).

## 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 196](#).

## 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 288](#).


## 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 237](#).

For more information about setting the responses for an operation, see ["Setting Responses" on page 242](#).

### Step 5: Add inputs to the flow


1. Right-click on the flow and select **Properties**.
2. In the Operation Editor, click the **Inputs** tab.
3. Click **Add input**  from the Inputs toolbar 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**  - to run an operation against multiple targets
5. In the **Assign From** field, define the data source for the input.

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 211](#).

### 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**  from the Step Inspector toolbar.
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.

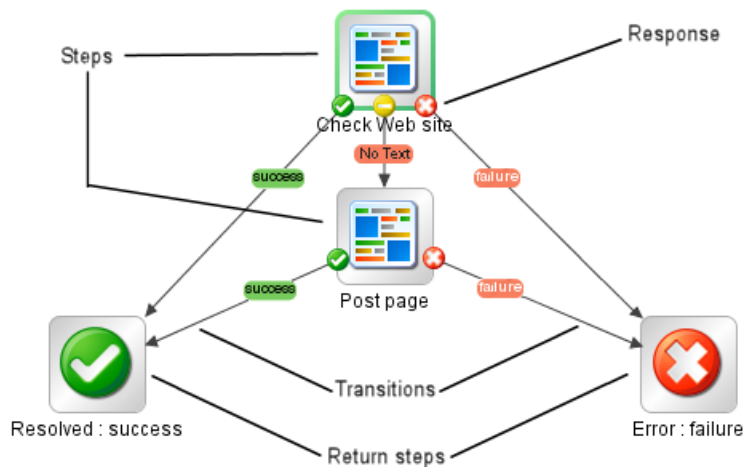
For more information and options, see ["Setting Step Results" on page 256](#).

### 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

**Important!** If you need to delete, create, or rename a flow inside your project, make sure to do this from within Studio, and not by deleting, creating, or renaming the item in the file system.



## **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 ["Authoring Best Practices" on page 26](#).

**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 380](#).

## 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.

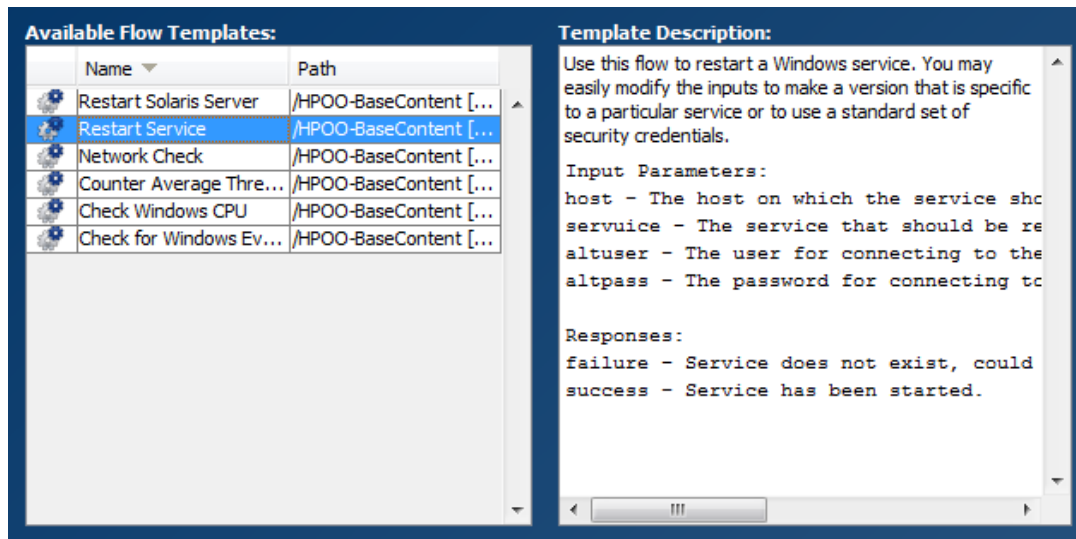
## Create a flow from a template


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.

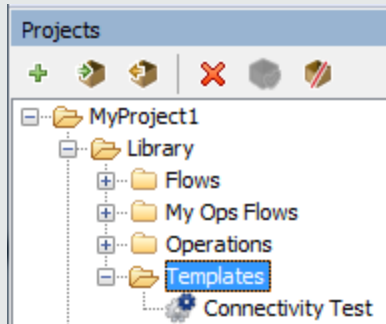
3. In the list of templates that appears, highlight a flow template to display its description. If required, drag the scroll bar to display the text.



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 **Dependencies** 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.



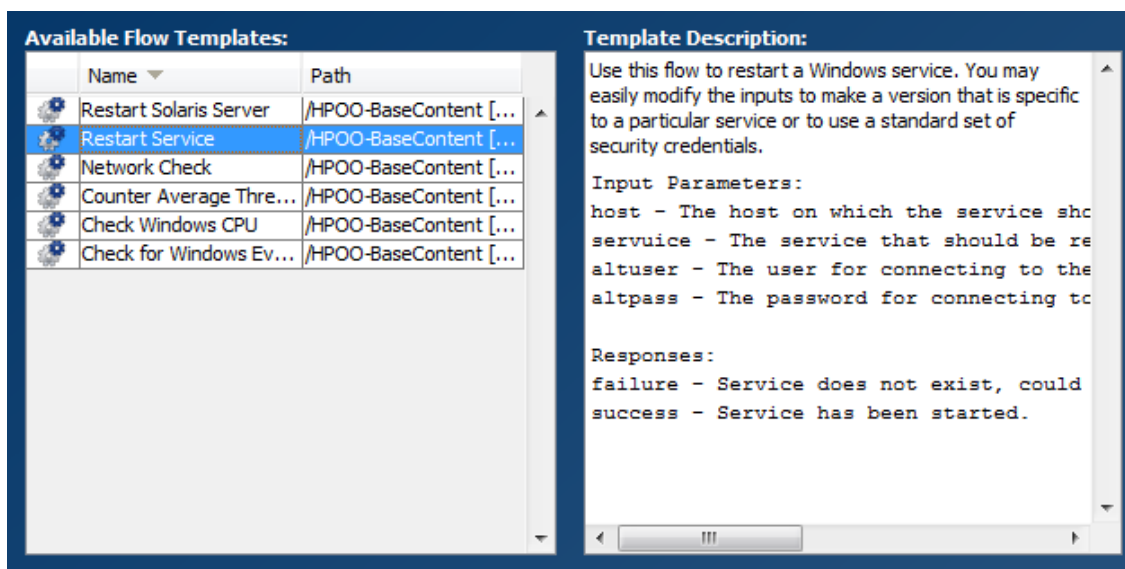
### Add a description of the flow

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 **Dependencies** pane, you will be able to modify the step, but note that the operation in the **Dependencies** pane is read-only.
- If you want to modify an operation before creating the step, copy it from the **Dependencies** 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 ["Authoring Best Practices" on page 26](#).

**Note:** You can use the Generate Documentation feature to make the information in the description available to authors and Management users. For more information, see ["Generating Documentation about Flow and Operations" on page 380](#).

- 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 **Dependencies** pane, select the operation that you want to add to the flow.

**Note:** The operations in the **Dependencies** 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 205](#).

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

1. In the **Dependencies** 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 360.

## 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**.

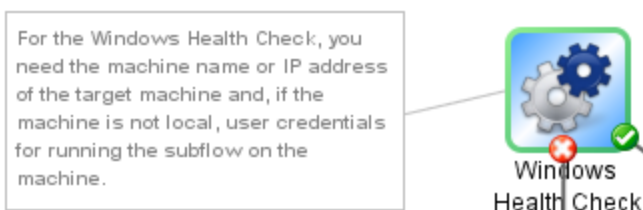
For best practices for writing a description, see "[Authoring Best Practices](#)" on page 26.




## 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 292](#).

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 non-blocking step

A non-blocking step does not block the rest of the flow. While it is running, the flow run continues to carry out the steps that come after it.


You would do this when a flow's subsequent actions do not depend on the outcome of the step. For example, you want a flow to send a notification after the failure response of a step. But you don't need the flow to wait while the notification is being sent.

**Note:** In a flat flow or subflow that contains a non-blocking step, the flow run will not end until the non-blocking step is finished. However, if it is part of a flow with multi-instance or parallel

steps, the flow run will not wait for the non-blocking step.

Note also that it is not possible for multi-instance or parallel split steps to be non-blocking.

1. With the flow open on the authoring canvas, right-click the step, and then click **Toggle Nonblocking**.

An orange lightning bolt appears on the step's icon, and the step automatically acquires a single response, **done** .



2. Connect the **done**  response to the next step.

**Note:** To change a non-blocking step back into a regular step, right-click the step and click **Toggle Nonblocking** again.

## Create a manual step

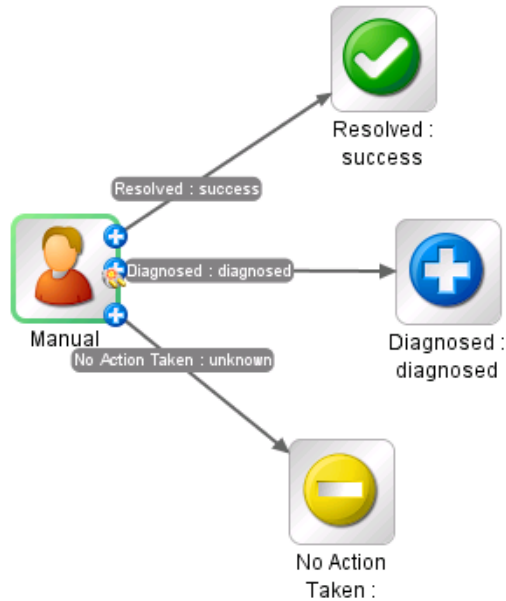
A manual step is one that offers a choice of actions. The user will need to select an action at runtime.

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 **Dependencies** 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 **Dependencies** 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 (up to 128 characters).
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.


The screenshot shows the 'Step Inspector' interface for a step named 'SQL Query'. The 'Display' tab is selected, showing a checkbox for 'Always prompt user before executing this step' which is currently unchecked. Below this are input fields for 'Prompt Title', 'Prompt Width' (set to 0), and 'Height' (set to 0). A large text area for 'Prompt Text' is also visible.

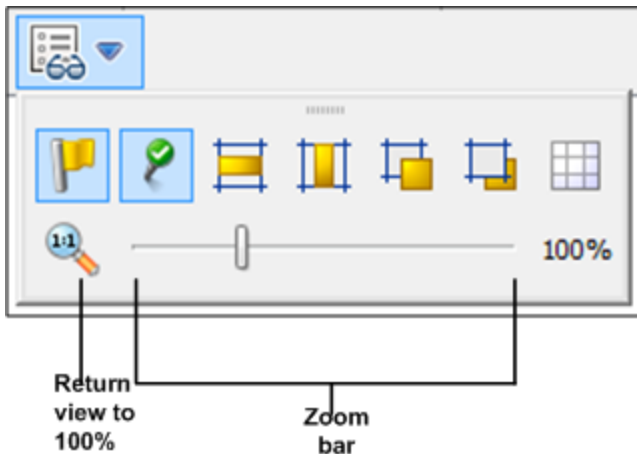
GUI item	Description
<b>Always prompt user before executing this step</b>	Select the checkbox if you want the prompt window to appear every time this step is run.

<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.



**Note:** By default, the grid is not displayed in the authoring pane. If you set the grid to display, using the **Show/Hide Grid** button , this state is not persistent after Studio is closed. To change the default behavior, so that the grid is displayed by default, you can open the **Studio.properties** file and set the property `dharma.studio.ui.activegrid=true`.

## 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

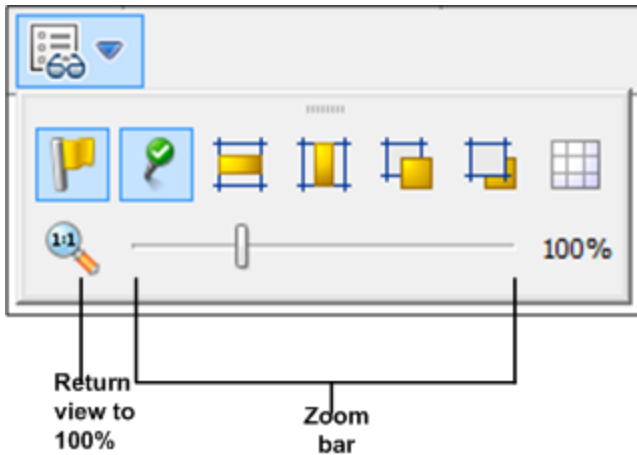
Flows may include objects that are stacked together in the authoring pane. This may occur in long flows where there are many items in the flow. In such cases, you need to bring the most important objects to the front of the stack.








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
<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 375.

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?

### Open a flow for editing

- Double-click a flow in the **Project** pane to open it for editing in the authoring pane.
- To open multiple flows in the authoring pane, select them using the SHIFT or CONTROL keys, right-click, and select **Open**.

### Jump to a step to edit it

If you have a complex flow with many steps, use the **Go to Step**  button to quickly jump to the step that you need to edit.

1. Open the flow in the authoring pane.
2. In the **Authoring pane** toolbar, click **Go to Step** .
3. Type the name of the step to jump to it, or the first letters of the step to select it from a list.
4. If desired, use the **Up** and **Down** arrows on your keyboard to navigate through the list of steps.

## 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.

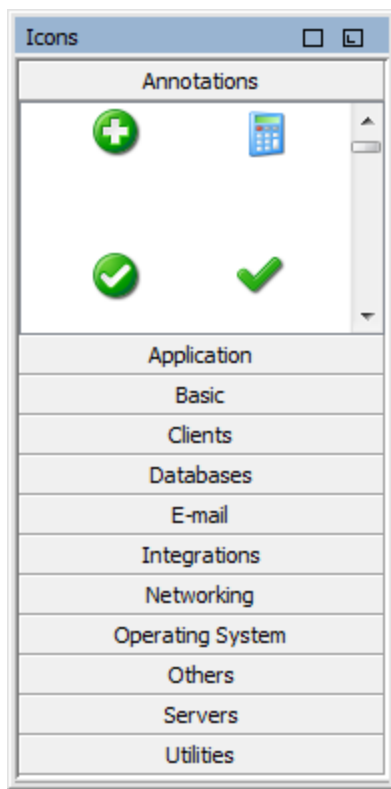
## Move a group of steps in the flow diagram

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 211](#).
  - Add or edit results in the step. For details, see ["Setting Operation Outputs" on page 252](#).



- Add or edit a description of the step. For details, see ["Creating Steps in a Flow" on page 196](#).
- 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 308](#).

**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 (up to 128 characters).
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 'Display' tab of the Step Inspector for a step named 'SQL Query'. The interface includes a 'Step Name' field with 'SQL Query' entered. Below this are tabs for 'Inputs', 'Results', 'Display' (selected), 'Description', 'Advanced', and 'Scriptlet'. A checkbox labeled 'Always prompt user before executing this step' is present and unchecked. Below the checkbox are fields for 'Prompt Title', 'Prompt Width: 0', and 'Height: 0'. At the bottom is a large text area labeled 'Prompt Text'.

GUI item	Description
<b>Always prompt user before executing this step</b>	Select the check box 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.

**Note:**

- An input that appears in red is a step input coming from a required operation input that was deleted for that step. For example:

queryValuesList	<input type="checkbox"/>		queryValuesList	Prompt user fr...	queryValuesList
delimiter	<input type="checkbox"/>		delimiter	Prompt user fr...	delimiter
url	<input checked="" type="checkbox"/>		url	Prompt user fr...	url

- An input that appears in gray is a step input coming from an optional operation input that was deleted for that step.
- You can resize each pane by dragging the divider bar between the two panes. Use the scroll bars at the bottom of each pane to show the unseen fields. The next time you open Studio, the divider bar is in the same position.
- There is always one input highlighted in the Inputs pane. After deleting an input, the first input in the Inputs pane is automatically highlighted.
- You can move columns in the Input pane by dragging and dropping.
- Inputs are processed according to their position in the inputs list, from top to bottom.

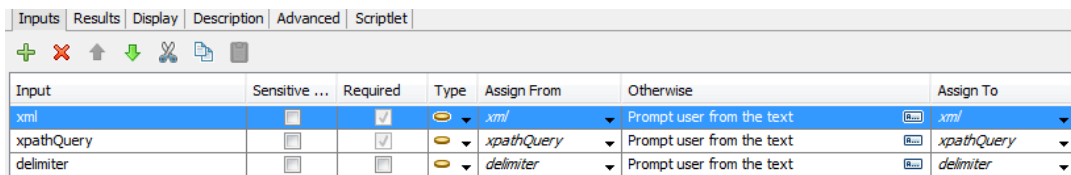
## ***Best Practices***

- Define all possible inputs in the flow description, while differentiating between optional and required inputs.
- If you mark an input as **Required** in an operation or subflow, the input will behave as a required input in all flows that use the operation/subflow as a step.
- Delete optional inputs in steps, if these are not required.
- After you mark an input as **Sensitive**, it remains hidden during the whole run. The input passes this behavior to every input/step result that is assigned from it.
- 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 Input 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.





The screenshot shows the 'Inputs' tab in the Input Inspector. The table has columns: Input, Sensitive, Required, Type, Assign From, Otherwise, and Assign To. The 'xml' row is selected.

Input	Sensitive ...	Required	Type	Assign From	Otherwise	Assign To
xml	<input type="checkbox"/>	<input checked="" type="checkbox"/>	☰	xml	Prompt user from the text	xml
xpathQuery	<input type="checkbox"/>	<input checked="" type="checkbox"/>	☰	xpathQuery	Prompt user from the text	xpathQuery
delimiter	<input type="checkbox"/>	<input type="checkbox"/>	☰	delimiter	Prompt user from the text	delimiter

3. If there are existing inputs, select the row that you want the new input to follow. For example, if you select the first row, the new input will appear on the second row, and will be the second item in the list.
4. Click the  button.
5. 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.

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

- **Single Value** 
- **List of Values**  This enables you to run an operation with an input that contains multiple values. The values must be provided in a list separated by the delimiter specified in the **Input Delimiter** field.

8. Specify the input source in the Input Inspector:


- To assign the value from a variable with the same name as the input, select the default name (marked by the prefix *<default>*) that appears at the top of the **Assign From** dropdown.
- To assign the value from a different flow variable, select the variable name in the **Assign From** dropdown.
- If you do not want to define a source for this input, select *<not assigned>* in the **Assign From** dropdown.

For a detailed description of how to define different input source types, see ["Specifying the Input Source " on page 222.](#)

**Note:** When you create a new input for an operation, flow or step, the default value for **Assign From/Assign To** is automatically set to **<not-assigned>** when installing Studio 10.50. This functionality can be changed using the properties **dharma.studio.ui.inputinspector.assignfrom.selected** or **dharma.studio.ui.inputinspector.assignto.selected** in the **Studio.properties** file.

This default also applies to the inputs of a new step, and to a step that was dragged from Projects/Dependencies Pane.

For details on the parameters in the Studio.properties file, see ["Configuring Studio Properties " on page 393.](#)

9. In the **Otherwise** list, you select the action to occur if the flow variable that you specified in the **Assign from** box does not exist or has no value stored in it. When you click on an input name, the right pane is automatically updated with its details. Click  to open the Otherwise dialog box or fill in the fields in the right pane. The options are:
  - **Use Constant:** Enter the constant value that will be used for the input. For example, an IP address that is always used.
  - **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.
    - **Prompt For: Text** - Prompt the user to enter information. Type the prompt that the user will see in the **User Message:** box. For example, Enter a value for eventId.
    - **Prompt For: Selection** Prompt the user to select a value. Select the type of list the user will select from:
      - Selection List:** Select the selection list from the **Named** dropdown list. All the selection lists are shown.

**Domain Term:** Select the domain term from the **Named** dropdown list. All the domain terms are shown.

**Flow Variable:** Select the name of a flow variable . Flow variables are defined in the results for flows. Type a delimiter for the flow variables in the **Source Delimiter** field.

**Note:** It is not recommended to use the **Prompt User** option for a step in the middle of a flow.

- **Use Previous Step Result:** Select a previous step result to be used if the **Assign From** variable for the input does not exist or has no value.

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.

- **System Account:** Select the system account name and credential type.
- **Logged-in user credentials:** Select the credential type of the logged-in user.

**Note:** The **System Account** and **Logged-in user credentials** fields do not appear if you define the input as **List of Values** in the **Type** field.

For more details about different types of input source, select the relevant task in "[Specifying the Input Source](#) " on page 222.

The default value that appears in this field depends on the settings in the **studio.properties** file. For details on these settings, see the parameter **dharma.studio.ui.inputinspector.assignfrom.selected** in "[Configuring Studio Properties](#) " on page 393.

**Note:** When you create a new input for an operation, flow or step, the default value for **Otherwise** is automatically set to **Use Constant**.

10. Specify the input target in the Input Editor:

- To assign the value to a variable with the same name as the input, select the default name (marked by the prefix *<default>*) that appears at the top of the **Assign To** dropdown.
- To assign the value to a different flow variable, select the variable name in the **Assign To** dropdown.

- If you do not want to define a target for this input, select *<not assigned>* in the **Assign To** dropdown.
11. 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** field on the right of the window.

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.

12. (Optional) To validate that the input is in the correct format, select a system evaluator from the **Validation Format** list.

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. For information about creating evaluators, see ["Evaluating Input Data" on page 236](#).

For example, if your flow requires users to enter an email address, you can use an evaluator to check that the input is entered with the correct email format. For information about creating evaluators, see ["Evaluating Input Data" on page 236](#).

13. (Optional) To record the value, to make it available for diagnostics or auditing, select **<run history>** from the **Record Under** list.
  - **<run history>**
  - one of the domain items in the list


## Rename an input

1. Open the **Properties** sheet (for a flow or operation) or the Input Inspector (for a step).
2. On the **Inputs** tab, double-click on an input name, or press **F2** and type in a new name.

or:

In the right pane, type in a new input name in the **Name** field.

## Remove an input

1. Open the **Properties** sheet (for a flow or operation) or the Input Inspector (for a step).
2. On the **Inputs** tab, select the input that you want to remove, and then click  **Remove Input** from the Input Inspector toolbar. A removed step input appears in italics. You can select multiple inputs using the **Ctrl** and **Shift** keys.

## 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. You can edit any of the input's fields as required.
4. Click **OK**.


The red/gray color is removed and the input is shown in black.

## Reference Material

### Input Inspector > Toolbar



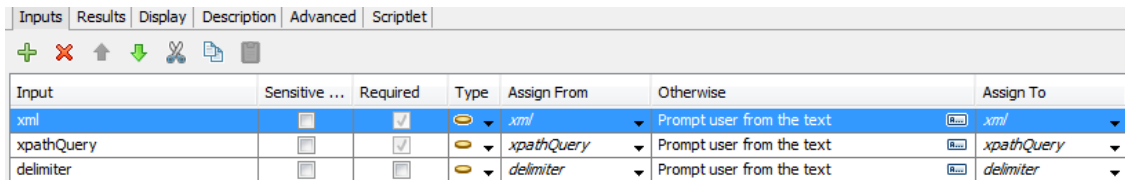
GUI item	Description
<b>Add input</b>	Adds a new input row. Type the name of the new input and click OK. The new input's attributes are assigned with default values.
<b>Delete input</b>	Deletes the selected input row.
<b>Move up</b>	Moves the selected input row higher in the list.
<b>Move down</b>	Moves the selected input row lower in the list.
<b>Cut</b>	Removes the selected input row from its current location. Use the <b>Paste</b> option to place the row in a new location.  <div style="background-color: #f0f0f0; padding: 10px;"> <p><b>Note:</b> If you try to remove an operation input at the step level, Studio automatically makes it appear in italics and moves it to the bottom of the inputs table. Required inputs appear in red, and optional inputs in gray.</p> <p>You can restore the input by changing the <b>Type</b>, <b>Assign To/Assign From</b> or <b>Otherwise</b> fields.</p> </div>
<b>Copy</b>	Copies the selected input row. Use the <b>Paste</b> option to place the copy in a new location.

<b>Paste</b> 	<p>Pastes the copied/cut input row at the current location. If the input name already exists, a message appears and the input is not pasted. (This is the default behavior.)</p> <p>If the input name already exists, Studio automatically names the new input <b>&lt;input&gt; (Copy 1)</b>.</p>
--	---

**Note:** You can also cut, copy and paste multiple inputs between steps, flows or operations. Select additional inputs by holding down the **Ctrl** key and clicking on the row.





### Input 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.



Input	Sensitive ...	Required	Type	Assign From	Otherwise	Assign To
xml	<input type="checkbox"/>	<input checked="" type="checkbox"/>	xml	xml	Prompt user from the text	xml
xpathQuery	<input type="checkbox"/>	<input checked="" type="checkbox"/>	xpathQuery	xpathQuery	Prompt user from the text	xpathQuery
delimiter	<input type="checkbox"/>	<input type="checkbox"/>	delimiter	delimiter	Prompt user from the text	delimiter

GUI item	Description
<b>Input</b>	Displays the name of the input.
<b>Sensitive</b>	<p>Makes this input hidden in the Central user interface and in the Studio internal and remote debugger. If this check box is selected, the input value will be encrypted in Central and Studio, will be displayed as asterisks and will not be persisted</p> <p>This flag is transient, i.e., assignment from a sensitive variable will also make the assigned variable sensitive. This transitivity is only seen during runtime execution, and is not reflected in the Studio user interface.</p> <p>When used in scriptlets, sensitive data is retrieved as encrypted.</p> <p>Sensitivity is also retained when using expressions for assignment. For example, <code>\${input1}</code>.</p>
<b>Required</b>	Makes this input mandatory.

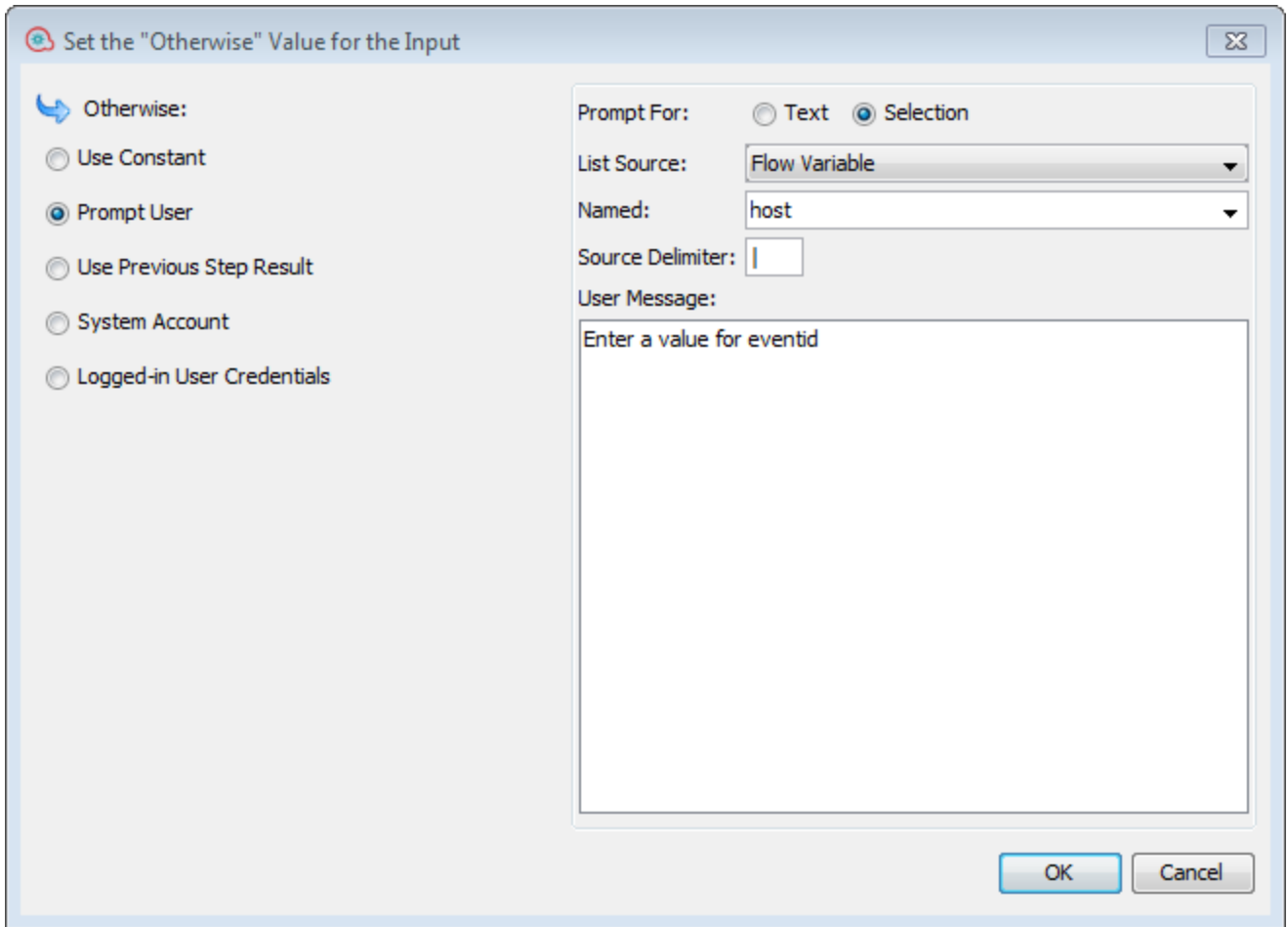
<b>Type</b> 	From the <b>Type</b> list, specify how the input gets its value. The choices are: <ul style="list-style-type: none"> <li>• <b>Single Value</b> </li> <li>• <b>List of Values</b>  to run an operation against multiple targets</li> </ul>
<b>Assign From</b>	Specifies where the input gets its value from. From the <b>Assign From</b> list, select an input from which to take the value.
<b>Otherwise</b>	Shows what action should occur if the flow variable that you specified in the <b>Assign From</b> column does not exist or has no value stored in it.  When you click on an input name, the right pane is automatically updated with its details.  Click  to open the <b>Otherwise</b> dialog box or fill in the fields in the right pane.
<b>Assign To</b>	Select the flow variable that you want to assign the input's value to.

**Note:** To edit the **Assign From** and **Assign To** directly, double-click in the field and type in the new value.

### Input Inspector > Inputs tab > Otherwise pane/Otherwise dialog box

The **Otherwise** pane appears on the right side of the Input Inspector. In this pane, you define what action should occur if the flow variable that you specified in the Assign From column does not exist or has no value stored in it. Some of these fields also appear in the Otherwise dialog box.

#### Otherwise Pane



### Otherwise Dialog Box

**Note:** The **System Account** and **Logged-in user credentials** fields do not appear if you define the input as **List of Values** in the **Type** field.

GUI item	Description
<b>Use Constant</b>	Enter the constant value that will be used for the input. For example, an IP address that is always used.

<p><b>Prompt User</b></p>	<p>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.</p> <ul style="list-style-type: none"> <li>• <b>Prompt For: Text</b> - Prompt the user to enter information. Type the prompt that the user will see in the <b>User Message:</b> box. For example, Enter a value for eventid.</li> <li>• <b>Prompt For: Selection</b> Prompt the user to select a value. Select the type of list the user will select from:           <p><b>Selection List:</b> Select the selection list from the <b>Named</b> dropdown list. All the selection lists are shown.</p> <p><b>Domain Term:</b> Select the domain term from the <b>Named</b> dropdown list. All the domain terms are shown.</p> <p><b>Flow Variable:</b> Select the name of a flow variable . Flow variables are defined in the results for flows. Type a delimiter to the domain terms in the <b>Source Delimiter</b> field.</p> </li> </ul> <p><b>Note:</b> It is not recommended to use the <b>Prompt User</b> option for a step in the middle of a flow.</p>
<p><b>Use Previous Step Result</b></p>	<p>Select a previous step result to be used in the case that this input has no value.</p> <p>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.</p>
<p><b>System Account</b></p>	<p>Enter the system account name and credential type.</p>
<p><b>Logged-in user credentials</b></p>	<p>Enter the user name or password for the logged-in user.</p>

<b>List Source</b>	<p>Select one of the following from the dropdown menu:</p> <ul style="list-style-type: none"><li>• <b>Selection List</b> - select from a set of predefined lists.</li></ul> <p>From the <b>Named</b> list, select the list you want to present to the user.</p> <div style="background-color: #f0f0f0; padding: 5px;"><p><b>Tip:</b> You can add to the set of predefined lists by creating a list. For information on creating a list, see <i>Creating selection lists for user prompts</i>.</p></div> <ul style="list-style-type: none"><li>• <b>Domain Term</b> - 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.</li></ul> <p>From the <b>Named</b> list, select the domain term list you want to present to the user.</p> <ul style="list-style-type: none"><li>• <b>Flow Variable</b> - create a list that is populated by the contents of a flow variable.</li></ul> <p>From the <b>Named</b> list, type or select the flow variable that contains the list. You can reference a flow variable from a different flow. For example, to reference a flow under the folder structure <b>folderA\folderB\flow1</b>, use the string <b>`\${folderA/folder/flow1}</b>.</p> <p>In the <b>Source Delimiter</b> box, type the character that separates the elements in the list.</p>
<b>User Message</b>	Type a prompt message to let the user know what kind of data is needed.

## 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.

- "Specify the input source as a single constant value "
- "Specify the input source as a single user-entered text "
- "Specify the input source as a single user selection "
- "Specify the input source as a constant list of values"


- "Specify the input source as list of input values obtained from user-typed text "
- "Specify the input source as a single user selection "
- "Specifying the Input Source "
- "Specify the input source as the system account details "
- "Specify the input source as the user login name or password"

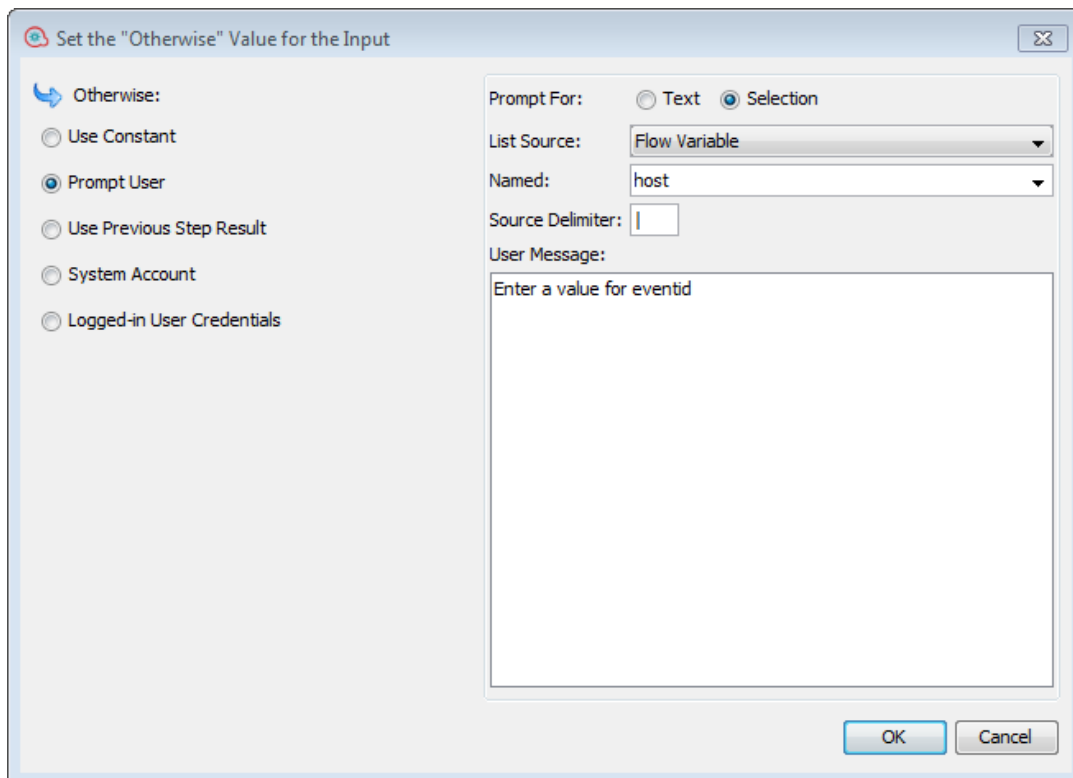
The tasks in this section are optional sub-tasks within the main task of setting up an input. See ["Creating Input" on page 211](#).

## ***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 Otherwise button  to open the Otherwise dialog box for that row or fill in the values directly in the right pane:



Set the "Otherwise" Value for the Input

Otherwise:

- Use Constant
- Prompt User
- Use Previous Step Result
- System Account
- Logged-in User Credentials

Prompt For:  Text  Selection

List Source: Flow Variable

Named: host

Source Delimiter: |

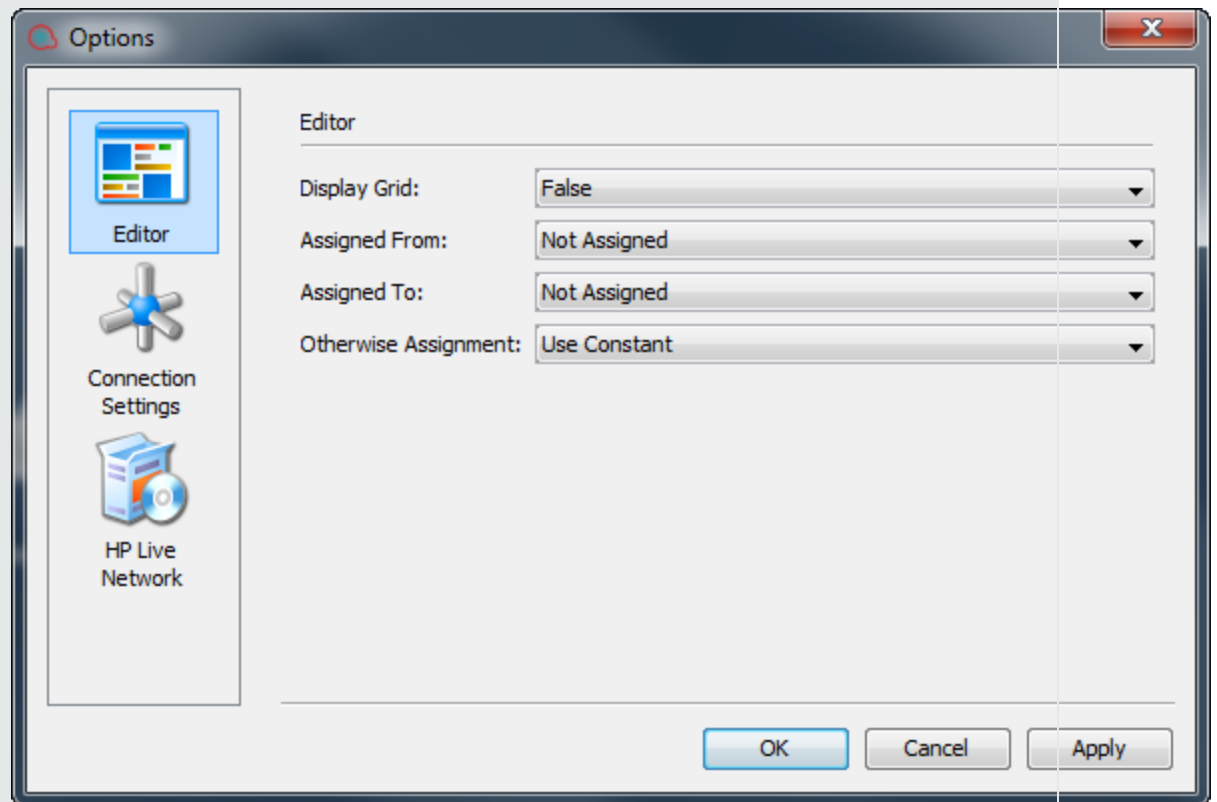
User Message:

Enter a value for eventid

OK Cancel

3. By default, *<not assigned>* appears in the **Assign from** field.
  - To assign that the input value from the value of a variable of the same name, select *<default>* - *<name>* from the **Assign from** dropdown menu.
  - To assign the input value from the value of a different input, select the input name from the **Assign from** dropdown menu.
  - You can also type in a value directly in this field.
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. The default value is **Use Constant** when installing Studio 10.50, and **Prompt User** when upgrading from a previous version of Studio.

**Note:** You can change these defaults in the Editor window under **Configuration > Options:**



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. To




reference a variable that is stored in a folder use the following format:  
`${foldername/variablename}`

### 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.


1. Create an input and select **Single Value** from the **Type** dropdown menu.
2. Click the **Otherwise** button  for that row to open the **Otherwise** dialog box and select **Prompt User** or select **Prompt User** in the right pane. The default value is **Use Constant** when installing Studio 10.50, and **Prompt User** when upgrading from a previous version of Studio.

**Note:** You can change this default by setting the `use.constant.default.value.input` parameter in the `Studio.properties` file.

3. In the **Prompt For:** area, select **Text**.
4. 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.

1. Create an input and select **Single Value** from the **Type** dropdown menu.
2. Click the **Otherwise** button  for that row to open the **Otherwise** dialog box and select **Prompt User** or select **Prompt User** in the right pane. The default value is **Use Constant** when installing Studio 10.50, and **Prompt User** when upgrading from a previous version of Studio.

**Note:** You can change this default by setting the `use.constant.default.value.input` parameter in the `Studio.properties` file.

3. In the **Prompt For:** area, select **Selection**.
4. From the **List Source** dropdown menu, 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. You can reference a flow variable from a different flow. For example, to reference a flow under the folder structure **folderA\folderB\flow1**, use the string **`\${folderA/folder/flow1}**.

In the **Source Delimiter** box, type the character that separates the elements in the list.

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 constant list of values

This type of 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 select **List of Values** from the **Type** dropdown menu.
2. Click the **Otherwise** button  for that row to open the **Otherwise** dialog box and select **Use Constant** or select **Use Constant** in the right pane. The default value is **Use Constant** when installing Studio 10.50, and **Prompt User** when upgrading from a previous version of Studio.

**Note:** You can change this default by setting the **use.constant.default.value.input** parameter in the Studio.properties file.

3. In the **Input Delimiter** box, type the character that separates the elements in the list of values.
4. 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:


```
${foldername/flowvariablename1}<delimiter>${foldername/flowvariablename2}
```

**Note:** You can include both typed values and variables in the same list. For example, `${folder1/flowvariableA}|${folder1/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.


1. Create an input and set the type to **List of Values**.
2. Click the **Otherwise** button  for that row to open the **Otherwise** dialog box and select **Use Constant** or select **Use Constant** in the right pane. The default value is **Use Constant** when installing Studio 10.50, and **Prompt User** when upgrading from a previous version of Studio.
3. 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 **Prompt For:** area, 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.

1. Create an input and set the type to **List of Values**.
2. Click the **Otherwise** button

 for that row to open the **Otherwise** dialog box and select **Prompt User** or select **Prompt User** in the right pane. The default value is **Use Constant** when installing Studio 10.50, and **Prompt User** when upgrading from a previous version of Studio.

3. In the **Input Delimiter** box, type the character that separates the elements in the list.
4. In the **Prompt For:** area, 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 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. Click the **Otherwise** button  for that row to open the **Otherwise** dialog box and select **Use Previous Step Result** or select **Use Previous Step Result** directly in the right pane.
3. If the input value comprises more than one value, 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.

### Specify the input source as the system account details

If you define the input type as single value, you can use user login credentials as an input source. This lets you enable a flow to perform tasks that require system account credentials.

Specify the input value source as the one of the credentials defined in the user account under which the flow is started (user name or password).


1. Click the **Otherwise** button  for that row to open the **Otherwise** dialog box and select **System Account** or select **System Account** directly in the right pane.
2. From the **Account Named** dropdown menu, 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 169](#).
3. In the Credential Type field, select **Username** or **Password**.

### Specify the input source as the user login name or password

You can also use credentials as an input source. This lets you enable a flow to perform tasks that require system account credentials.

Specify the input value source as the one of the credentials defined in the user account under which the flow is started (user name or password).


**Note:** This input method is for Studio 9.x compatibility only. If you want to define a new input for Studio 10.x, use the **System Account** option described in ["Specify the input source as the system account details" above](#).

1. Click the **Otherwise** button  for that row to open the **Otherwise** dialog box.
2. Select **Logged-in User Credentials** or select **Logged-in User Credentials** directly in the right pane.
3. In the Credential Type dropdown, select **User Name** or **Password**.

### Specify the input source as the system account details

If you define the input type as single value, you can use user login credentials as an input source. This lets you enable a flow to perform tasks that require system account credentials.








Specify the input value source as the one of the credentials defined in the user account under which the flow is started (user name or password).

1. Click the **Otherwise** button  for that row to open the **Otherwise** dialog box and select **Use Previous Step Result** or select **System Account** directly in the right pane.
2. From the **Account Named** dropdown menu, 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 169](#).
3. In the Credential Type field, select **Username** or **Password**.

## Reference Material

### Input Inspector > Toolbar












GUI item	Description
<b>Add input</b> 	Adds a new input row. Type the name of the new input and click OK. The new input's attributes are assigned with default values.
<b>Delete row</b> 	Deletes the selected input row.
<b>Move up</b> 	Moves the selected input row higher in the list.
<b>Move down</b> 	Moves the selected input row lower in the list.
<b>Cut</b>  (or Ctrl + X on the keyboard)	Removes the selected input row from its current location. Use the <b>Paste</b> option to place the row in a new location.  <b>Note:</b> If you try to remove a required input, Studio automatically marks it as optional and it appears in italics. You can restore the input by marking it as <b>Required</b> again.
<b>Copy</b>  (or Ctrl + C on the keyboard)	Copies the selected input row. Use the <b>Paste</b> option to place the copy in a new location.
<b>Paste</b>  (or Ctrl + V on the keyboard)	Pastes the copied/cut input row at the current location. If the input name already exists, a new input is added with the same name with an addition of (Copy 1). For example, if you copy and paste an input named host, you will have two inputs: host and host (Copy 1).


**Note:** You can also cut, copy and paste multiple inputs between steps, flows or operations. Select additional inputs by holding down the **Ctrl** key and clicking on the row.

## Input 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.

Input	Sensitive ...	Required	Type	Assign From	Otherwise	Assign To
xml	<input type="checkbox"/>	<input checked="" type="checkbox"/>	 xml	Prompt user from the text	 xml	xml
xpathQuery	<input type="checkbox"/>	<input checked="" type="checkbox"/>	 xpathQuery	Prompt user from the text	 xpathQuery	xpathQuery
delimiter	<input type="checkbox"/>	<input type="checkbox"/>	 delimiter	Prompt user from the text	 delimiter	delimiter

GUI item	Description
<b>Input</b>	Displays the name of the input.
<b>Sensitive Data</b>	<p>Makes this input hidden in the Central user interface and in the Studio internal and remote debugger. If this check box is selected, the input value will be encrypted in Central and Studio, will be displayed as asterisks and will not be persisted</p> <p>This flag is transient, i.e., assignment from a sensitive variable will also make the assigned variable sensitive. This transitivity is only seen during runtime execution, and is not reflected in the Studio user interface.</p> <p>When used in scriptlets, sensitive data is retrieved as encrypted.</p> <p>Sensitivity is also retained when using expressions for assignment. For example, <code>\${input1}</code>.</p>
<b>Required</b>	Makes this input mandatory.
<b>Type</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> </ul>

<p><b>Assign From</b></p>	<p>Specifies where the input gets its value from. From the <b>Assign From</b> list, select an input from which to take the value or type in a value directly in this field.</p> <p>By default, <i>&lt;not assigned&gt;</i> appears in this field.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p><b>Note:</b> In some situations, the flow level input will not appear in Central:</p> <ul style="list-style-type: none"> <li>• If you have not assigned a value to <b>Assign From</b> and you have set <b>Otherwise</b> to <b>Use Constant</b></li> <li>• If the value of <b>Assign From</b> is different from the flow input name</li> </ul> </div>
<p><b>Otherwise</b></p>	<p>Shows what action should occur if the flow variable that you specified in the Assign From column does not exist or has no value stored in it.</p> <p>Click  to open the Otherwise dialog box or fill in the fields in the right pane.</p>
<p><b>Assign To</b></p>	<p>Select the flow variable that you want to assign the input's value to or type in a value directly in this field.</p> <p>By default, <i>&lt;not assigned&gt;</i> appears in this field.</p>

### Step Inspector > Inputs tab > Otherwise pane/Otherwise dialog box

The **Otherwise** pane appears on the right side of the Input Inspector. In this pane, you define what action should occur if the flow variable that you specified in the Assign From column does not exist or has no value stored in it. Some of these fields also appear in the Otherwise dialog box. If you make changes in the **Otherwise** pane, the **Otherwise** dialog box is automatically updated accordingly and vice versa.

#### Otherwise Pane



Name:  Input Type:

Assign from Variable:

**Otherwise:**

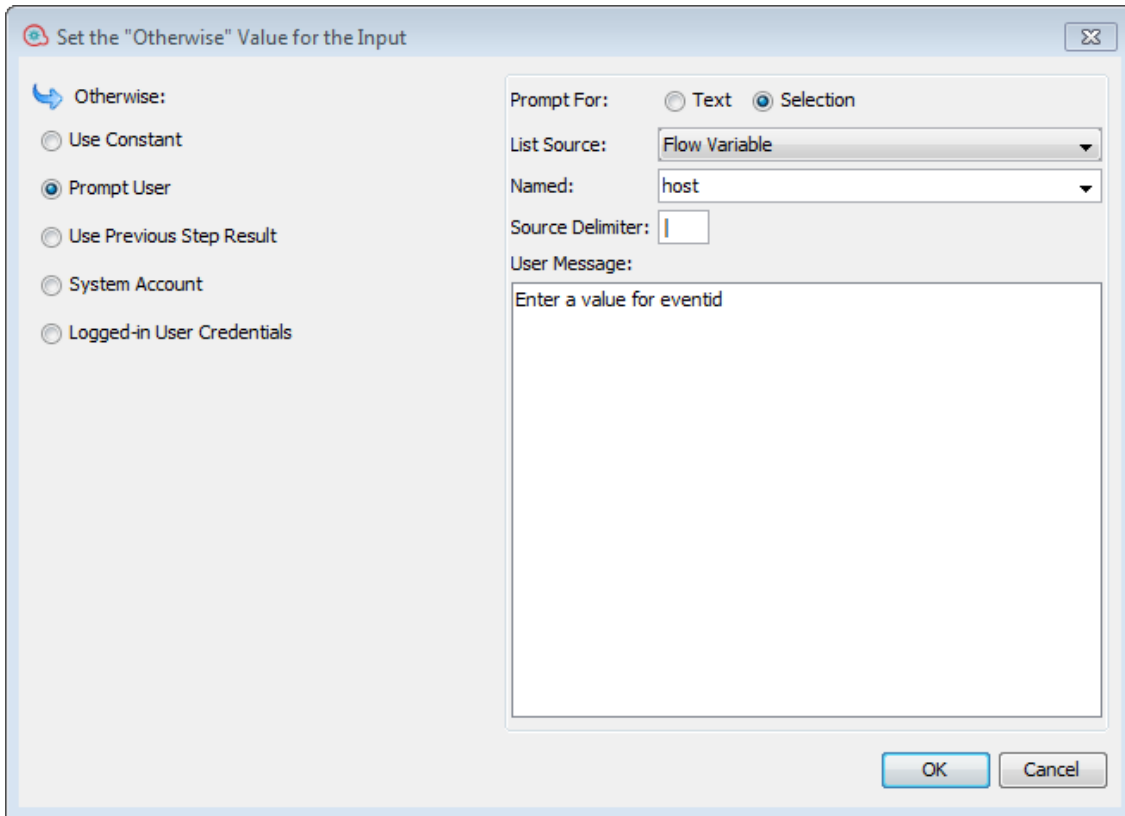
- Use Constant
- Prompt User
- Use Previous Step Result
- System Account
- Logged-in User Credentials




Constant Value:


Assign to Variable:

Sensitive Data  Required Validation Format:

### Otherwise Dialog Box



GUI item	Description
<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> </ul>

<p><b>Assign From column</b></p>	<p>Specifies where the input gets its value from. From the <b>Assign From</b> list, select an input from which to take the value.</p> <p><b>Note:</b> If you have not assigned a value to <b>Assign From</b> and <b>Otherwise</b> is set as <b>Use Constant</b>, the flow level input will not appear in Central.</p> <p>If you have assigned a value to <b>Assign From</b> and <b>Otherwise</b> is set as <b>Use Constant</b>, the flow level input will appear in Central.</p>
<p><b>Otherwise column</b></p>	<p>Shows what action should occur if the flow variable that you specified in the <b>Assign From</b> column does not exist or has no value stored in it.</p> <p>Click  to open the <b>Otherwise</b> dialog box or fill in the fields in the right pane.</p>
<p><b>Assign To column</b></p>	<p>Select the flow variable that you want to assign the input's value to.</p>
<p><b>Name</b></p>	<p>Displays the name of the step (read only).</p>
<p><b>Input Type</b></p>	<p>Displays the input type. May be modified here.</p>
<p><b>Assign from Variable</b></p>	<p>Enter or select the name of the flow variable that will be the source of the input.</p>
<p><b>Otherwise</b></p>	<p>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.</p>
<p><b>'Otherwise: &lt;action&gt;' Configuration</b></p>	<p>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.</p>
<p><b>Assign to Variable</b></p>	<p>Select the flow variable that you want to assign the input's value to.</p>
<p><b>Input Delimiter</b></p>	<p>Enter the delimiter to be used to separate values in</p>
<p><b>Required</b></p>	<p>Makes this input mandatory.</p>
<p><b>Validation Format</b></p>	<p>Validates the input's value with a system evaluator. Select a system evaluator from the dropdown list.</p> <p>For example, if you want to prompt the user to provide an IP address, use the IP validation format. Studio will then check that the input provided is in valid IP address format.</p>

<b>Record Under</b>	Makes the value available for diagnostics or auditing. This field is a legacy field from OO 9.x.  You can select the item under which you want to record the input in the database from the <b>Domain Terms</b> list.
---------------------	---

## 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.

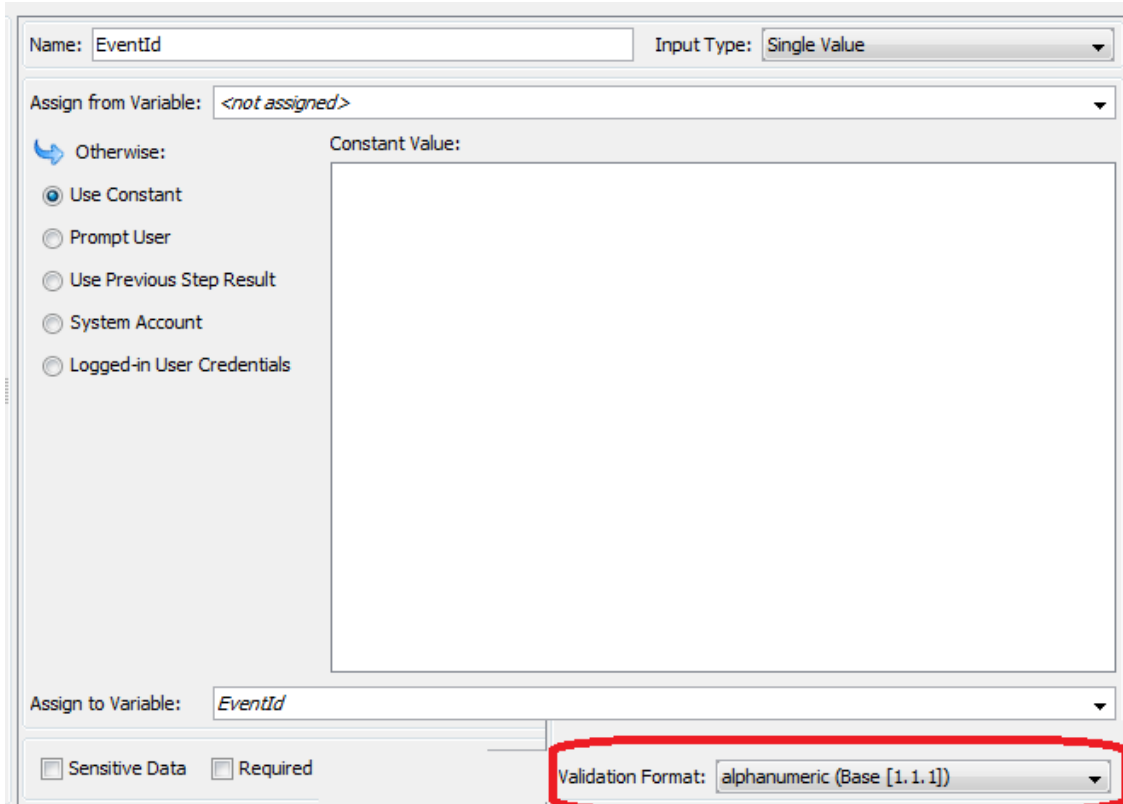
Evaluators use the following:

- 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 313](#)
- Scriptlets – for more information, see ["Using Scriptlets in a Flow" on page 308](#)

## 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.



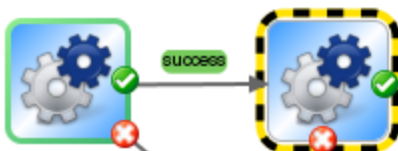
The screenshot shows the Input Editor interface for an input named 'EventId'. The 'Input Type' is set to 'Single Value'. The 'Assign from Variable' dropdown is set to '<not assigned>'. Under the 'Otherwise:' section, 'Use Constant' is selected. The 'Constant Value' field is empty. The 'Assign to Variable' dropdown is set to 'EventId'. At the bottom, there are checkboxes for 'Sensitive Data' and 'Required'. The 'Validation Format' dropdown menu is highlighted with a red rectangle and is set to 'alphanumeric (Base [1. 1. 1])'.

**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 211](#).

## 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 242](#).

**Note:** In HP OO 10.10, transition descriptions are now limited to 1000 bytes.

## 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.

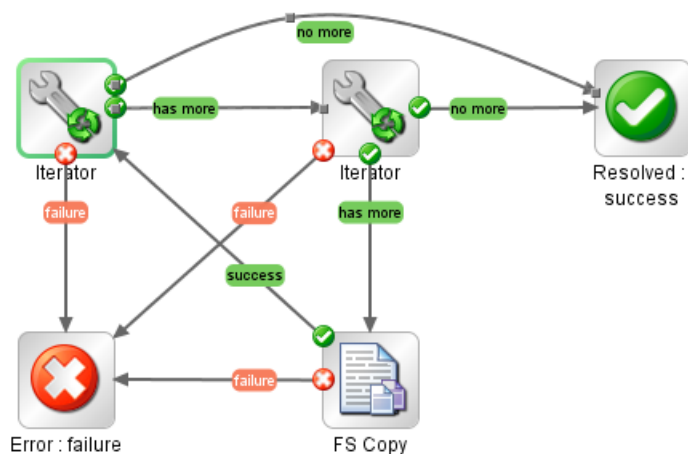
### Count the completion of the transition in the flow's ROI value

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.

Inspector

Transition Name: success

Gated Transition

Check user's groups before proceeding

Required Group: AUDITOR

Hand-off flow run after this transition

Transition ROI Value: 0.0

Description

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 288](#).

## 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.

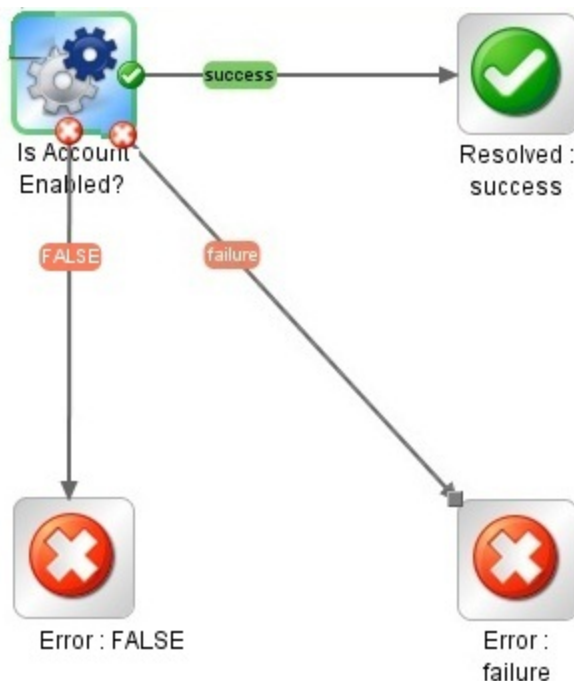
For example, you can create a rule that will only give a **Success** response if the results include a value that is greater than 1.

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

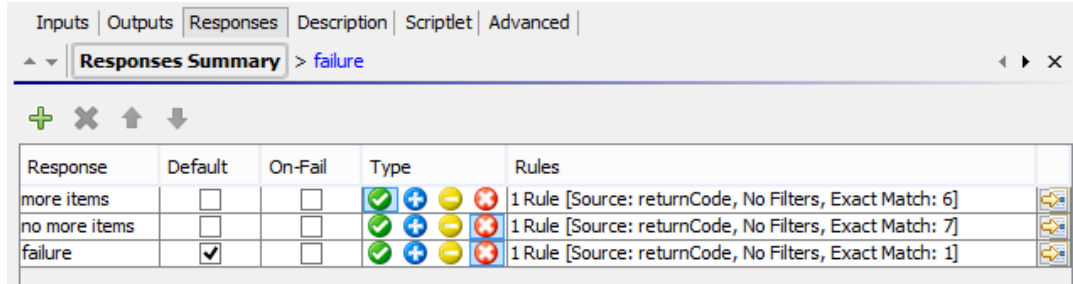
- Avoid confusing a failed operation with a negative result. For example, if an operation asks a question for which the answer may be TRUE or FALSE, a FALSE answer is not the same as a failure. In such a case, you need two **Error** return results: one for a FALSE result and one for a failure of the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



## What do you want to do?

### Add a response to an operation

1. Click the **Responses** tab of the operation.



2. Select the row that you want the new response to follow. For example, if you select the first row, the new response will appear on the second row.
3. Click **Add Response** and then type a name for the new response.
4. 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.
5. 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.
6. 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.

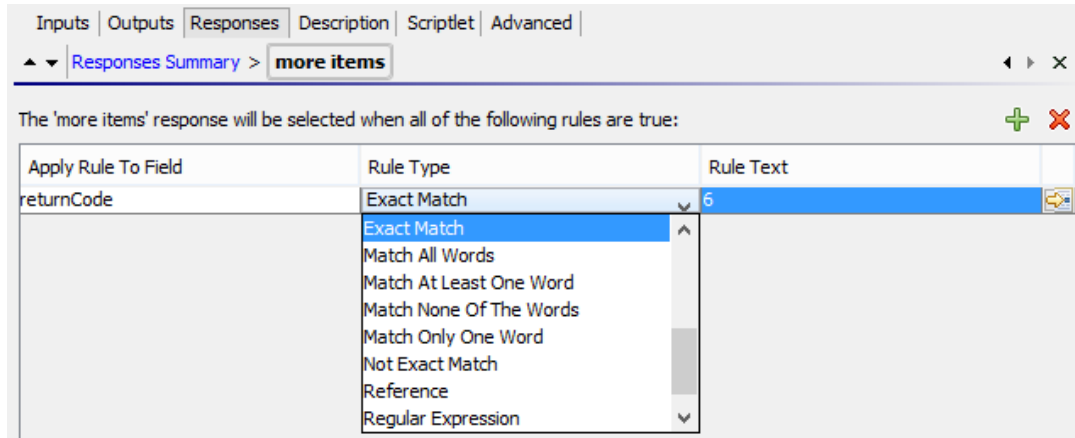
7. To create a rule for the response, at the right end of the response's row, click the right-pointing arrow . For more information, see *Create a rule for the response*, below.

### 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.



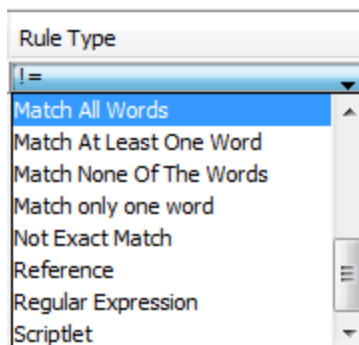
3. In the **Apply Rule to Field** column, select the results field whose value you want to test a rule against.

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 324](#).

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.

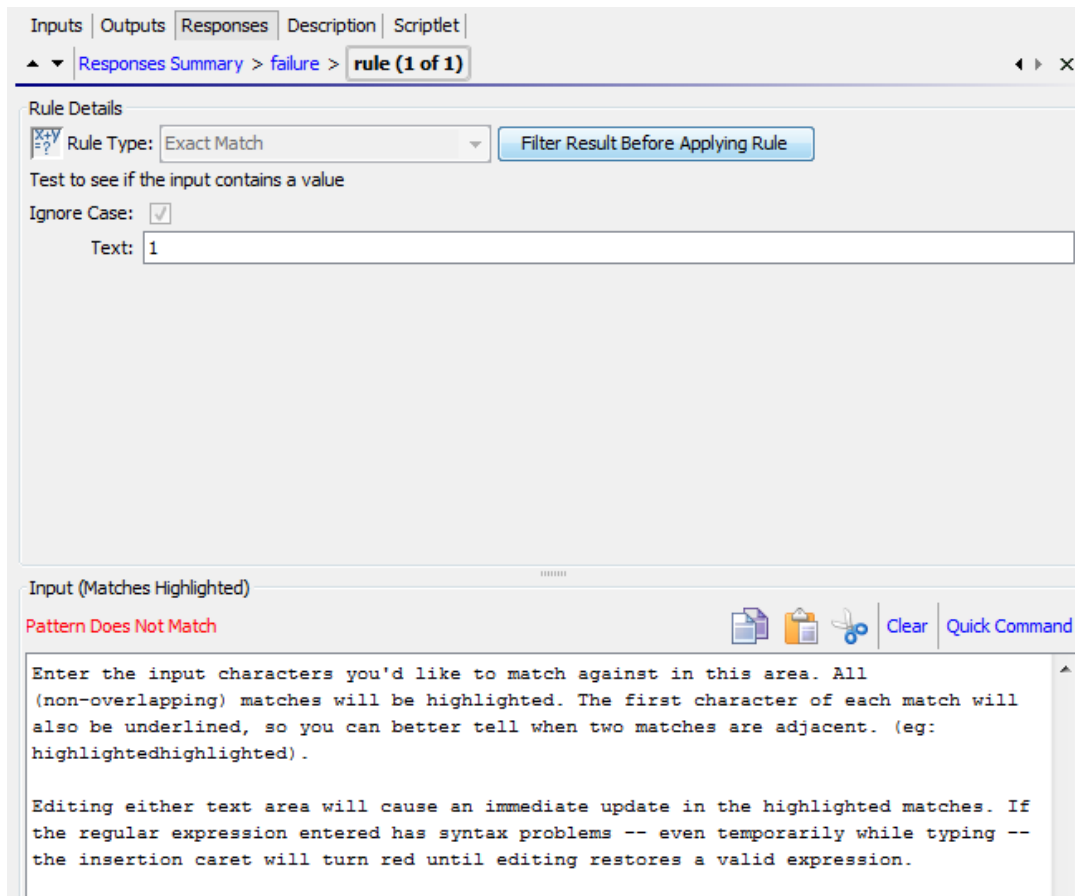
5. In the **Rule Text** column, type the text to use in the test.

## 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

Test to see if the input contains a value

Ignore Case:

Text:

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 308](#).

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 313](#).

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 265](#).

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 313](#).

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 288.

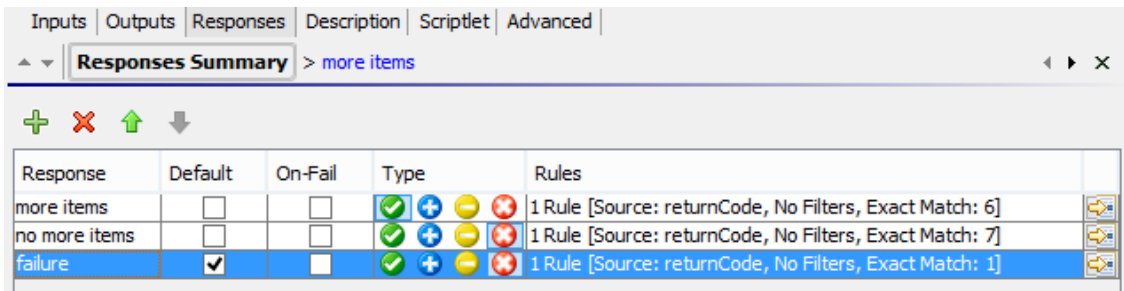
### Delete a response from an operation or flow


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**.

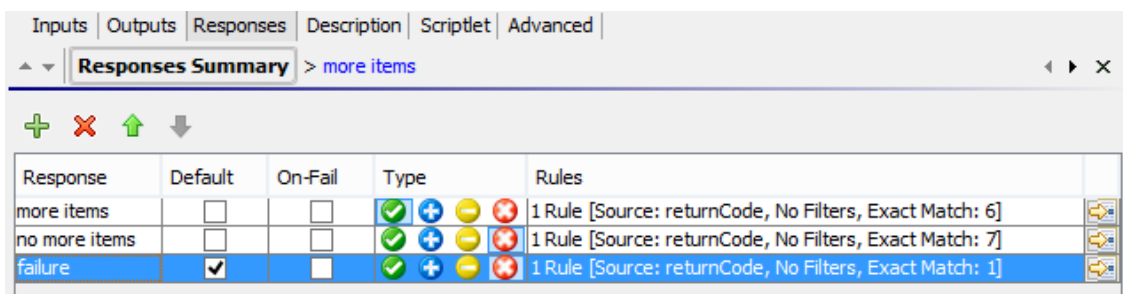








GUI item	Description
<b>Add Response</b>	Adds a new response row.
<b>Remove Response</b>	Removes the selected response row.
	Click to move the selected response up or down in the list.

### Operation Properties sheet > Responses tab

The **Responses** tab in an operation's **Properties** sheet is where you specify the possible responses for an operation.



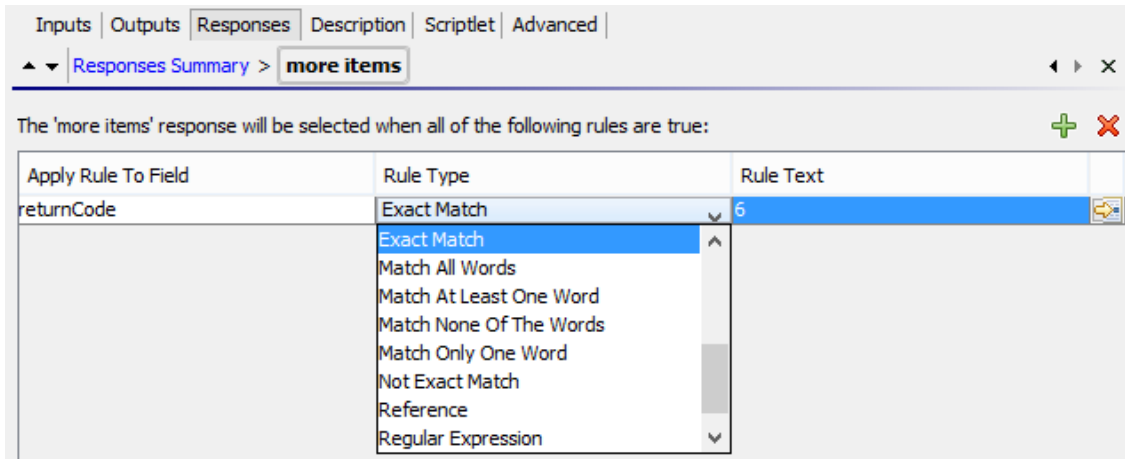




GUI item	Description
<b>Add Response</b>	Adds a new response row.
<b>Remove Response</b>	Removes the selected response row.
	Click to move the selected response up or down in the list.
<b>Default</b>	Select to identify a response as the default response. The default response is the one chosen if none 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.

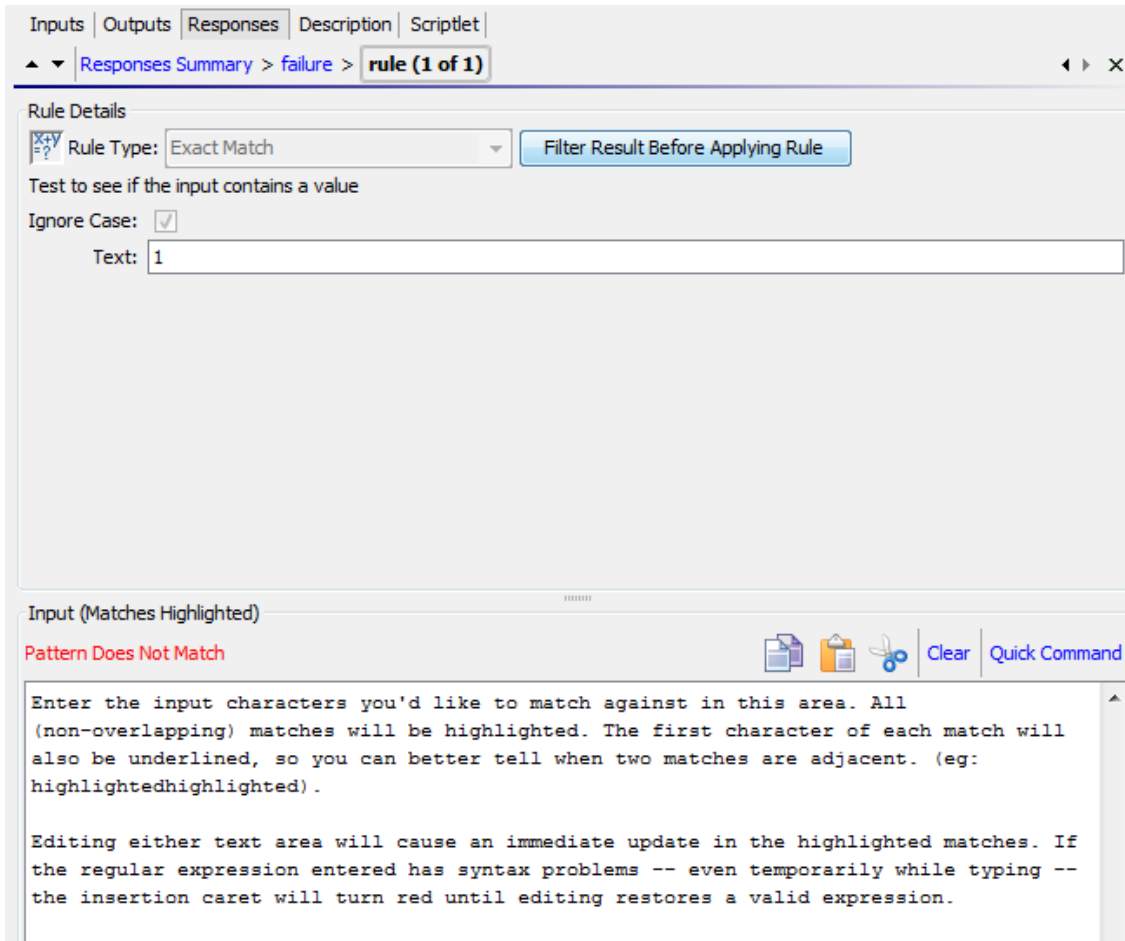


GUI item	Description
<b>Add Response</b>	Adds a new response row.
<b>Remove Response</b>	Removes the selected response row.
	Click to move the selected response up or down in the list.
<b>Apply Rule to Field</b>	Select the results field whose value you want to test a rule against. The result fields you 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 you limit a response so that it only occurs when a particular condition of the result is true.



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" below](#).

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 256](#).

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 265](#).

## 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.

If an operation contains sensitive data that you would like to hide from the end-user, you can define this at the step level. See ["Setting Step Results" on page 256](#).

### *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**. You can see the primary output in the Outputs Summary above the list of outputs in the Outputs tab.
  - A **secondary output** in an operation is another output, in addition to the primary output.

**Tip:** You can narrow an output to a more focused selection by creating one or more filters for the output. See "[Filtering Output and Results](#)" on page 265.

## ***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.

Note that this can only be done at step level.

1. Right-click the operation in the **Project** pane and select **Properties**.


**Note:** To open multiple operations, select them using the SHIFT or CONTROL keys, right-click, and select **Open**.

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 the 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 265](#).

### 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.

## Delete an output from an operation

1. Right-click the operation in the **Project** pane and select **Properties**.
2. Select the **Outputs** tab.
3. Select the output you want to delete and click  ..

## Cut, copy and paste an output

1. Open the **Properties** sheet for the operation and select the **Outputs** tab.
2. Select an output to cut/copy.
3. From the **Outputs** toolbar, click  or  .
4. Move to the row above which you want to paste the result.
5. Click  .
6. Save the operation.

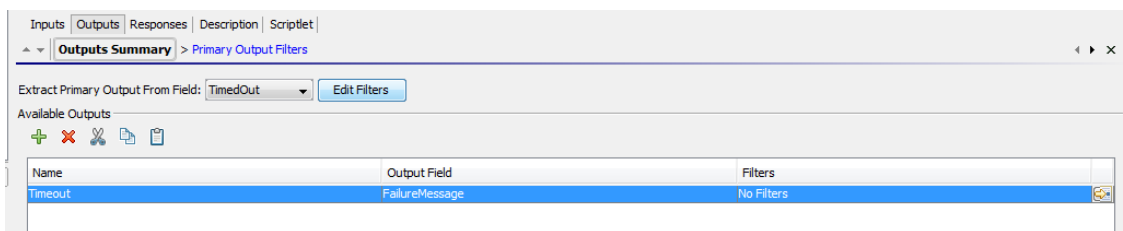
### Note:


- You can copy outputs to other operations.
- You can paste outputs with same name. Studio automatically names the new output **<output> (Copy 1)**.

## 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

### Properties sheet > Outputs Toolbar



GUI item	Description
<b>Add output</b> 	Adds a new output row. Type the name of the new output and click OK. The new output's attributes are assigned with default values.
<b>Delete output</b> 	Deletes the selected output row.
<b>Cut</b>  (or Ctrl + X on the keyboard)	Removes the selected output row from its current location. Use the  option to place the row in a new location.
<b>Copy</b>  (or Ctrl + C on the keyboard)	Copies the selected output row. Use the  option to place the copy in a new location.
<b>Paste</b>  (or Ctrl + V on the keyboard)	Pastes the copied/cut output row at the current location. If the output name already exists, Studio automatically names the new output <b>&lt;output&gt; (Copy 1)</b> .

## 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 and you can mark the operation's raw results as sensitive. Results contain output from the operation and additional information that you want to propagate.

You can also mark a result as sensitive data.

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 282](#).

**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 292](#).

**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 252](#).

For security purposes, there may be cases in which you need to hide the results of an operation. For example, an operation may generate a random password, and you want to hide this password from the end user. You can specify this in the Sensitive Data area. You can apply this functionality on an individual step, not only on an entire operation.

**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 265](#).

## ***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 252](#).

### **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 265](#).

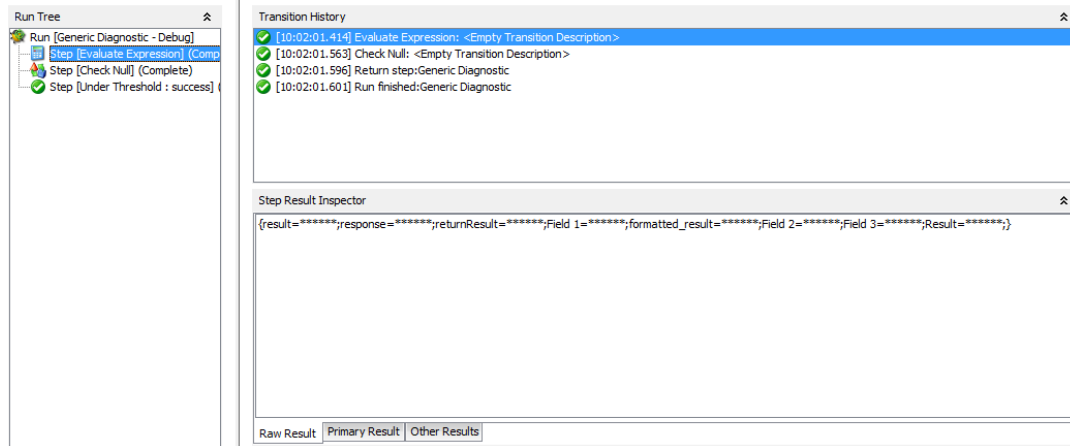
## Hide the operation's raw result

1. In the Sensitive Data area, select the **Hide the operation's results** check box. This will hide the entire raw results of the operation, and make this information displayed as asterisks in the Studio and Central user interface.

After this check box is selected, the operation's primary output will also be hidden and displayed as asterisks in the Studio and Central user interface.

When used in scriptlets, sensitive data is retrieved as encrypted.

For example, if you select the **Hide the operation's results** check box for the step Evaluate Expression, it is displayed in Studio as shown below, and similarly in Central:



## Hide the step result

1. Double-click a step in the authoring pane.
2. Select the **Results** tab.
3. Find the step result that you want to hide from the list of step results.
4. In the Results tab, select the **Sensitive Data** check box.

This makes the result hidden in the Central user interface and in the Studio internal and remote debugger. The result value will be displayed as asterisks and will not be persisted to the OO database.

This flag is transient, i.e., assignment from a sensitive variable will also make the assigned variable sensitive. This transitivity is only seen during runtime execution, and is not reflected in the Studio user interface.

If a result is assigned from a sensitive input or sensitive result, it will also be sensitive. If a result is assigned from an operation raw result that is hidden, you must mark it as sensitive if you want to hide it in the Studio and Central user interface.

When used in scriptlets, sensitive data is retrieved as encrypted.




Sensitivity is also retained when using expressions for assignment. For example, `$(input1)`.

## Hide the step raw result from the end-user

1. Double-click a step in the authoring pane.
2. Select the **Results** tab.
3. In the Sensitive Data area, select **Hide Operation Raw Results**.
4. Save the step.

**Note:** If the step contains a sub-flow, there is no **Hide Operation Raw Results** section.


## Cut, copy and paste a result

1. Double-click a step in the authoring pane.
2. Select the **Results** tab and select a result to cut/copy.
3. From the **Results** toolbar, click  or .
4. Move to the row above which you want to paste the result.
5. Click .
6. Save the step.

### **Note:**

- You can copy step results to another step results pane only. Flow outputs can be copied to other flows or operations.
- You can paste step results with same name. The new result will be identical to the existing result.

## Delete a result from a step

1. Double-click a step in the authoring pane.
2. Select the **Results** tab and from the Results toolbar, click .
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.

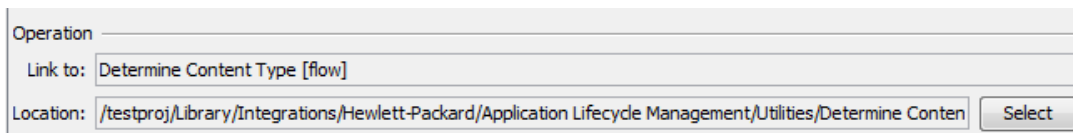
## Change the operation linked to a step

You can select a different operation to be linked to a step. After selecting the new operation, you can determine whether to overwrite existing inputs with the inputs of the new operation or to merge

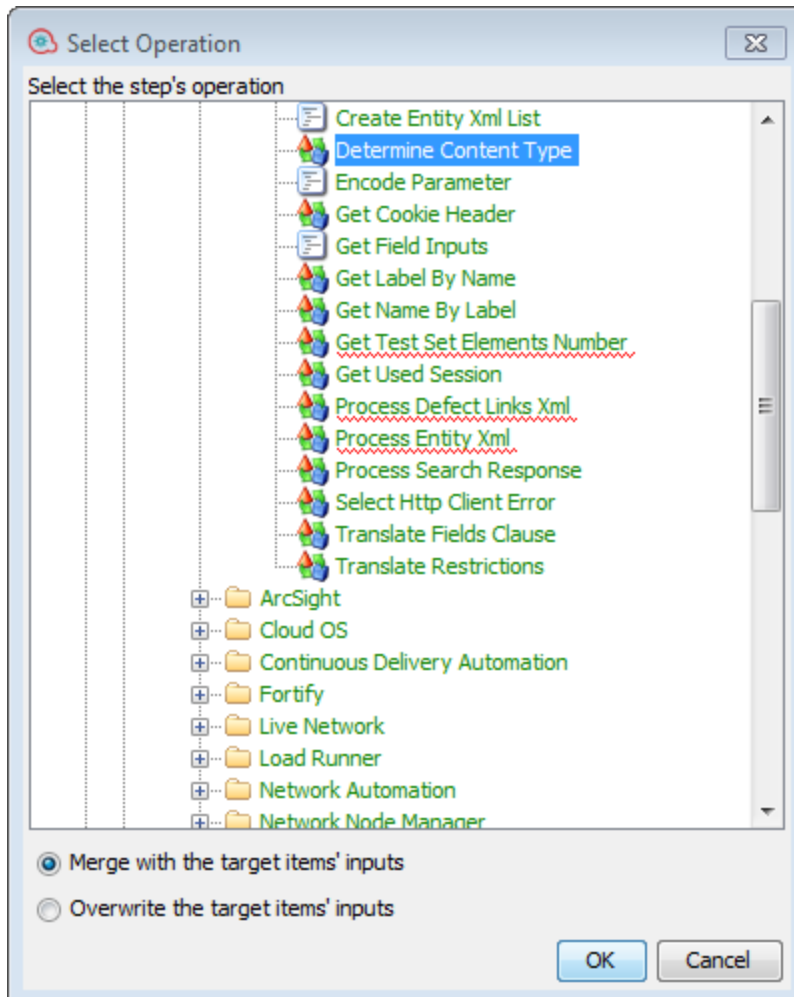
them with the existing inputs.

1. Double-click a step in the authoring pane.
2. In the Step Inspector, select the **Advanced** tab.

You can see the location and name of the current operation. For example:



3. Click the **Select** button. The Select Operation window opens:



4. Browse to the new operation.
5. To add the new operation's inputs to the current inputs, select **Merge with the target items'**

inputs.

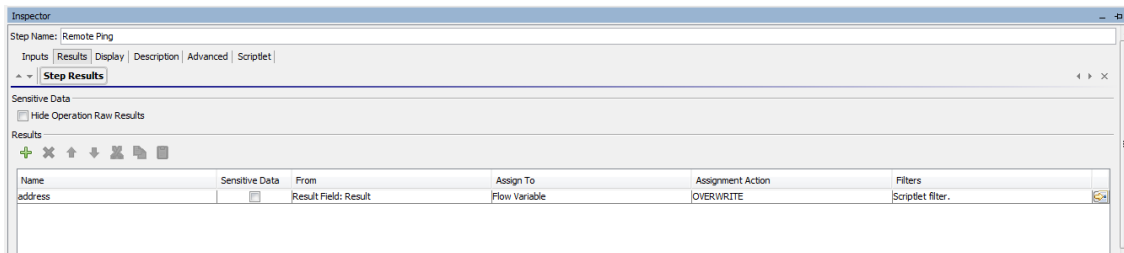
**Note:** If an input name already exists, the original input remains and is not overwritten.

6. To delete the current inputs and add only the new inputs, select **Overwrite the target items inputs**.
7. Click OK.
8. 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.



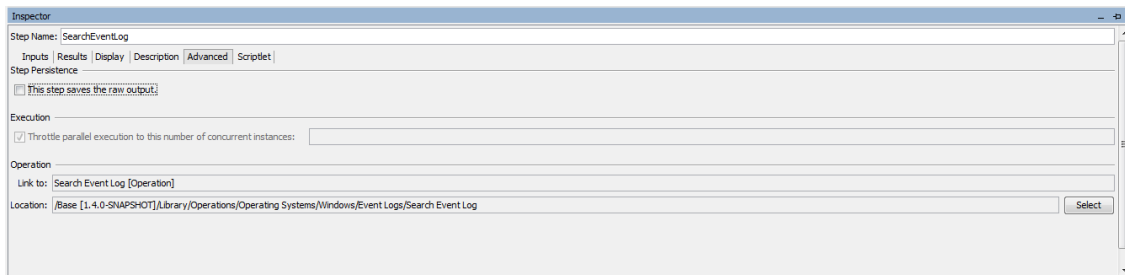
GUI item	Description
<b>Name</b>	Enter a name for the result. This name will be used as the name of the flow variable or flow output field.
<b>Sensitive data</b>	Makes this result hidden in the Central user interface and in the Studio internal and remote debugger. The result will be displayed as asterisks and will not be persisted to the OO database.
<b>Add Result</b>	Adds a new result row.
<b>Remove Result</b>	Removes the selected result row.
<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>

<p><b>Assignment Action</b></p>	<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="border: 1px solid gray; padding: 5px; 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>

**Note:** After you mark a step result as Sensitive Data, it remains sensitive during the whole run (and therefore encrypted). The step result passes this behavior to every input/step result that is assigned from it.

### 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>This step saves the raw output</b>	Select this check box if you need to save all of the step's raw results.
<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.

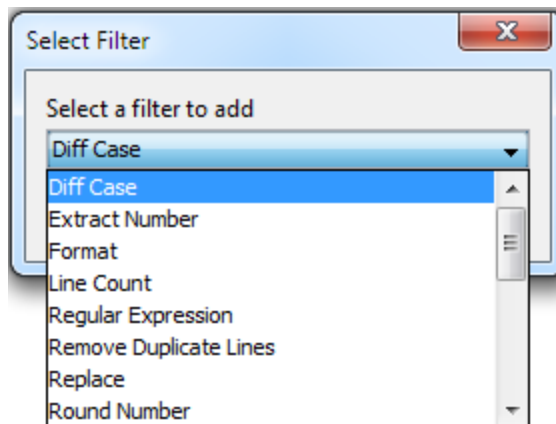
### *What do you want to do?*

#### 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**

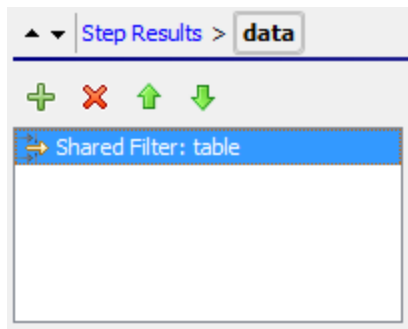
The filters are applied to the data in the **Test Filter Input** box (in top-to-bottom order), and the filtered results appear in the **Test Output** box.

## 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 179](#).

## Filter Options

### Diff Case

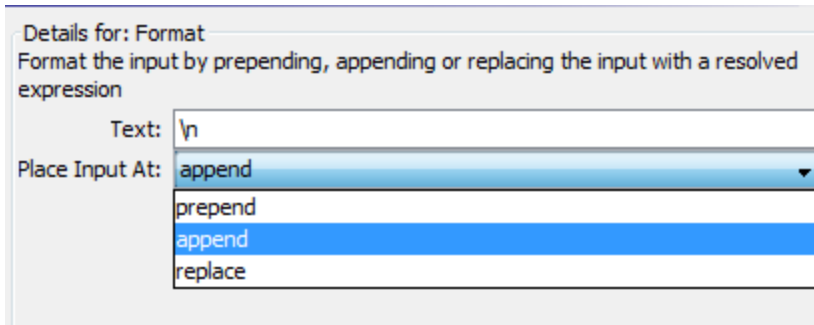
A **Diff Case** filter changes all the characters in the string either to upper case or to lower case. If you leave the **To Upper Case** check box cleared, the filter changes all the characters to lower case.

### 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.



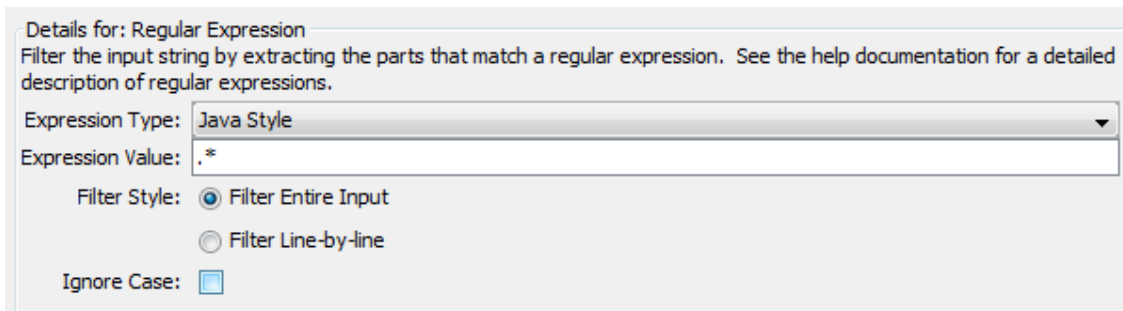
1. In the **Text** box, type the text you want to attach to the result or use to replace the result.
2. From the **Place Input At** list:
  - To attach the text to the front of the existing text, select **prepend**
  - To add the text to the back of the existing text, select **append**
  - To replace 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 313](#).

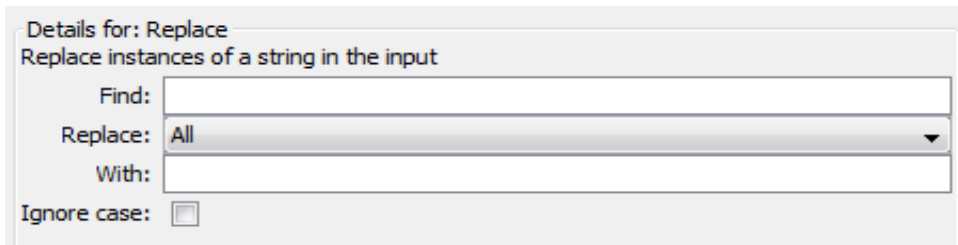
## Remove Duplicate Lines

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

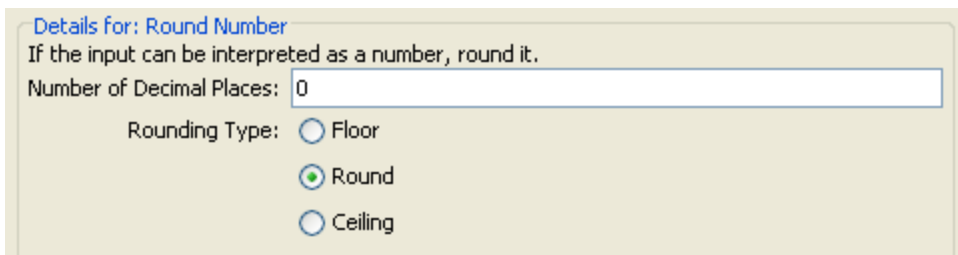
This filter replaces the first or last instance or all instances of one string with another string.



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

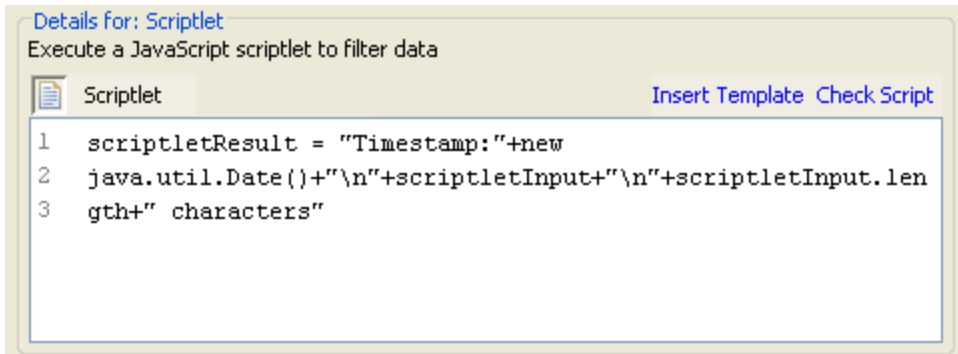
This filter rounds numbers up or down.



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**.

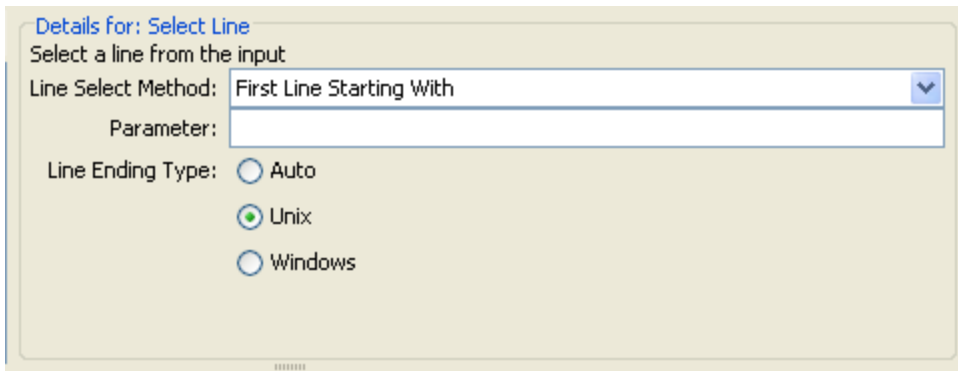
The template that is inserted is specific to the language that you chose and includes the most commonly used commands for accessing flow variables, values of global variables, operation results, and inputs, and for setting and manipulating flow variable values and results.

2. To debug the scriptlet, click **Check Script**.

For more information about scriptlets, see ["Using Scriptlets in a Flow" on page 308](#).

## Select Line

This filter defines a line that you want to extract from the raw results.



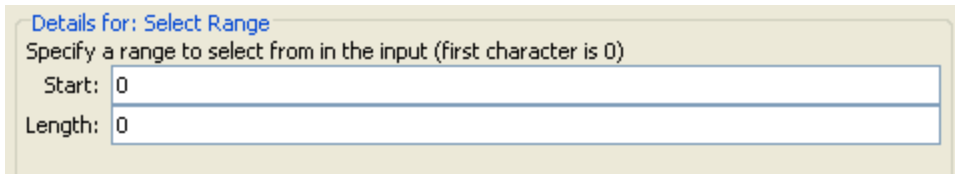
1. From the **Line Select Method** list, select a criterion for the line that you're interested in.
2. In the **Parameter** box, type a string that the line contains.
3. From the **Line Ending Type** group, select one of the following:
  - If the text that you're filtering was generated on a Unix operating 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.



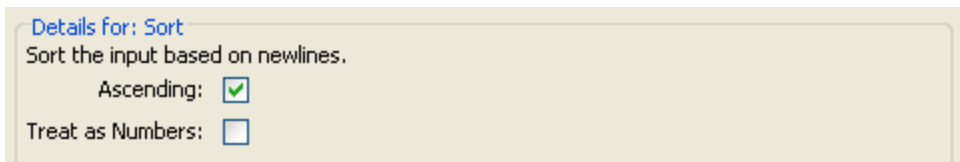
Details for: **Select Range**  
Specify a range to select from in the input (first character is 0)  
Start: 0  
Length: 0

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.

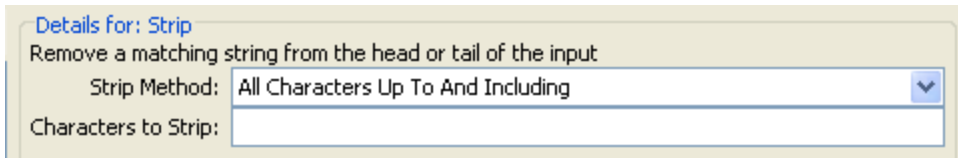
Note that ascending ASCII order is roughly as follows, for English characters:

- 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:

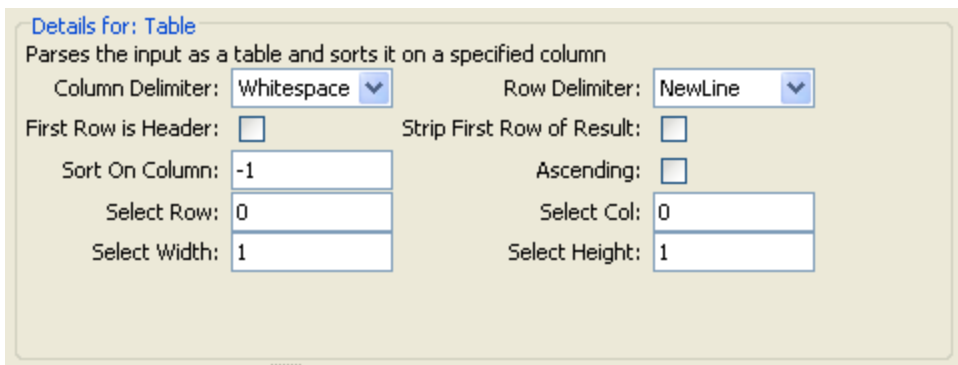
1. From the **Strip Method** list, select how you want the filter to strip the raw results. You can specify the following options for stripping the string that you specify in the **Characters to Strip** text box:
  - **All Characters Up To** the string
  - **All Characters Up To And Including** the string
  - **All Characters After** the string
  - **All Characters After And Including** the string
2. 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:

`jdove@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.

### Details for: XPath Query

Filters an xml document based on an xpath query and returns the results of the query.

XPath Query:

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]
```

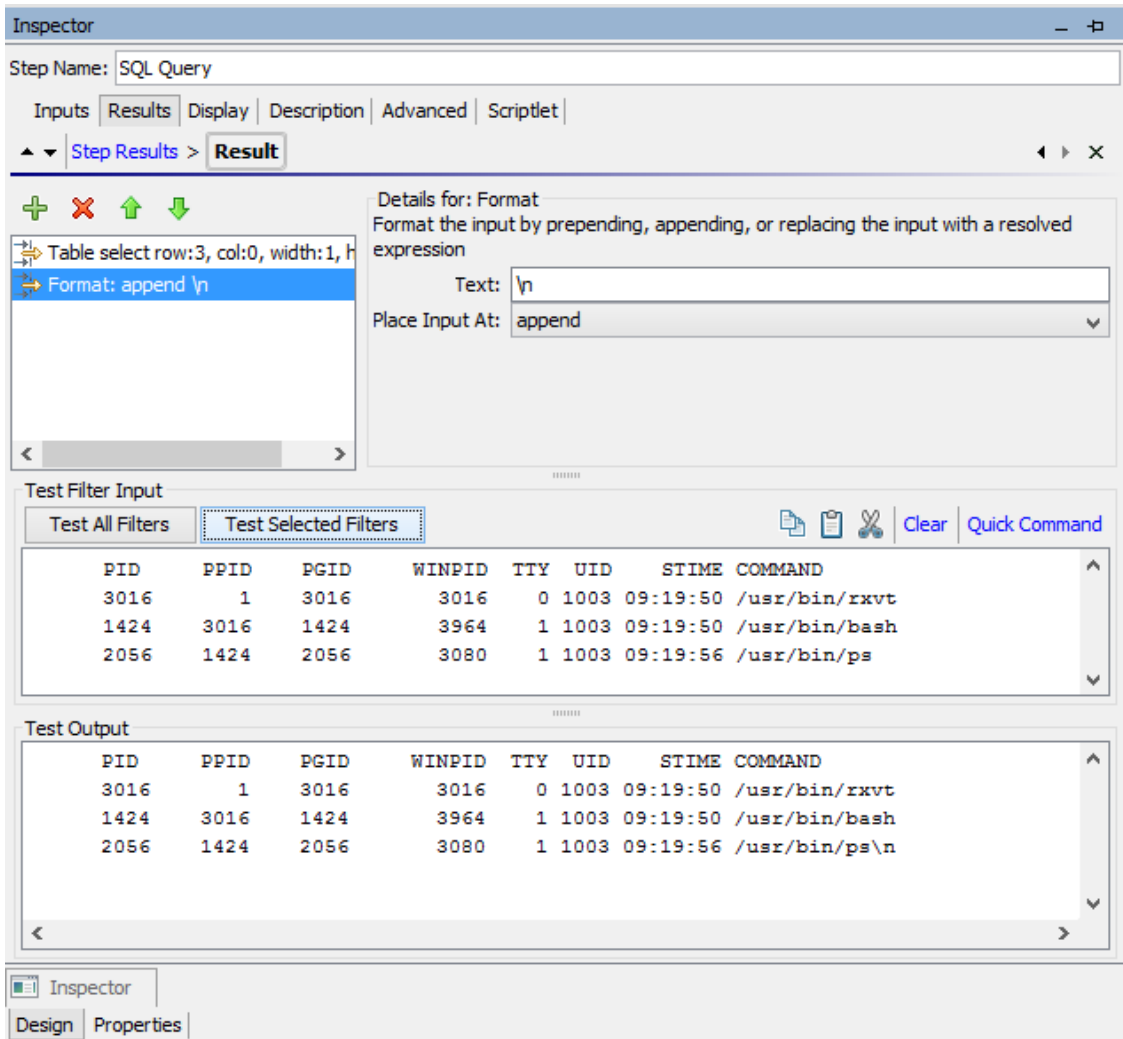
Because `<volume>` is a child of `<company>`, you do not need to specify its relative path.



## 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. From the **Assign to** dropdown, select the variable to which you want the value to be assigned.
4. Save.

For more information about creating an input, see ["Creating Input" on page 211](#).

### 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 252](#).

If necessary, to obtain the precise results you want, create one or more filters for the result. See ["Filtering Output and Results" on page 265](#).

### 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	#	Green Arrow	User Icon
-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
  - 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 322.

### 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.

## 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.

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"	-		✓






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
	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.



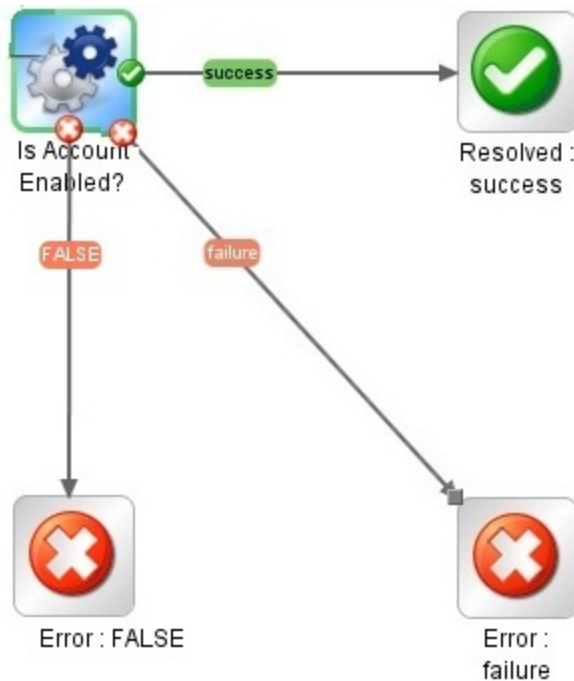
**Note:** A flow that is intended solely to gather data should return **Resolved**  when it is complete, rather than **No Action Taken** .

- **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 189](#).

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

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.

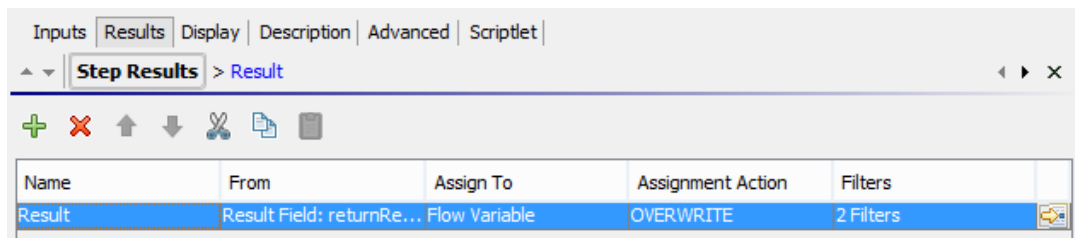
## What do you want to do?

### 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 256](#)).
3. Configure the result so that result data is stored in a **flow output field**. This makes the data available outside of the subflow.



- 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 265](#)).
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.

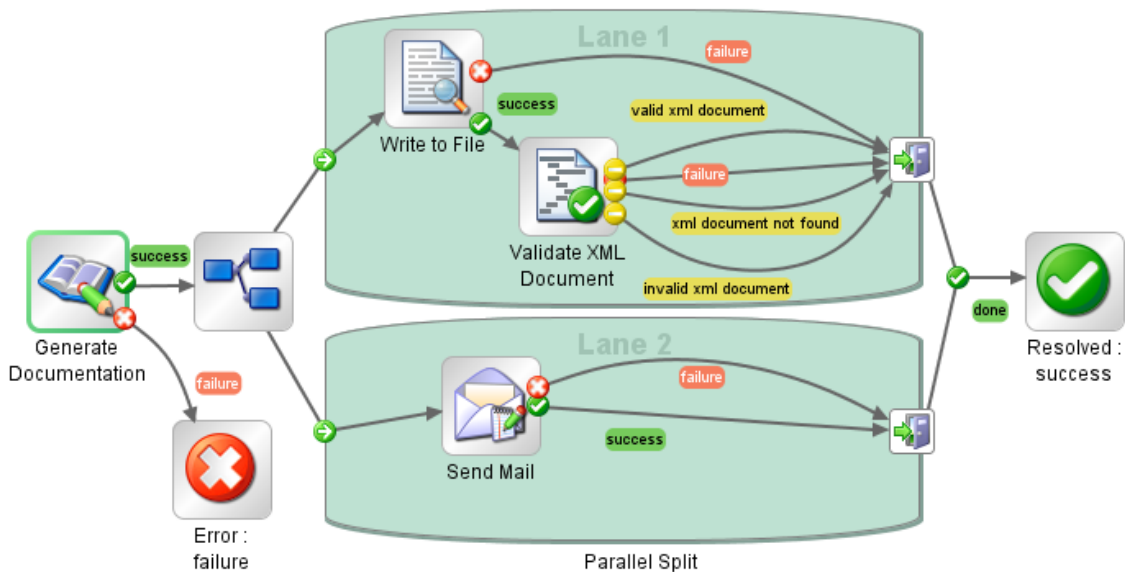
## Creating a Flow with Parallel Split Steps

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.



**Note:** It is not possible for parallel split steps to be non-blocking.

A parallel lane cannot include a response step.

## 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.

**Note:** You cannot add a response step in a parallel lane.

b. Connect the steps within each lane.

c. Connect the last step in the lane to the **Lane-end** icon .

4. Connect the parallel split 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

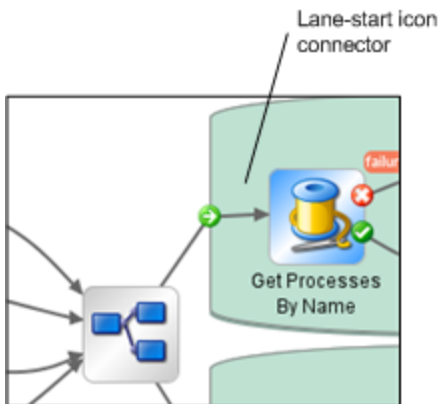
By default, lanes are called **Lane 1**, **Lane 2**, and so on.

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 336](#).

## 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.

**Note:** It is not possible for multi-instance steps to be non-blocking.

A multi-instance lane cannot include a response step.

## ***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.

Name	From	Assign To	Assignment Action	Filters
var1	Result Field: Result	Flow Variable	OVERWRITE	No Filters
var2	Result Field: Result	Flow Variable	PREPEND	No Filters
var3	Result Field: returnRe...	Flow Variable	OVERWRITE	No Filters
var4	Result Field: TimedOut	Flow Variable	OVERWRITE	No Filters
var5	Result Field: returnRe...	Flow Variable	OVERWRITE	No Filters

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 = wv (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 308](#).

## 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

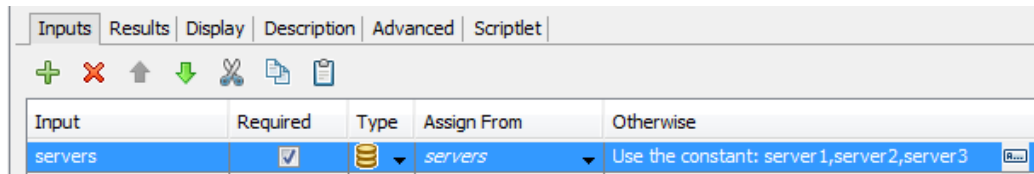
1. On the authoring pane toolbar, click the **Step Palette** button  to display the **Step** palette.
2. From the **Step** palette, drag the **Multi-instance**  icon to the authoring canvas.
3. From the **Projects** pane, drag the flow or operation into the multi-instance lane.


**Note:** You can add multiple flows and operations to the multi-instance lane.

You cannot add a response step in a 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**.

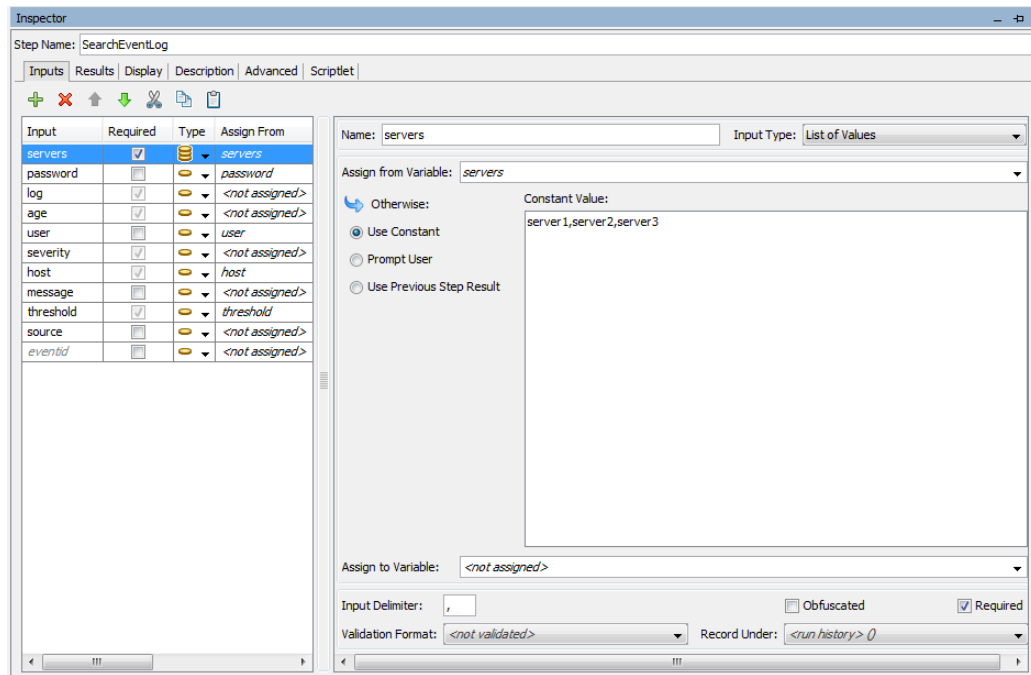


- d. Click the right-pointing arrow  at the end of the row to open the Input Editor for that row.
- e. In the **Input Delimiter** box, type a delimiter (a character(s) that separates the elements in the list).

**Note:**


- o To define a delimiter that contains several characters, use quotation marks. For example, "\$%^".
- o To use a quotation mark as part of the delimiter, type it in twice. For example, to define the delimiter "\$%", type ""\$%.

- 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 222.

5. Connect the different parts of the multi-instance step:

- a. Connect the **Lane-start** icon  to the first step in the multi-instance lane.
- b. If there are multiple steps in the multi-instance step, connect the steps.


- c. Drag all the response lines from the last step in the lane to the **Lane-end** icon .

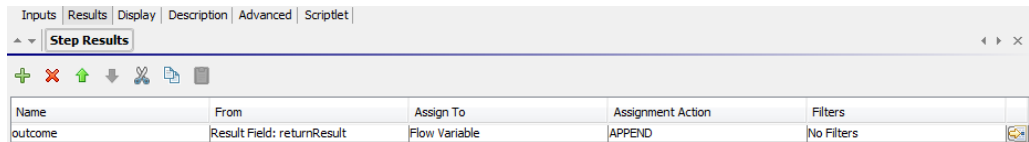


6. Apply the list of targets to each internal step:



- a. For each of the internal steps inside the multi-instance lane, open the Step Inspector and add an input.
  - b. Open the Input Editor, and in the **Assign From** 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:

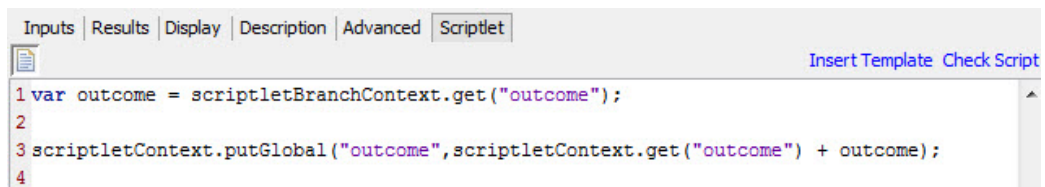
- a. Open the Step Inspector for the multi-instance step by double-clicking the **Multi-step** icon. 
- b. Click the **Results** tab and add a result.
- c. In the **Assign To** column, assign the result to a flow variable.
- d. Give a name to the flow variable that will hold the data, for example, **outcome**.
- 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 Field: returnResult	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

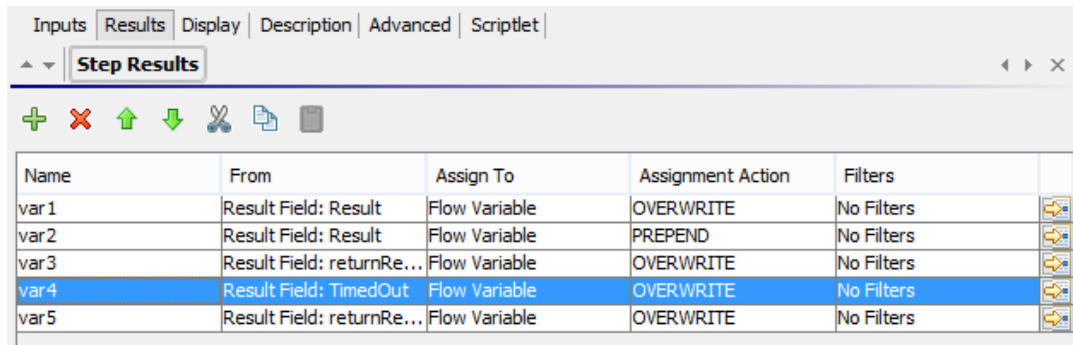
**Multi-instance**  icon.

- 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.



Name	From	Assign To	Assignment Action	Filters
var1	Result Field: Result	Flow Variable	OVERWRITE	No Filters
var2	Result Field: Result	Flow Variable	PREPEND	No Filters
var3	Result Field: returnRe...	Flow Variable	OVERWRITE	No Filters
var4	Result Field: TimedOut	Flow Variable	OVERWRITE	No Filters
var5	Result Field: returnRe...	Flow Variable	OVERWRITE	No Filters


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:

```
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);
```

## Limit (throttle) the number of instances a multi-instance step can run on simultaneously

If running a multi-instance step on many instances simultaneously may slow down your system's performance, you can set a throttle level for the step, by limiting the number of targets the multi-instance step can simultaneously start the flow for.


The performance is related to the target system(s), and whether or not one flow can use all of OOs worker threads.

If throttle is not set, the number of instances is the same as the number of inputs.

If throttle is set, the number of instances is the minimum between throttle size and remaining inputs.

1. Open the Step Inspector for the step, and then click the **Advanced** tab.
2. Under **Execution**, select the **Throttle parallel execution** check box.
3. In the box, type the maximum number of instances of the step that should run at once.


## 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 336](#).

# Using Scriptlets in a Flow

Scriptlets (written in Nashorn or 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

**Note:** When referencing a system property in a scriptlet, you must use the complete path. For example, if there is a system property under the following folder structure `folderA\folderB\my_ci`, use the string `${folderA/folder/my_ci}` to reference it.

## Resources to help you write scriptlets

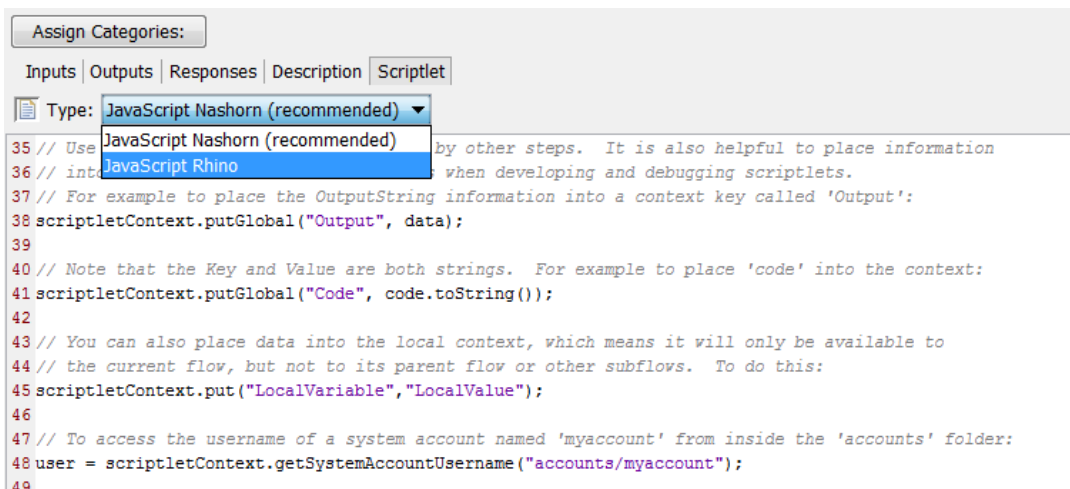
- Scriptlet templates (in Nashorn or Rhino JavaScript) are available in the Scriptlet Editor.
- Default scriptlets are available in the **Configuration\Scriptlets** folder.

- 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.
3. From the **Type** list, select the JavaScript language in which you will write your script: Nashorn (recommended) or Rhino.



```
35 // Use JavaScript Nashorn (recommended) by other steps. It is also helpful to place information
36 // into JavaScript Rhino when developing and debugging scriptlets.
37 // For example to place the OutputString information into a context key called 'Output':
38 scriptletContext.putGlobal("Output", data);
39
40 // Note that the Key and Value are both strings. For example to place 'code' into the context:
41 scriptletContext.putGlobal("Code", code.toString());
42
43 // You can also place data into the local context, which means it will only be available to
44 // the current flow, but not to its parent flow or other subflows. To do this:
45 scriptletContext.put("LocalVariable", "LocalValue");
46
47 // To access the username of a system account named 'myaccount' from inside the 'accounts' folder:
48 user = scriptletContext.getSystemAccountUsername("accounts/myaccount");
49
```

4. Click **Insert Template**.
5. Follow the guidelines in the template to write your script.
6. Click **Check Script** to check for errors.
7. 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 265.
7. Click **Check Script** to check for errors.

If an error is found in a scriptlet, the error is marked with a red underline, as is the flow that contains the scriptlet.



**Note:** Flows containing invalid scriptlets are not included when you create content packs from a project. See ["Exporting a Content Pack" on page 347](#).

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.



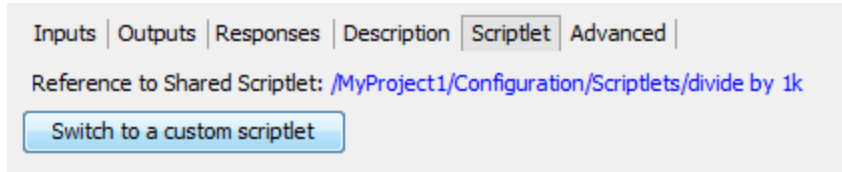
5. Create the scriptlet in one of the following ways:
  - Click **Insert Template** to use the scriptlet template as a basis.
  - Copy and paste the text from an existing scriptlet in another operation or from the **Configuration\Scriptlets** folder.

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.



Assign Categories:


Inputs | Outputs | Responses | Description | Scriptlet

Type: **JavaScript Nashorn (recommended)** ▼

JavaScript Nashorn (recommended) by other steps. It is also helpful to place information  
 JavaScript Rhino when developing and debugging scriptlets.

```

35 // Use
36 // int:
37 // For example to place the OutputString information into a context key called 'Output':
38 scriptletContext.putGlobal("Output", data);
39
40 // Note that the Key and Value are both strings. For example to place 'code' into the context:
41 scriptletContext.putGlobal("Code", code.toString());
42
43 // You can also place data into the local context, which means it will only be available to
44 // the current flow, but not to its parent flow or other subflows. To do this:
45 scriptletContext.put("LocalVariable", "LocalValue");
46
47 // To access the username of a system account named 'myaccount' from inside the 'accounts' folder:
48 user = scriptletContext.getSystemAccountUsername("accounts/myaccount");
49
  
```

GUI item	Description
Scriptlet icon 	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:

<b>Wildcard</b>	<b>Uses</b>
<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
<code>\d</code>	Any digit 0-9
<code>\D</code>	Any non-digit
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\s</code>	Any white space character
<code>\S</code>	Any non-white space character
<code>\t</code>	Tab
<code>\w</code>	Any letter, number, or underscore
<code>\W</code>	Anything except a letter, number, or underscore

The modifiers for regular expressions are:

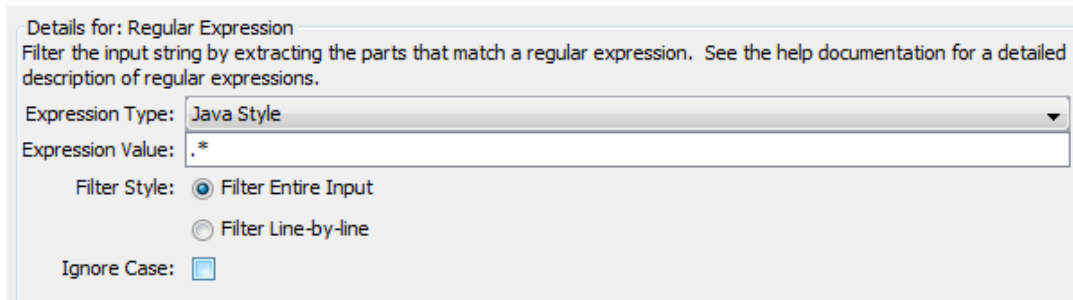
<b>Modifier</b>	<b>Effect</b>
<code>*</code>	Match zero or more
<code>+</code>	Match one or more
<code>?</code>	Match zero or one
<code>{n}</code>	Match exactly n occurrences
<code>{n,}</code>	Match n or more occurrences
<code>{n,m}</code>	Match between n and m occurrences
<code>[abc]</code>	Match either a, b, or c
<code>[^abc]</code>	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 265](#).
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.



3. From the **Expression Type** list, select **Java Style**. Do not use the other styles; they have been deprecated.
4. 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.

5. For **Filter Style**, select **Filter Entire Input** or **Filter Line-by-line**, according to how you want the filter applied to the raw results.
6. To make the regular expression not case-sensitive, select **Ignore Case**.
7. Click **Test Selected Filters** to test the filter.
8. 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	00:00:0 0	Bash
0	R	512	2659	2160 4	0	76	0	-	1110	-	pts/1	00:00:0 0	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**.
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:

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.

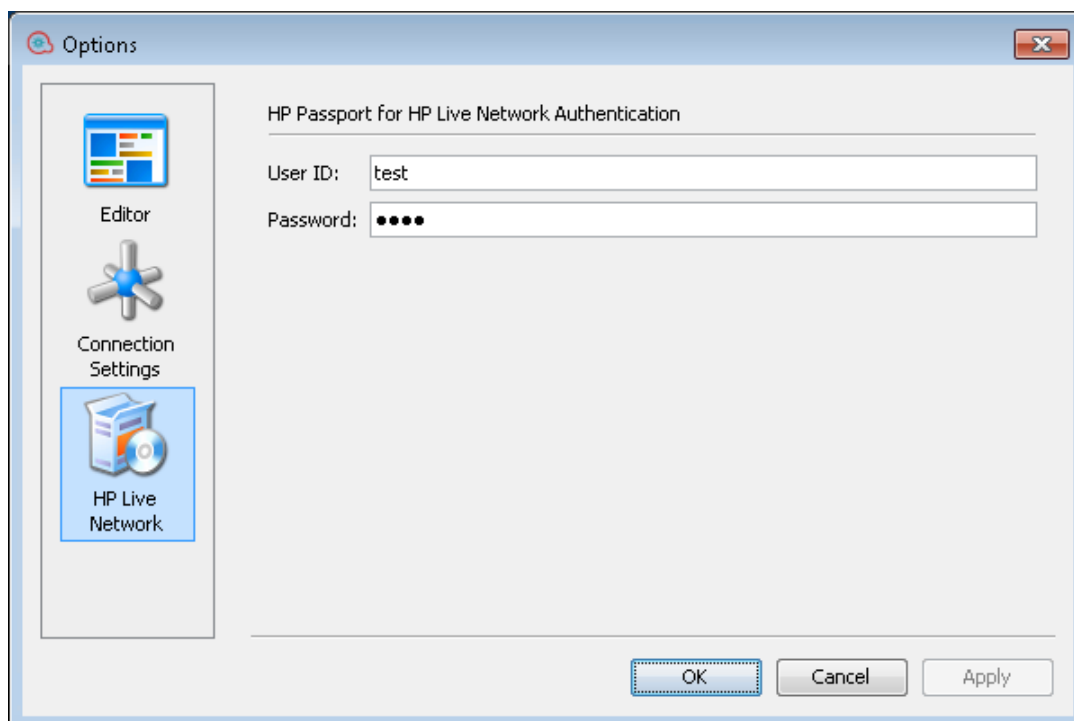
## Searching Content on HP Live Network from Studio

A Studio author can search for relevant information and content on the HP Live network based on their HPLN profile access permissions directly from Studio.

### Setting up the HPLN Connection from Studio

1. From the **Configuration** menu, select **Options > HP Live Network**.
2. In the **HP Live Network Settings** dialog box, enter your HP Live Network user id and password.

**Note:** The user id and password that you enter in this dialog box will determine the access permissions and will affect the search results.

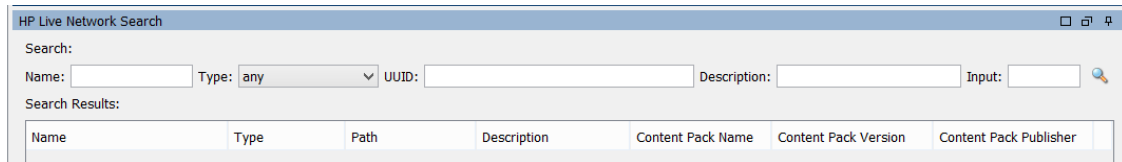


3. Click **Apply** or **OK**.

## Searching HP Live Network from Studio

The HPLN Search pane is located at the bottom of the Studio Workspace and is open by default.

Enter a search criteria and then press Enter or click the Search icon.



Name	Type	Path	Description	Content Pack Name	Content Pack Version	Content Pack Publisher
------	------	------	-------------	-------------------	----------------------	------------------------

HP Live Network searches returns a maximum of 100 results.

**Note:** \, (, ), & characters are not supported inside the name, description, input, UUID for HP Live Network searches.

## Search Types

The following search types are available:

- **Search by type:** You can search for a flow, operation, system account, system property, selection list, domain term, system evaluator, system filter, scriptlet, group alias, category, and role alias.
- **Search by name:** Searches by name. You can use wild card characters \* for incomplete words. The wild card \* matches anything. If no name is specified then any name is a match. This search is case insensitive.
- **Search by description:** Searches the descriptions. You can use wild card characters \* for incomplete words. The wild card \* matches anything. If no name is specified then any description is a match. This search is case insensitive.
- **Search by input name:** Searches the input name. You can use wild card characters \* for incomplete words. The wild card \* matches anything. It will return all items except for configuration items inside all content packs with specific meta-data of each that have inputs matching the search string. This search cannot be used for configuration items.
- **Search by UUID:** Enter the UUID as an exact string. The search is matched based on = operator. This search will return the flow, operation or configuration item with that UUID.
- **Combined search:** Allows you to search with multiple criteria. This includes two or more criteria of type, name, description, input using the AND operation.

**Note:** A search type with any and all other fields empty is not allowed. In this case the search button is disabled.

## Search Results

The results are displayed in a table which you can sort according to the column name. In addition you can filter the results using multiple values.

When you right-click on a row you can view the full description of the search result or connect directly to the location of the content on HP Live Network.

HP Live Network Search

Search: Name:  Type:  UUID:  Description:  Input:

Search Results:

Name	Type	Path	Description	Content Pack Na...	Content P...	Content Pack Publis...
Operating System Dete...	operation	Library/Utility Operations	<pre> Detects the operating system that is ...	HP Operations Or...	1.0.142	Hewlett-Packard
Operating System Dete...	operation	Library/Utility Operations	<pre> Detects the operating system that is ...	HP Operations Or...	1.0.121	Hewlett-Packard
Operating System Dete...	operation	Library/Utility Operations/Deprecated	<pre>OS Detector Connects to a specified h...	HP Operations Or...	1.0.142	Hewlett-Packard
Operating System Dete...	operation	Library/Utility Operations/Deprecated	<pre>OS Detector Connects to a specified h...	HP Operations Or...	1.0.121	Hewlett-Packard
Operating System Dete...	operation	Library/Utility Operations/Deprecated	<pre>OS Detector Conn... View description	HP Operations Or...	1.0.142	Hewlett-Packard
Operating System Dete...	operation	Library/Utility Operations/Deprecated	<pre>OS Detector Connects to a specified h...	HP Operations Or...	1.0.142	Hewlett-Packard
Operating System Dete...	operation	Library/Utility Operations/Deprecated	<pre>OS Detector Connects to a specified h...	HP Operations Or...	1.0.121	Hewlett-Packard
Get Operating System	flow	Library/Integrations/Microsoft/Syst...	<pre>Runs the powershell command "Get-...	HP Operations Or...	1.0.111	
Get Virtual Machine Op...	operation	Library/Integrations/Microsoft/Hype...	<pre>Gets the operating system version of ...	HP Operations Or...	1.0.111	
Get Supported Guest O...	operation	Library/Integrations/Hewlett-Packar...	Returns a list of supported guest operating s...	SA Content Pack ...	1.0.0	
Get Supported Guest O...	operation	Library/Integrations/Hewlett-Packar...	Returns a list of supported guest operating s...	SA Content Pack ...	1.0.1	
Get Supported Guest O...	operation	Library/Integrations/Hewlett-Packar...	Returns a list of supported guest operating s...	SA Content Pack ...	1.0.2	

HP Live Network Search

Search: Name:  Type:  UUID:  Description:  Input:

Search Results:

Name	Type	Path	Description	Content Pack N...	Content ...	Content Pack Publis...
Enable CDP	flow	Library/Accelerator Packs/Network...	<pre>Enables CDP on a Cisco router.Input ...	Cisco	1.0.112	
Enable All Interfaces	flow	Library/Accelerator Packs/Network...	<pre>Enables all network interfaces of a C...	Cisco	1.0.112	
Enable RIP for All Sub...	flow	Library/Accelerator Packs/Network...	Adds all locally connected networks to be a...	Cisco	1.0.112	
Enable Routing	flow	Library/Accelerator Packs/Network...	<pre>Enables routing on a Cisco router.In...	Cisco	1.0.112	
Set Enabled Secret fo...	flow	Library/Accelerator Packs/Network...	<pre>Sets the enabled secret on all route...	Cisco	1.0.112	
CDP Enable	operation	Library/Operations/Network/Cisco/...	<pre>Enables the CDP on all supported int...	Cisco	1.0.112	
CDP Interface Enable	operation	Library/Operations/Network/Cisco/...	<pre>Enables the CDP on a specific interfa...	Cisco	1.0.112	
Enable Interface	operation	Library/Operations/Network/Cisco/...	<pre>Enables the specified interface on a ...	Cisco	1.0.112	
Enable Local Logging	flow	Library/Operations/Network/Cisco/...	<pre>Enables logging to RAM on a Cisco r...	Cisco	1.0.112	
Enable Local Logging	flow	Library/Operations/Network/Cisco/...	<pre>Enables logging to RAM on a Cisco r...	Cisco	1.0.112	
Enable DNS Client	operation	Library/Operations/Network/Cisco/...	<pre>Enables the DNS client on the router ...	Cisco	1.0.112	
Enable Routing	operation	Library/Operations/Network/Cisco/...	<pre>Enables routing on a Cisco router.In...	Cisco	1.0.112	

**Note:** If you want to find a specific item from the latest released content pack, you can filter by the content pack name and then sort descending by the content pack version. The first items in the list are from the latest released version.

HP Live Network Search

Search: Name:  Type:  UUID:  Description:  Input:

Search Results:

Name	Type	Path	Description	Content Pack Name	Content Pack Version	Content Pack Publisher
Modify User	operation	Library/Integrations/Hewlett-Packard/Pr...	<pre>Modifies an existing user ...	HP Operations Orchestration HP Solutions CP	1.0.202	Hewlett-Packard
Modify User	flow	Library/Integrations/Hewlett-Packard/On...	<pre>Modify an existing user pa...	HP Operations Orchestration HP Solutions CP	1.0.202	Hewlett-Packard
Modify User	operation	Library/Integrations/Hewlett-Packard/Pr...	<pre>Modifies an existing user ...	HP Operations Orchestration HP Solutions CP	1.0.134	Hewlett-Packard
Modify User	flow	Library/Integrations/Hewlett-Packard/On...	<pre>Modify an existing user pa...	HP Operations Orchestration HP Solutions CP	1.0.134	Hewlett-Packard
Modify User	operation	Library/Integrations/Hewlett-Packard/Pr...	<pre>Modifies an existing user ...	HP Operations Orchestration HP Solutions CP	1.0.117	Hewlett-Packard
Modify User	flow	Library/Integrations/Hewlett-Packard/On...	<pre>Modify an existing user pa...	HP Operations Orchestration HP Solutions CP	1.0.117	Hewlett-Packard



## hpln-index-generator Tool

The hpln-index-generator tool, located in `<oo_install_folder>/studio/tools/lib/hpln-index-generator.jar`, generates an index file describing the flows, operations and configuration items and with the corresponding metadata inside a content pack.

## Validating Content

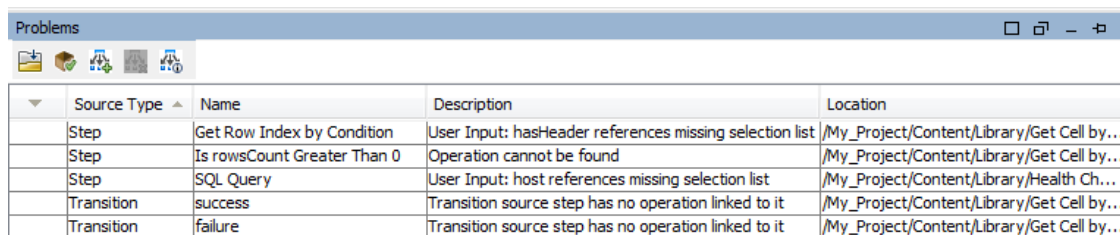
Before releasing your content, it is important to test and validate the flows in your project. Studio provides the following tools to help you do this:

- 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.



The screenshot shows the 'Problems' pane in Studio. It contains a table with the following data:

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...

### *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.
3. Double-click a row in the **Problems** pane to open the item for editing.

**Note:** To open multiple items, select them using the SHIFT or CONTROL keys, right-click, and select **Open**.

### Validate all the flows and operations in a project

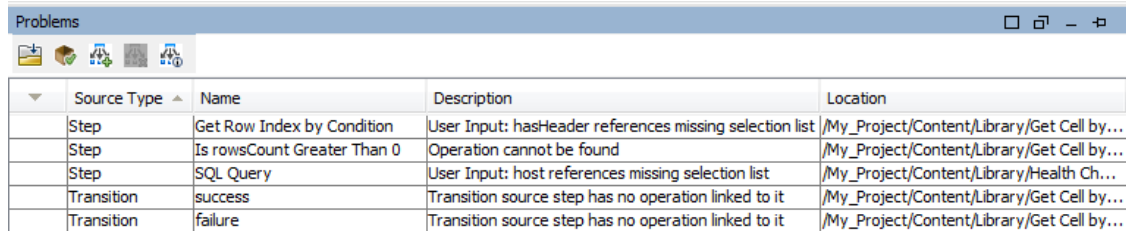
1. Open the project that you want to validate.
2. From the **Tools** menu, select **Validate Flows and Operations**.

A list of any problems appears, with their locations and descriptions, to guide you in repairing the problems.

## Reference Material

### Problems pane

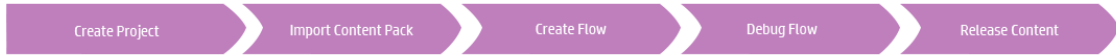
The **Problems** pane, which you open with the **Problems** tab on the lower edge of the Studio window, lets you check whether a selected flow or operation is valid.



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...

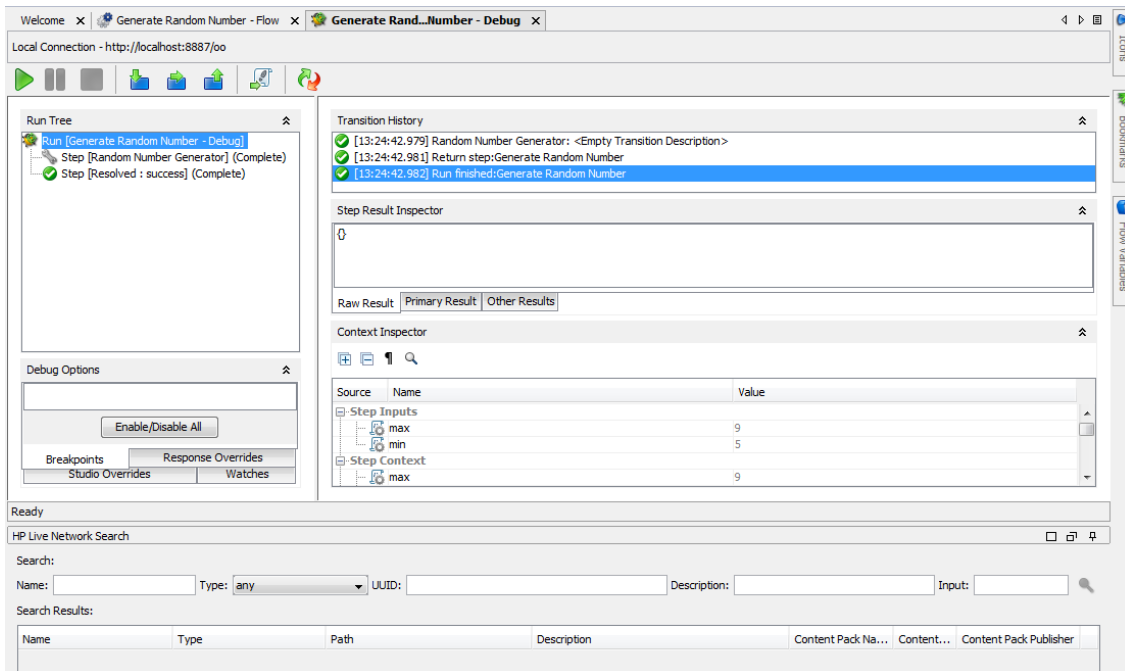
GUI item	Description
<b>Source Type</b>	Displays the type of the element in which there is a problem.
<b>Name</b>	Displays the name of the element in which there is a problem.
<b>Description</b>	Describes the problem, to guide you in how to repair it.
<b>Location</b>	Displays the location of the element 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.

From each step in the debugger, you can jump directly to the relevant step in the flow that opens in the Flow Editor.

### Note:

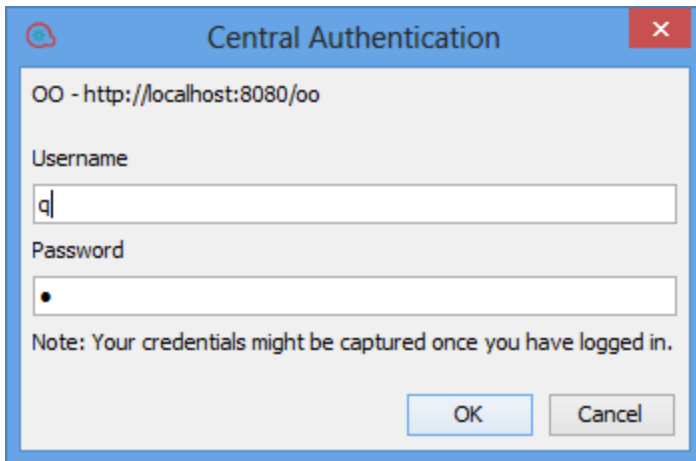
- Values resulting from encrypted or hidden inputs are retrieved with \*\*\*\*\*characters. You must edit these values in the Context Inspector before triggering the flow, otherwise the

value of the applied input will be \*\*\*\*\* during the new run.

- In the Context Inspector, you can see the names of hidden outputs, but their values are shown as obfuscated \*\*\*\*\*.

## Debugging Flows that Require User Authentication

If authentication is enabled for a flow in Central (in the **Enable Authentication** check box), when you select to debug the flow, you are prompted to enter user credentials in the Central Authentication dialog box. After they have been entered once, the **Username** and **Password** credentials are remembered for the current session, and you do not need to reenter them when debugging additional flows.



For details on the authentication settings, see "Setting up Security Settings" in the Central Guide.

**Note:** If authentication has been enabled in Central, you cannot log in with a user that has a colon as part of the user name.


## Best Practices

It is recommended to debug subflows before debugging their parent flows.

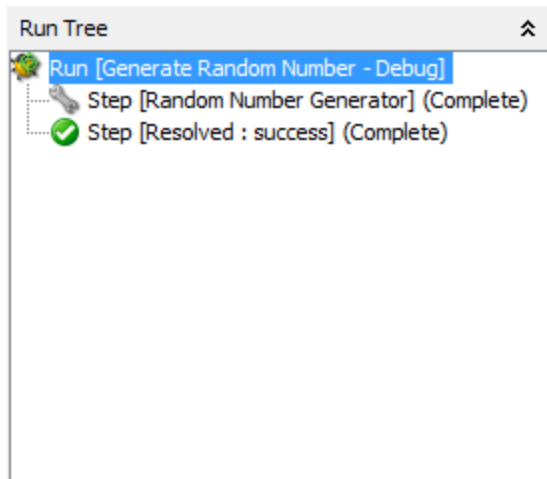
## What do you want to do?

### 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.

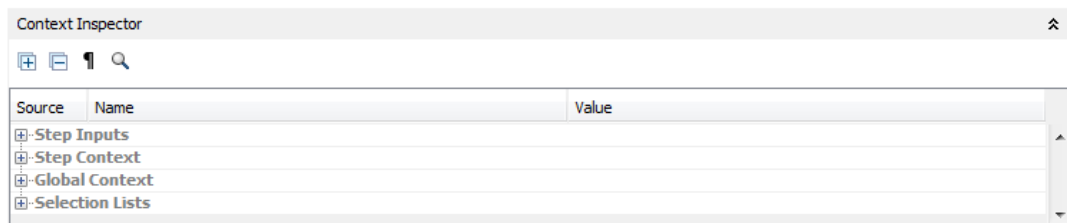
2. To run the flow to its end, click the **Play**  button in the debugger toolbar.
3. To see the information for a completed step, click the step in the **Run Tree** pane.




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

These actions allow a flow developer to step in and out of the running of a subflow, while debugging a flow. Both actions are available when the debugged flow is paused, waiting for the user's action.

**Note:** It is recommended to debug subflows before debugging their parent flows.



- To step into a step's subflow, click the **Step Into**  button. The debugger will start running the subflow and will pause on the first step of the subflow.

Clicking **Step Into** at the beginning of the parent flow tells the debugger to pause on the first step of the parent flow.

- To step out of the subflow, click the **Step Out**  button. The debugger will run the rest of the steps in the current subflow invocation and will pause on the first step following the subflow (in the parent flow). If the current step is in the parent flow, the action will behave like a resume action.

## 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 pane.
- 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

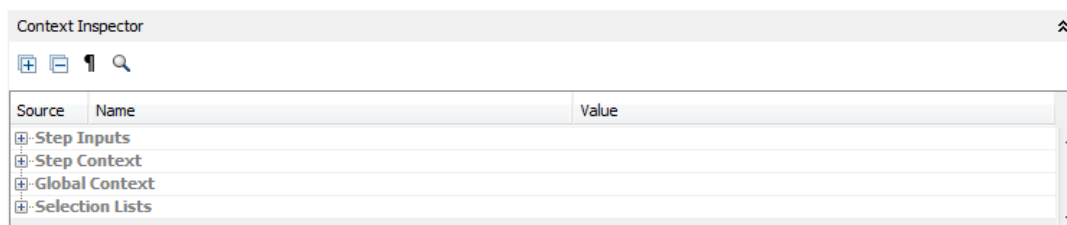
To see how a flow behaves with different values for its flow variables, you can change the value of a flow variable before running a step.

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.

You can search, sort, and filter by columns.



Source	Name	Value
+	Step Inputs	
+	Step Context	
+	Global Context	
+	Selection Lists	

- 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.

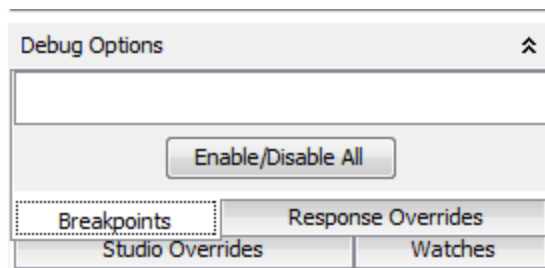
1. Open the flow open in the authoring pane, and right-click the step where you want to set the breakpoint.
2. Select **Debugging > Set Breakpoint**.

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 **Debug Options** 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.

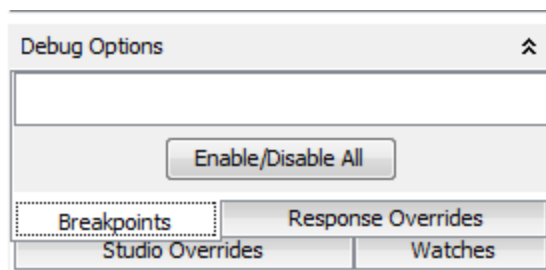
For example, if you have a step in a flow for which you don't have the necessary information, you might want to test the rest of the flow, regardless of the certain failure of that step. You can force the run to follow the response and transition that you want, rather than the failure response that would 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.

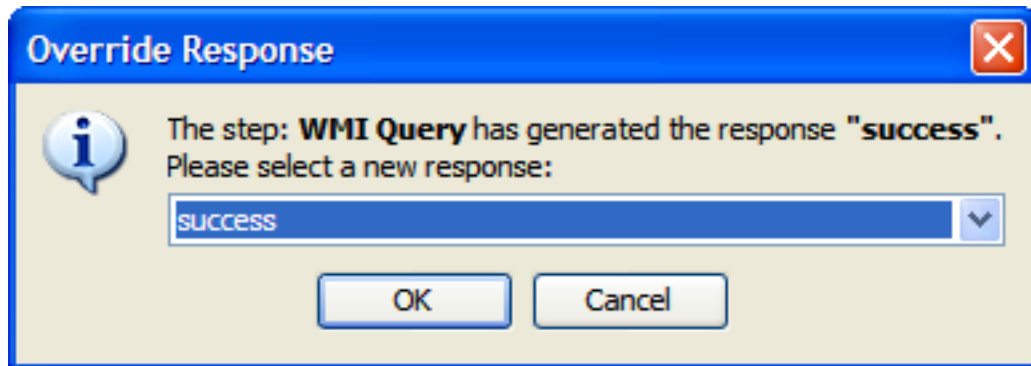
In the debugger **Debug Options** pane, the **Response Overrides** tab shows the existing response overrides.



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.



### Jump to a flow step from the Run Tree

- Right-click on a step in the Run Tree and select **Go to step**

or:

Click the **Go to step** button in the toolbar .

The Flow Editor opens showing the current flow with the step selected.

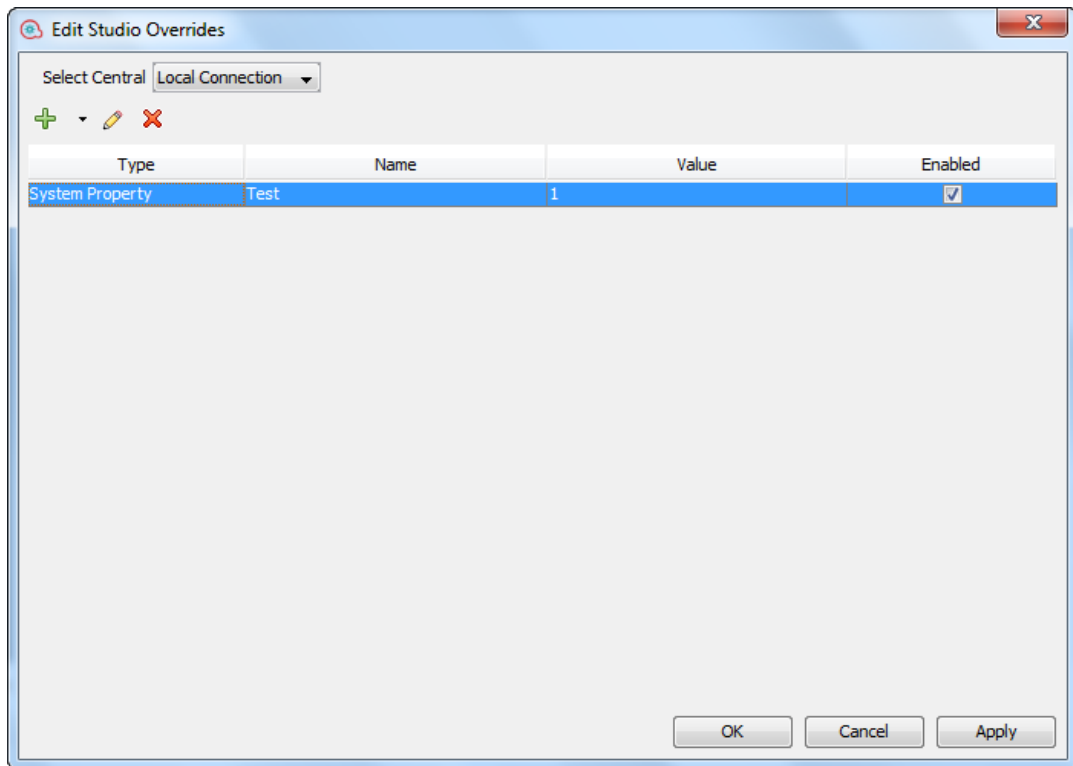
### Setting Overrides in Studio

In Studio you can define overrides per Central connection, for system properties and system accounts. Creating overrides allows you to change configuration items, system properties, and system accounts (from content packs), before triggering and without changing values in the context inspector. For example you can use these overrides to adjust a read-only value that was imported from a content pack.

**Note:** You can enable or disable individual overrides. Overrides are unique by name and type, and take precedence over Central overrides.

To access the **Studio Overrides**:

1. From the **Configuration** menu, select **Studio Overrides**.



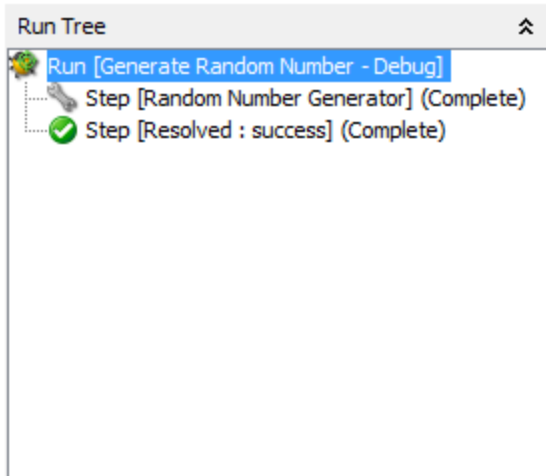
2. Select the Central connection from the **Select Central** drop-down list. When a Central is selected, the overrides of that Central are displayed.
3. You can perform the following:
  - Add a system property or system account override.
  - Edit the selected system property.
  - Delete the selected system property.
4. Click **Apply**, to apply the changes and **OK** when you are done.

**Note:** If you override a system property with a blank value, the default value from the **System Properties** definition is used and not the blank value.

**Note:** When referencing a system property, you must use the complete path. For example, if there is a system property under the following folder structure **folderA\folderB\my\_ci**, use the string **`\${folderA/folder/my\_ci}** to reference it.

## Reference Material

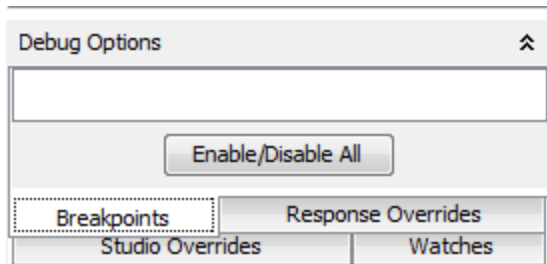
### Run Tree pane



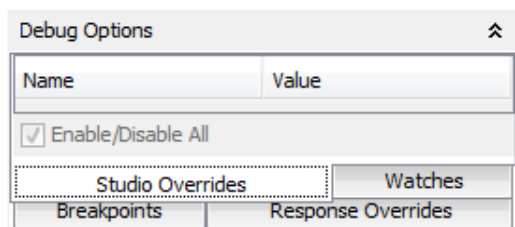
The **Run Tree** pane shows each step that runs, including steps in subflows of the flow.

Steps that will run simultaneously in an actual execution are run in a serial sequence in the debugger.

### Debug Options pane



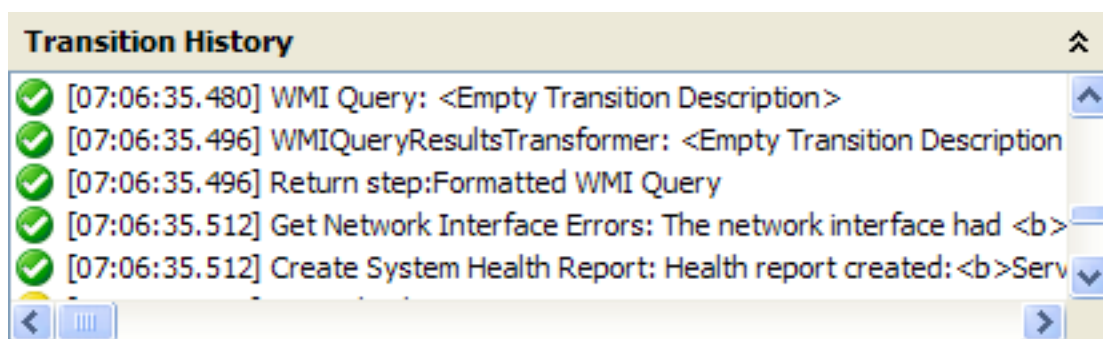
- **Breakpoints** are flags that enable you to automatically pause a run at a certain step in order to examine the results, the path of the run, or the values in the flow variables at a that point.
- **Response Overrides** force the response that you select, regardless of the result of the operation.
- **Studio Overrides** displays the enabled Studio overrides. In this view, you can enable and disable them as required.



- **Watches** enables the user to create a watch list of variables during the debug. These variables are entries of the various contexts: Step Inputs, Step Context, Global Context, and the Configuration Items: Selection Lists, System Properties, and System Accounts. In the Watch list You can add a variable by name, and all entries with that name from the various context and from the lists of Configuration Items will be shown in the watches.

The **Debug Options** 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.

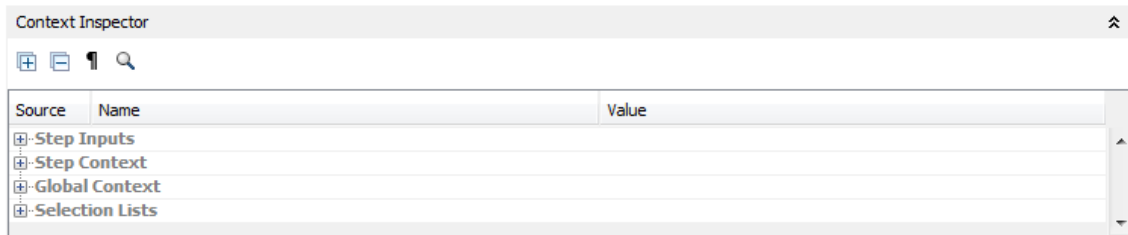
### Step Result Inspector pane



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.

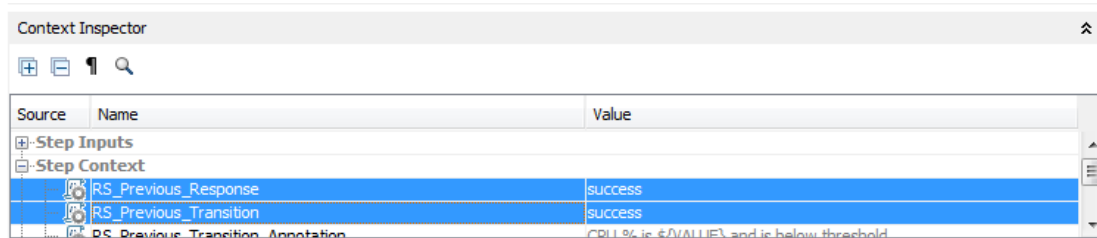
## Context Inspector pane



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.

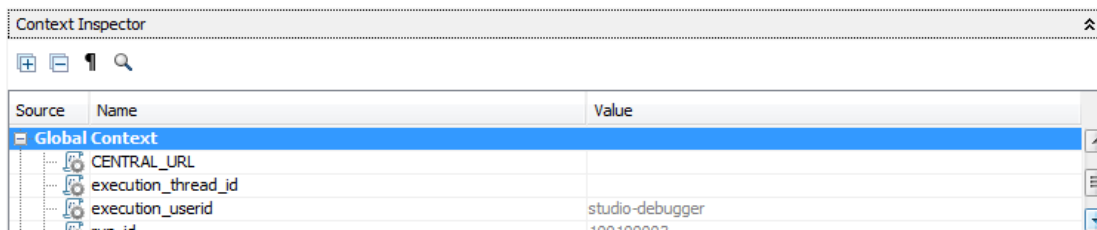
- The values in the **Step Input** section are the values that were assigned to the input before the step started. The text boxes that contain the values for flow variables are color coded.



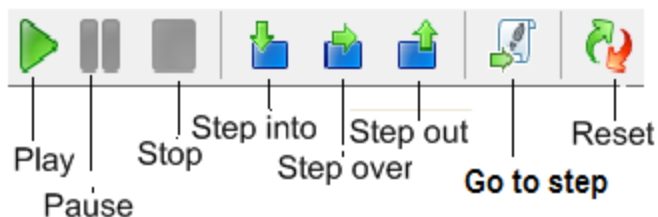
- The values in the **Step Context** section are the values that were updated after the step began.










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>Go to step</b> 	Jump to a flow step from the run tree	
<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 295](#).

### 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 299](#).

## Debugging a Remote Central with Studio

Studio Remote Debugging allows an HP OO user to troubleshoot and debug flow runs in a remote Central. This allows customers to use Studio when investigating issues in their Central environments without having to manually deploy fixes and modify the Central Flow Library.

Before debugging a flow, the user defines the URL to connect to Central. For more information, see the *HP OO Security and Hardening Guide*.

You can debug the existing flows, locally changed flows or new flows. Studio takes the flow being debugged, along with all the subflows and operations (existing, changed or new) and sends them to Central for execution.

All the debugging or execution of the flow occurs on the selected Central using a merge of the Studio configuration items, Central configuration items and Context Inspector values.

Configuration items can be overwritten either from Studio or from Central. Upon triggering a flow (just before the flow starts), HP OO determines the initial value of the configuration items as following:

- If the user has changed the configuration item in the Context Inspector, this new value is used regardless of any overrides.
- If a configuration item is not overwritten (neither Central nor Studio) and was not changed in the Context Inspector, the value is used as is.

- If a configuration item is overridden in Central, but not in Studio overrides, the value is the one from Central.
- If a configuration item is overridden in Studio overrides, the value is the one from Studio overrides.

During the debug, when you pause, resume, step over, and so on, the user can change values with the Context Inspector as needed.

**Note:** The only configuration items that can be overridden in Central are system properties and system accounts. Override settings for configuration items in new system accounts are used.

Changes of the flows and configuration items do not affect or modify the Central library. These items are volatile, meaning that they are only visible for the debug session.

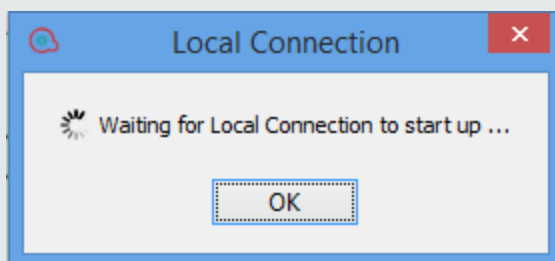
The traces left by a Remote Debugging session are visible in the Central Run Explorer (drill-down), and Dashboard, as any other execution. The source for such traces can be eventually identified as originating from a debugger execution and can be filtered out. The drill-down reflects the flow sent from the Studio (with the eventual changes).

During the debug session, the user can change the context of the execution through the Context Inspector.

**Note:** All runs started from the Remote Debugger have the Extended persistence level.

For details on run persistence, see "Persistence Level for the Run Log" in the *HP OO Central User Guide*.

**Note:** In some cases, Studio may start up faster than the time it takes to initialize the local connection. In these cases, if you start debugging using the local connection right after Studio starts up (the **Local Connection** button shows a progress icon), you will see the following message:



## Prerequisites

- **Align Content Packs**

To start the Remote Debugging session, you need to import into the Studio workspace the Central relevant content packs, with the appropriate version. Therefore, any content packs used by a debugged flow must have been deployed to Central before starting the debug session.

- **Group aliases**

If you created a new group alias in Studio (and used them with certain operations), you will need to manually configure the groups in Central. The debugger cannot decide to which worker group the group alias should map. If the assigned group of an operation is not found (during the Remote Debugging execution), it will behave as if it was triggered from Central, pause and prompt the user about the problem.

- **Authorized Users**

Only users that have been assigned the **Remote Debugging** permission are allowed to trigger the **Debugger** on a remote Central. See the *Central User Guide* for more information.

**Note:** A user who has remote debugging permission is exposed to all flows, but not to system accounts. The system accounts are only available to users with appropriate permissions.

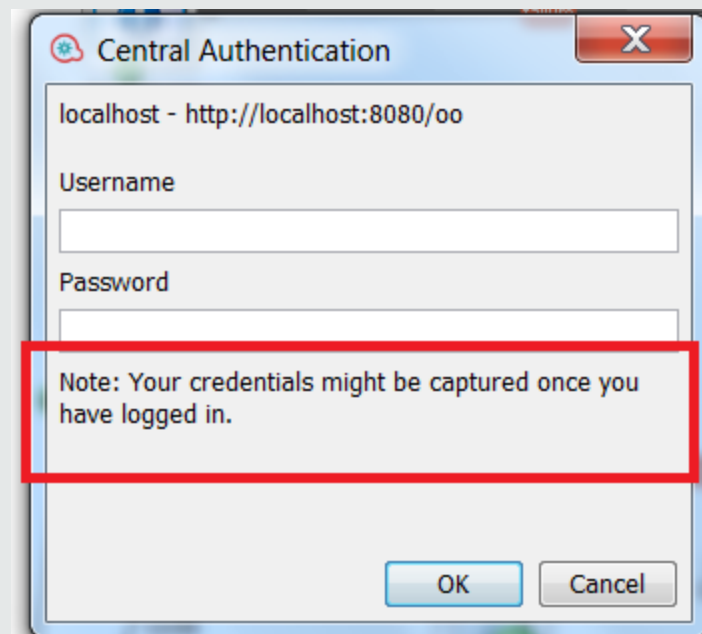
**Note:** This restriction does not apply when debugging using the local connection (the default connection that Studio uses if you do not configure a remote Central).

## Typical Workflow

1. Assign to a user a role containing the **Remote Debugging** permission.
2. Set up the Studio workspace to match the Central Library (content packs, flows to be debugged).
3. Configure, if necessary, the Central connection.
4. (Optional) Change the flow before debug.
5. Select a connection from the debug option. When you click the debug option, you are prompted for credentials. These credentials are saved and remembered in Studio.

**Note:** If the **Enable Capture of Logged-in User Credentials** check box is selected in

the **System Configuration Workspace > Security > Settings** option in Central, the following warning message opens in the Studio remote debugger:




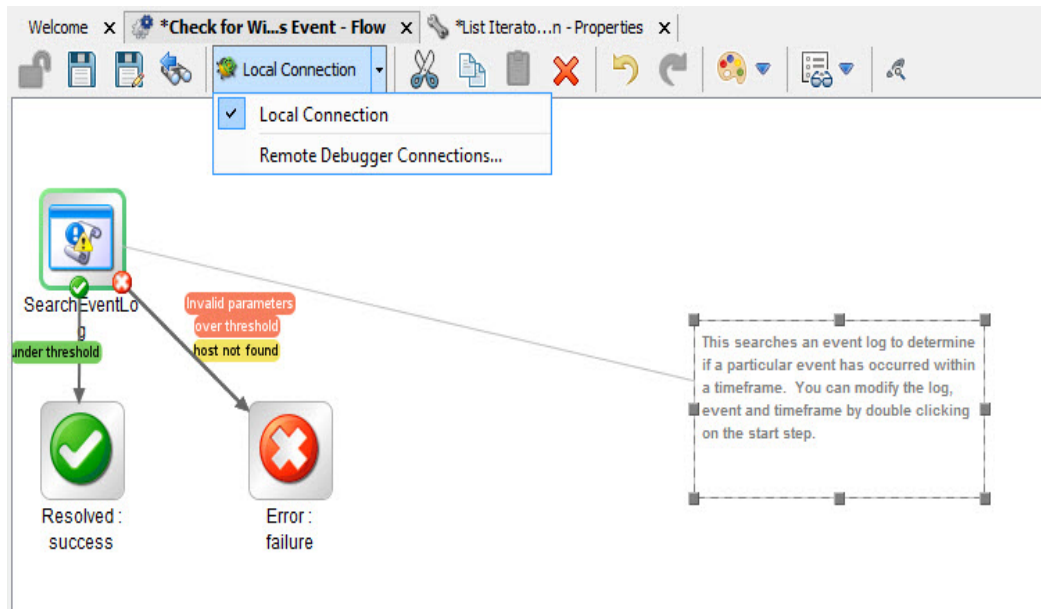
**Note:** If authentication has been enabled in Central, you cannot log in with a user that has a colon as part of the user name, when logging into Central for remote debugging.

6. Execution runs with a merge of the **Studio Configuration Items**, **Central Configuration Items** and **Context Inspector**. For more information on the Context Inspector, see "[Validating Content](#)" on page 322.
7. (Optional) Set breakpoints, and change variables and configuration items by the context inspector during debug.
8. Execution completes.

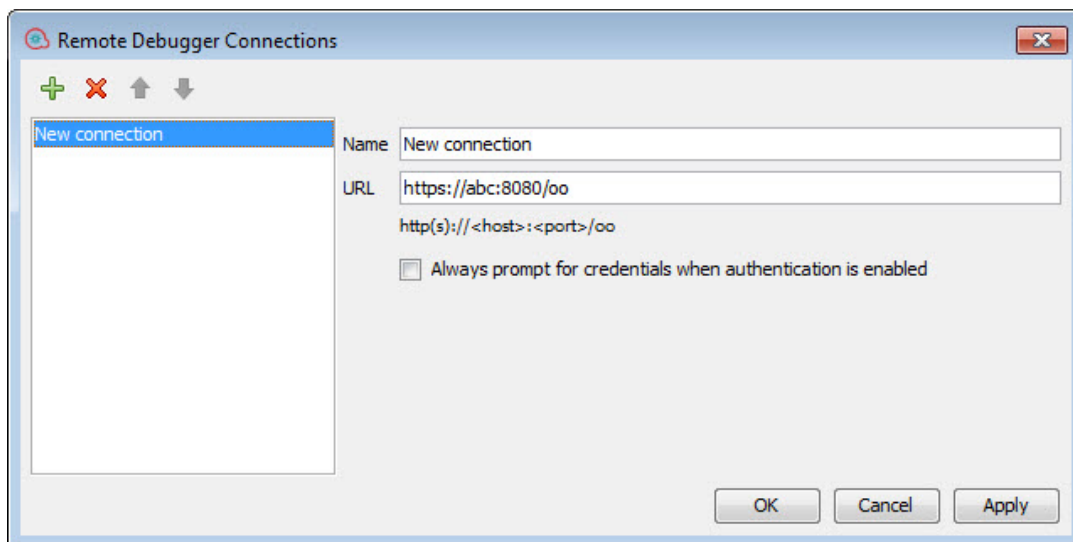
## What do you want to do?

### Add or Edit Central Connections

1. In Studio, select **Configuration > Remote Debugger Connections**.
2. Click the debug icon , and from the list select a connection to a remote Central. **Local Connection** is the default connection.



3. To add, edit or delete a connection, select **Remote Debugger Connections**. The **Remote Debugger Connections** dialog box contains a list of available connections. You can edit these connections, delete existing connections and add new connections.



- **Delete:** Select the connection that you want to delete and either press the delete key or click the minus (x) red button.
- **Add:** To add a new connection, click the green (+) button. Enter a name for the new connection and the URL of the remote Central.

**Note:** There is no validation with the connection name and URL. The user is required to

check these settings. You must assign a unique connection name.

**Always prompt for credentials when authentication is enabled:** When you select this option and the selected Central has authentication enabled, it causes Studio to always prompt the user for credentials; if it has already been authenticated, the authentication form will have the credentials already filled in.

**Note:** If the user needs to reset the user name or password without restarting Studio, in the **Remote Debugger Connections**, check **Always prompt for credentials when authentication is enabled**.

**Configuring LDAP user and domain:** When you create a new connection to Central and connect using a LDAP user, you are prompted to select the domain, username and password.

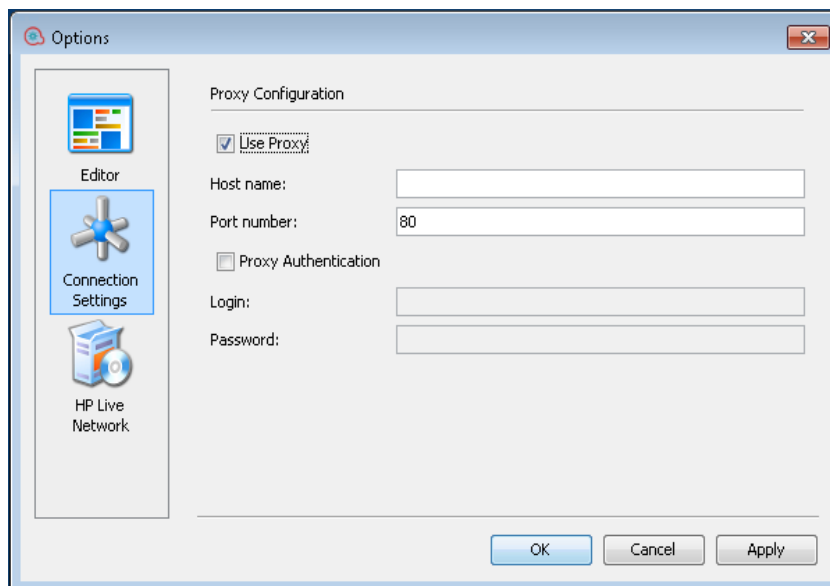
**Note:** You cannot configure the Remote Connections in the **studio.properties** file.

## Set the Proxy for Debugging on a Remote Central

When configuring the connection to a Central, you can set the HTTP proxy information: host, port, user name and password. The proxy supports the basic authentication scheme.

To setup the proxy:

1. In Studio from the **Configuration** menu, select **Options > Connection Settings**. The **Proxy Configuration** dialog box appears.



2. Enter the proxy information. You must enter a host and port number. When you are done, click **Save**. The proxy settings are validated on the host and port number.

## Import Certificates Automatically with a Remote Debugger Connection

1. When you select a remote Central connection using HTTPS, you are prompted with the SSL Certificate message.

The SSL Certificate message displays the certificate's issuer distinguished name, which identifies the entity that signed the certificate and the certificate's subject distinguished name on separate lines.

If the certificate is not trusted, the dialog asks whether you want to trust this certificate.

**Note:** Studio supports both self-signed and CA signed certificates.

- If you want to trust the certificate, click **Connect**. The Studio trustStore is updated with the certificate(s), and you will be able to connect to Central having a successful SSL handshake.
  - If you don't want to trust the certificate, click **Cancel**. The Studio trustStore will not be affected and the connection to Central will fail.
2. If the URL of the server and the certificate do not match, the Hostname Validation dialog box is shown:
    - If you click **Don't Trust**, the connection is not permitted.
    - If you click **Always Trust**, the hostname validation failure is ignored and the connection is allowed.

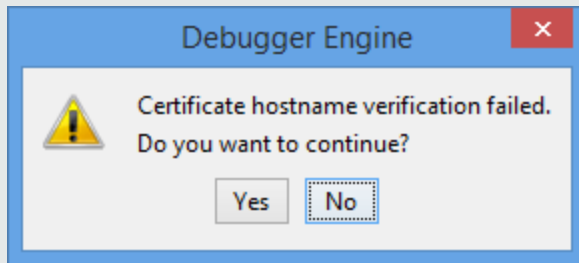
However, when you restart Studio, the hostname validation message will be displayed again, because you are expected to correct this issue. You should correct the URL to match the hostname in the certificate, or change the certificate to be according the connection URL.

**Note:** The SSL Certificate dialog box also appears when you connect to a Git server via HTTPS.

For SVN, the certificate chain details are presented in the studio.log.

**Note:** In the **studio.14j.ini** file under the Studio installation folder, the default values for **self-signed** is false, and the default value for **verify.hostname** is true.

If the hostname verification fails, an error message opens. You can choose to continue or abort the operation.



## Debugging a Flow on a Remote Central

1. Build a flow using operations and flows from an imported content pack or use an existing one.
2. Open the flow.
3. Run the Debug and select a remote connection that you setup earlier.

**Note:** Performing a debug on a Central that is part of a production system could be problematic as it can affect the production data and operation. The Debugger view displays a banner with the name of the remote connection.

## What do you want to do?

### Debug a Flow on a Remote Central

To debug a flow:

1. Build a flow using operations and flows from an imported content pack or use an existing one.
2. Open the flow.
3. Run **Debug** and select a remote connection that you setup earlier.

**Note:** Performing a debug on a Central that is part of a production system could be problematic as it can affect the production data and operation. The Debugger view displays a banner with the name of the remote connection.

### Reuse Existing Flow Inputs

You can use the inputs of a run as input for a new run. This allows you to troubleshoot failed flows, without having to re-enter or prepare the inputs.

To load or specify a run ID from the current Central for re-running a flow:



1. In Studio, open a flow for debugging.
2. Select a remote Central.
3. Enter the inputs of an execution.
4. Run the flow.
5. Click **Load Execution Inputs**.

A dialog box appears with a text field that displays the execution id. Studio loads the inputs of the specified execution id (they are loaded from the currently selected Central). If there are no inputs, then the execution did not include any inputs or the execution was not found. In this case a message appears stating that the current execution has not changed.

After the inputs have loaded, they are matched (based on name) against the inputs of the current flow. The existing inputs of the flow are updated (no new inputs are added) only if the corresponding loaded value is not empty. If no input of the current execution is changed, a message is displayed.

The context inspector highlights the modified inputs and displays the new values.

6. You can now trigger the run with the inputs of the specified execution id.

Inputs are loaded as string values regardless of their type (single value, list of values).

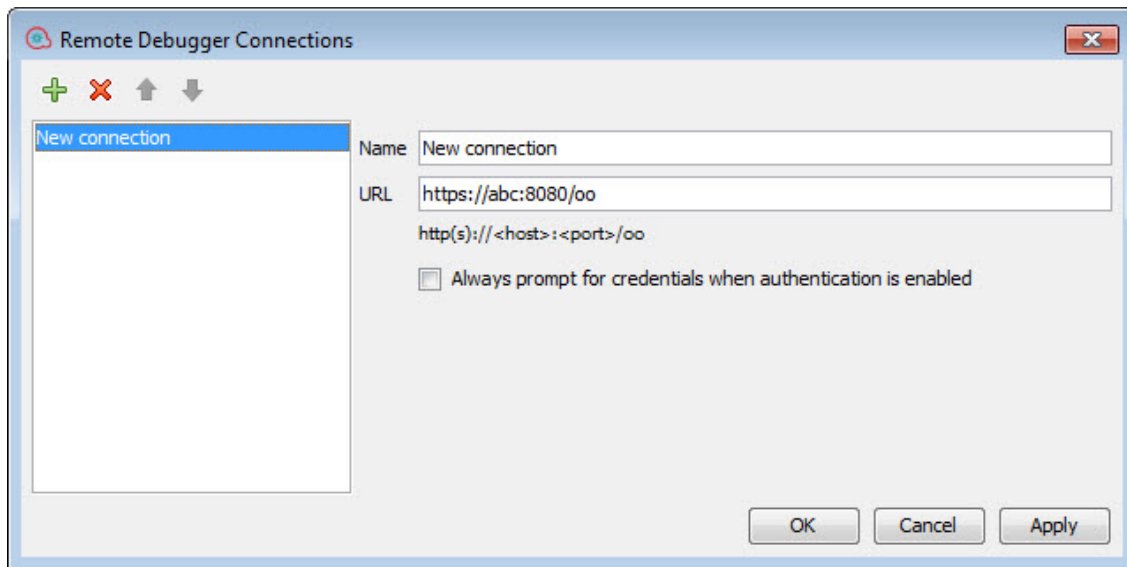
## Rerun a Flow





You can rerun a triggered flow using the same inputs as the previous run. In the debug view, the **Rerun** option is disabled initially and enabled after the first run of this flow. When you click **Rerun**, the toolbar changes to the view ready for debug. When you click **Play**, you are no longer prompted for eventual prompt inputs of the flow. The new run will use the flow inputs of the previous run.

## Reference Material

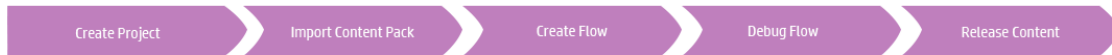
### Remote Debugger Connections

The **Remote Debugger Connections** dialog box contains a list of available connections.



GUI item	Description
<b>Add connection</b> 	Adds a new connection. Type the name and URL of the new connection and click OK.
<b>Delete connection</b> 	Deletes the selected connection.
<b>Move up</b> 	Moves the selected connection higher in the list.
<b>Move down</b> 	Moves the selected connection lower in the list.

## 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 in HP OO Central.

**Note:** Invalid flows or operations will not be included in the content pack.

When you create a content pack from a project, by default, Studio names it with a unique version number. Then, every time you create another content pack from the same project, Studio assigns it with the next minor version number. The naming mechanism is dependent on various conditions and is described in detail in ["Content Pack Versioning Lifecycle"](#) below.

**Note:** Your content pack can include folders inside the configuration items.

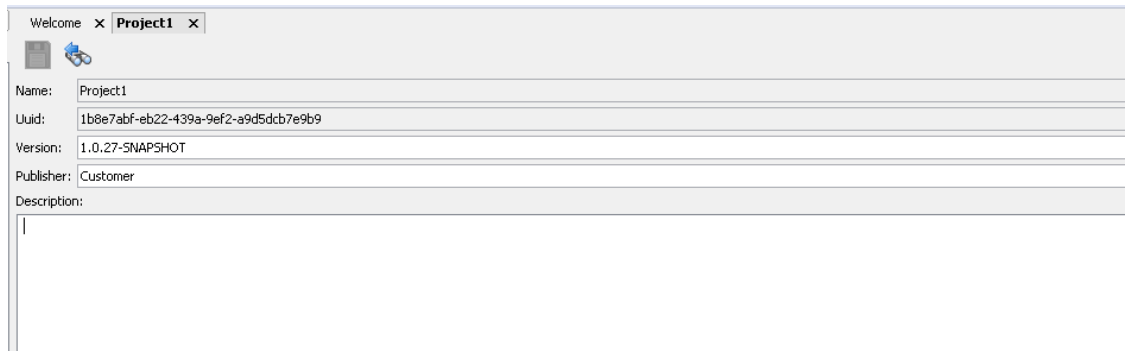
**Note:** Content packs created with a Studio version above 10.20 from projects that contain folders inside a configuration section (e.g., System Properties, System Accounts) cannot be imported into versions of Studio earlier than 10.20.

However, content packs created with a Studio version above 10.20 from projects that have no folders in any configuration section, can be imported into versions of Studio earlier than 10.20.

## Content Pack Versioning Lifecycle

By default, Studio handles the versioning of content packs automatically. After a content pack is created, the last digit of the version number (generally the minor-minor section) is incremented by 1 and a –SNAPSHOT suffix is added to the current project version. This means that when you work on the project, you are adapting an in-development version.

For example:



Name:	Project1
Uuid:	1b8e7abf-eb22-439a-9ef2-a9d5dcb7e9b9
Version:	1.0.27-SNAPSHOT
Publisher:	Customer
Description:	

You can also change the content pack version manually by typing a new version number in the Version field.

**Note:**

- Studio increments only the last minor version automatically. If you want to increment a major version (e.g., 1.0.1 to 2.0.1), you have to do this manually.
- If you enter a version number that is not in the standard major.minor.minor.minor.minor... (1.x.x.x.x) format, the version number will not be automatically incremented when you create another content pack.
- The number of minors in a version number is variable. So, version numbers such as 1, 1.2, 1.2.3.4.5.06 and 1.1.1.1.1.1.1.2 are also automatically incremented.

## Content Pack Versioning Without Revision Control

When working in Studio without version control, Studio provides an automatic method of incrementing the project version when creating a new content pack. In this way, you can easily create multiple content packs with consecutive version numbers, e.g., 1.0.1.1, 1.0.1.2, 1.0.1.3.

When opening the Create Content Pack wizard, Studio reads the version in the content pack properties, removes the –SNAPSHOT suffix, and presents the version number in the Content Pack Properties step.

After clicking **Create CP** in the Dependencies Management step, the content pack is created, and the last digit of the version number is automatically incremented by 1.

The –SNAPSHOT suffix is added to the Project version which means that when you work on the project, you are adapting an in-development version.

**Note:**

- Studio increments only the last minor version automatically. If you want to increment a major version (e.g., 1.0.1 to 2.0.1), you have to do this manually.

- If you enter a version number that is not in the standard major.minor.minor.minor.minor... (1.x.x.x.x) format, the version number will not be automatically incremented when you create another content pack.
- The number of minors in a version number is variable. So, version numbers such as 1, 1.2, 1.2.3.4.5.06 and 1.1.1.1.1.1.1.2 are also automatically incremented.

## Content Pack Versioning With Revision Control

When working with revision control, every time you create a new content pack, Studio adds an SCM (SVN or Git) tag that represents a snapshot of the current state of the repository (for Git) or project (for SVN). This tag incorporates the content pack name and version number.

Using this tag, you can easily reset to/update to a previous project version and reset your current environment to the same sources that were used at the time a content pack was created.

In this way, you can easily create a patch for a specific content pack.

**Note:** Before creating a content pack release, you must ensure that you have no local uncommitted changes on the project and that the project revision is up-to-date. The content pack creation will fail if these two conditions are not met.

However, if the conditions are not met, you can create a temporary "SNAPSHOT" content pack. For this content pack, the version number is not incremented, and you will not be able to run a **Revert to an older project version** operation.

## Recommended Best Practices

- 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 do not contain configuration items with identical names. It is recommended however that you prevent duplicate configuration item names.
- If you have moved content from one content pack to another, the content in the more recently deployed content pack will overwrite the content that was deployed first. For example, you deploy Contact Pack A, containing flow1 and flow2. Central now contains flow1 and flow2, which are assigned to Contact Pack A. Then, you deploy Contact Pack B, which contains flow1. Central now contains flow2 assigned to Contact Pack A and flow1 assigned to Contact Pack B.

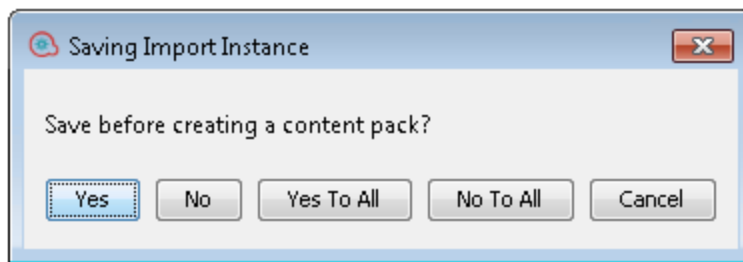
## What do you want to do?

### Create a content pack

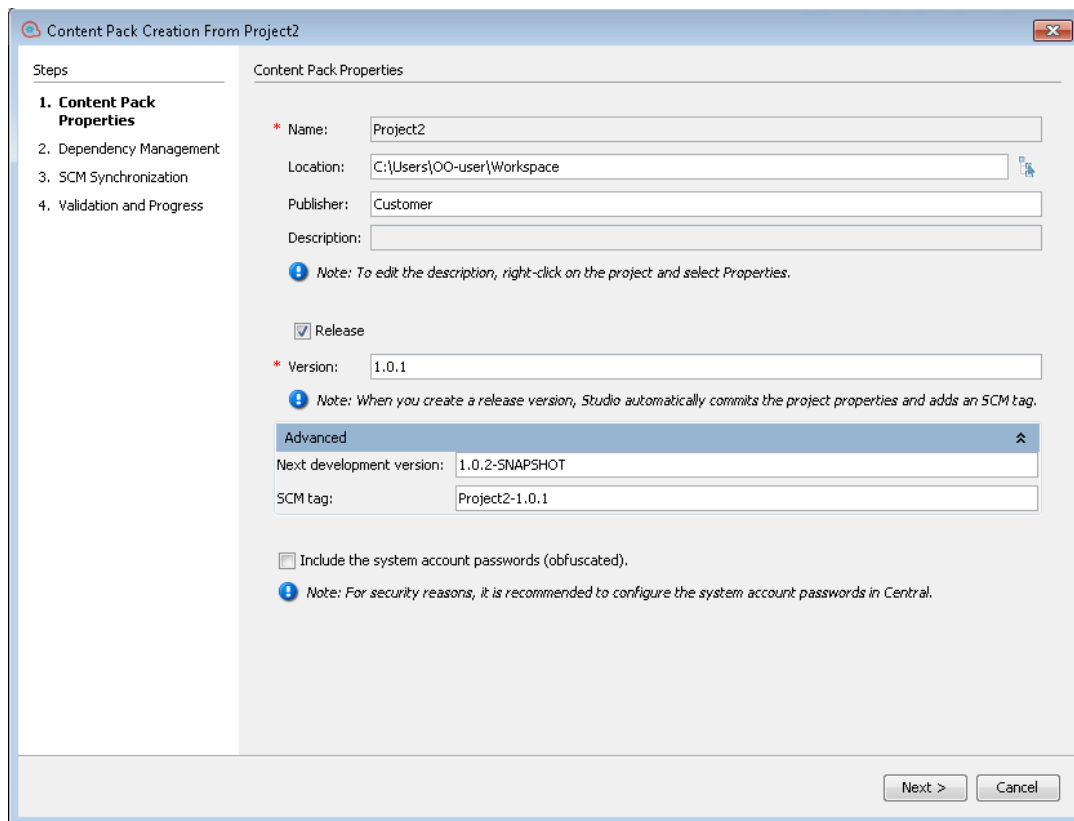
1. In the **Projects** pane, select the project from which you want to create a content pack.
2. Select **File > Create New Content Pack**.

**Note:** Alternatively, you can select the **Create Content Pack**  button in the **Projects** pane or right-click on the project and select **Create Content Pack**.

If there are unsaved editors open, the **Saving** dialog box gives you the option to save the changes. Click **Yes to All** to save all changes in the open editors, or click **Yes** or **No** for each one individually.



3. In the Content Pack Properties step of the **Create Content Pack** wizard, enter the content pack details:



**Note:**

- The **Name** field is taken directly from the project name and you cannot change it.
- The new project version is the Development version set in the Advanced tab of the Content Pack Properties step of the Content Pack Creation wizard.

4. In the **Location** field, enter or browse to the location where you want to save the content pack. By default, the path to the project workspace is selected.

**Note:** By default, the path shown here is the last location in which you created a content pack.

5. In the **Publisher** field, enter the publisher of the content pack. This information will appear in the content pack's **Properties** page.
6. To edit the **Description** field, right-click on the project in the Projects tree and select **Properties**. Then, type a description in Description area.
7. Select the **Release** check box to create a final version of the content pack for publishing. After

checking this box, the -SNAPSHOT suffix is removed from the version number in Version field.

**Note:**

- If the Release check box is disabled, this means you are not connected to any source control management system.
- If you select the **Release** check box, the SCM Synchronization window is added as Step 3 in the Content Pack wizard. See "[Content Pack Creation Wizard - Step 3](#)" on [page 359](#) for details.

8. In the **Version** field, the next minor version number is shown. The last digit of the version number is automatically incremented when you create a new content pack. However, you can change this number manually by typing a different version number. For details, see "[Content Pack Versioning Without Revision Control](#)" on [page 348](#).

**Note:** When creating a release version of the content pack, Studio commits the project properties and adds an SCM tag to the content pack.

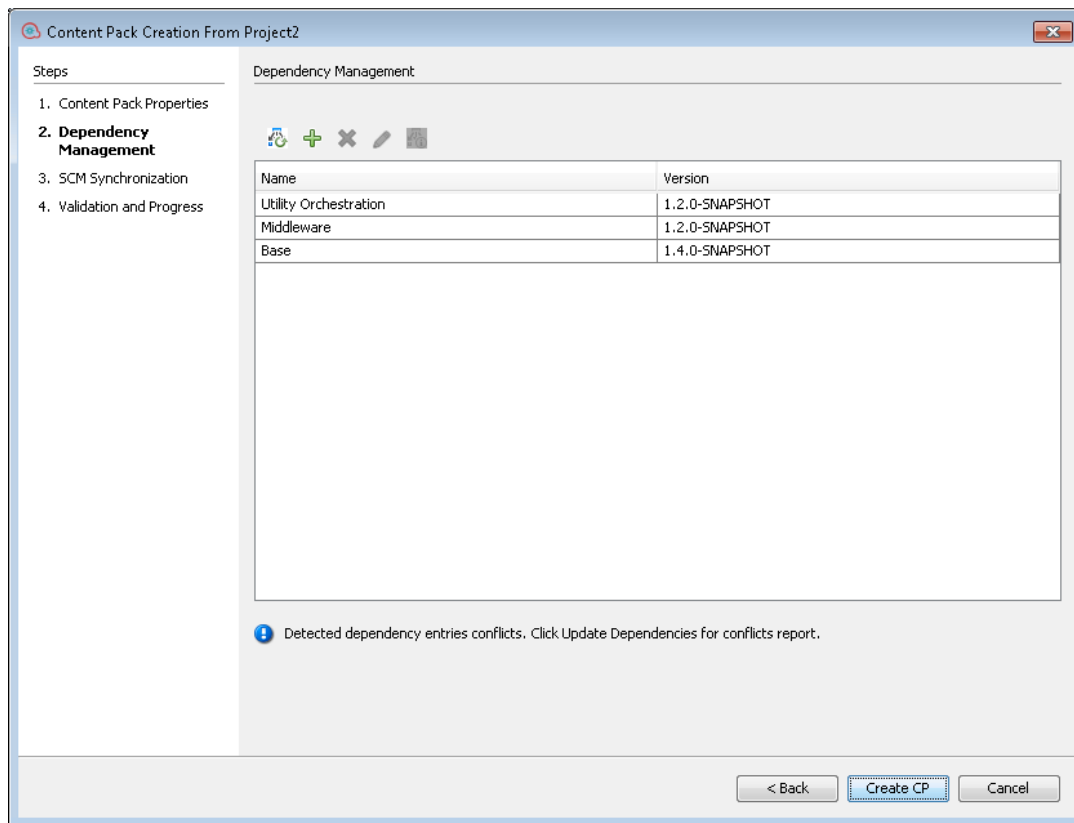
9. **For source control only:** Select the **Release** check box if you want to increment the revision number when you commit the changes to SVN/Git.
10. Select **Advanced** to show the following fields:
  - a. **Development version:** The next version with the extension -SNAPSHOT automatically appears in this field. You can change it as required.
  - b. **For source control only:SCM Tag:** By default, the SCM tag name that will be used when releasing the content pack is [<projectname>-<version>]. However, you can change this as required, and type a different name that helps you to identify the version in the source control.
11. Select the **Include the system accounts passwords?** check box if you want the content pack to include passwords for the system accounts. When the content is deployed on Central, the user names and passwords will be deployed.

**Note:** The passwords will be obfuscated inside the content pack. However, this is NOT a safe option and is not recommended. Alternatively, it is recommended to use the option of configuring the system accounts in Central. For details, see "Setting Up Configuration Items for a Content Pack" in the *HP OO Central User Guide*.

12. Click **Next**.

If the dependencies are not up-to-date, the Dependency Management window opens:





13. From this window, you can manage dependencies - add, delete, edit and update dependencies as in the Dependencies Editor. See ["Managing Content Packs and Dependencies in a Project" on page 131](#) for full details.

14. Click **Next**.

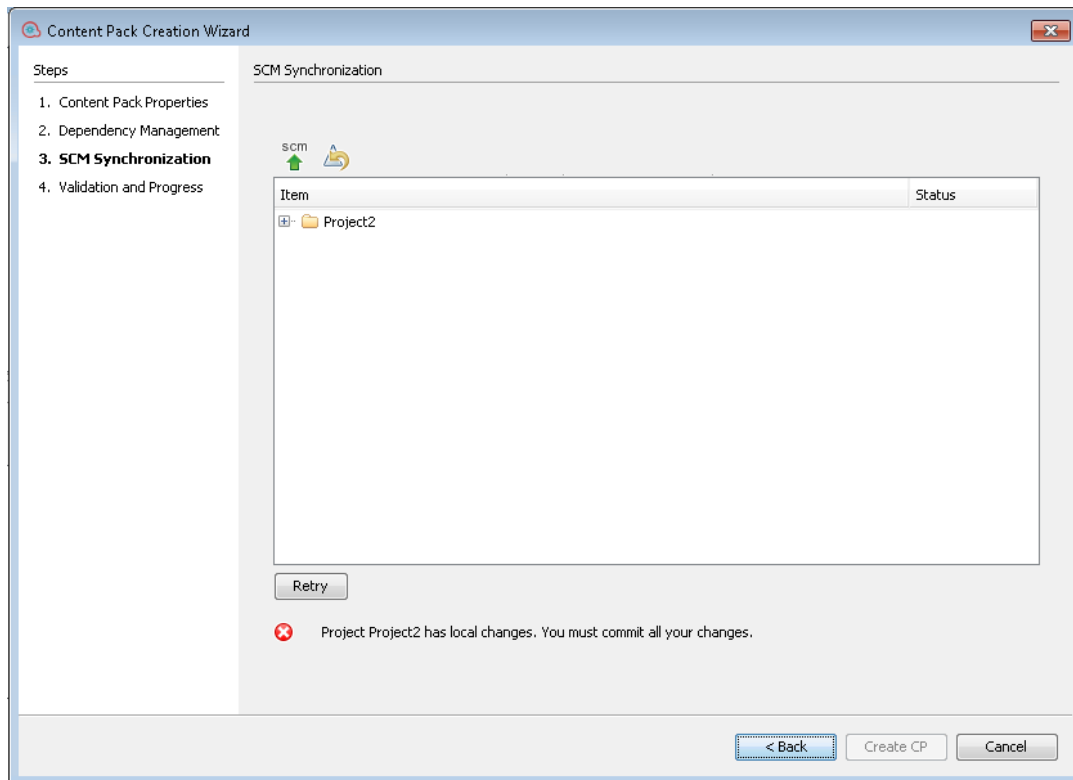
If you selected the **Release** check box in the first step, the SCM Synchronization window opens.

**Note:** The SCM Synchronization page displays only in the following cases:

- If the project is not synchronized
- Changes were made to the project dependencies in the **Dependency Management** window
- If there are uncommitted changes in the workspace.

Otherwise, you will see the Validation and Progress window after clicking **Next**.

If there are any SCM synchronization issues between the local version and the main repository, Studio displays them in the SCM Synchronization window. You must solve these issues before continuing. You can use the SCM toolbar to commit, push or revert changes as described in "[Working with Git Source Management System](#)" on page 88.



**Note:** If you modified the dependencies in the Dependencies management step, they will also appear as changes in the SCM Synchronization step.

After solving the SCM issues, click **Retry**. If there are no more issues, you will see the following message:

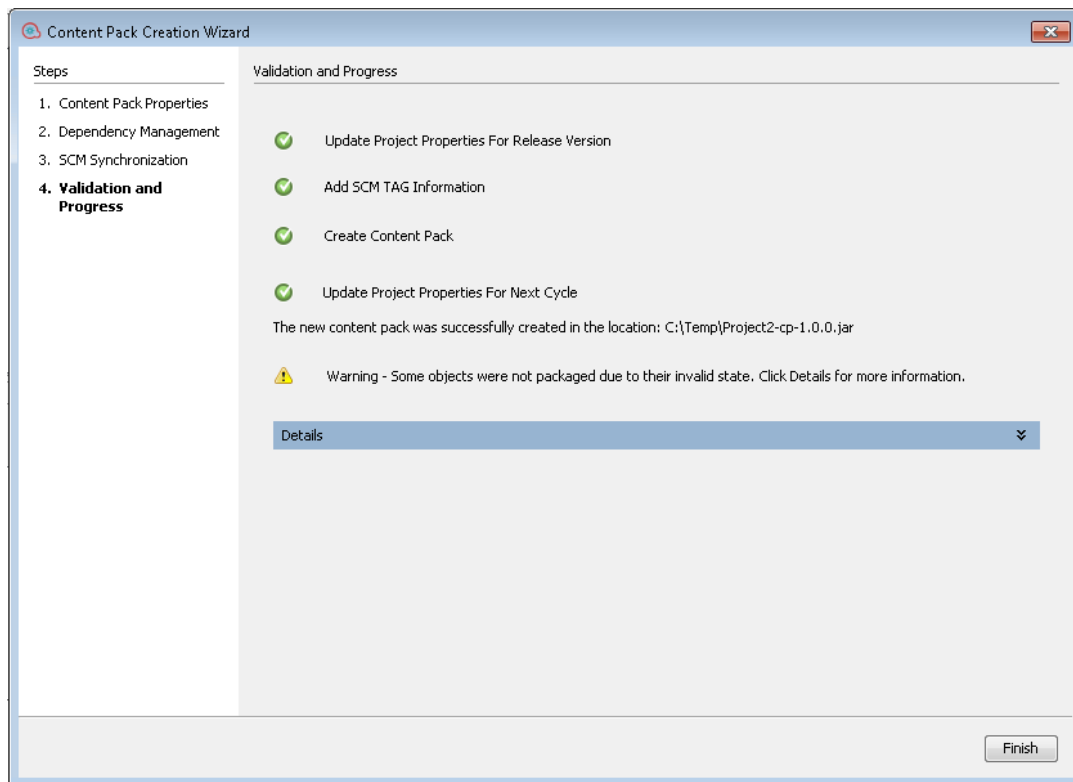
You can now continue creating the content pack.

15. Click **Create CP**.

All the commit operations that were performed for the current branch are automatically pushed to the main Git repository and the content pack is created.

If there are any invalid flows or dependency warnings in the project, Studio presents them. Click **Details** to see the warnings.

Invalid flows are not packaged into the content pack.



The content pack is created in the specified location.

16. After the new content pack is created, a message appears, with a link to the location in which the content pack was created. Click on the link to access the content pack.
17. Click **Finish** to close the wizard window.

The new content pack can now be deployed and run, or imported into another project.

**Note:** After the content pack is created, the **contentpack.properties** file contains the version number as follows:

- **Content Pack: contentpack.properties** contains the published version. For example, 1.8.3.
  - **Project: contentpack.properties** contains the next snapshot version. For example, 1.8.4-SNAPSHOT.
- 
- The version number is saved in the pom.xml file. The pom.xml file also includes project dependencies information.

## View/edit the project version

1. In the **Projects** pane, select the project whose version you want to view/edit.
2. Select the **Properties** option in the Projects pane.

or:

Right-click the project and select **Properties**.

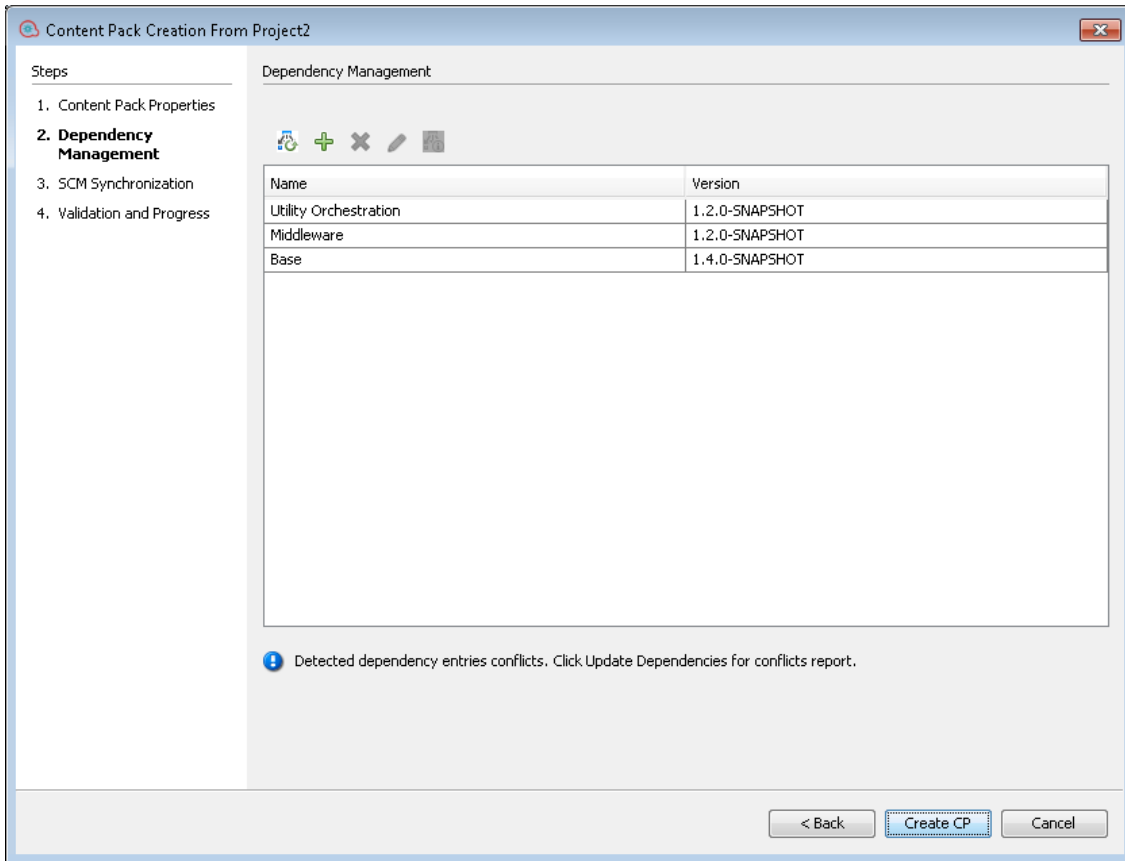
## Reference Material





### Content Pack Creation Wizard - Step 1

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.

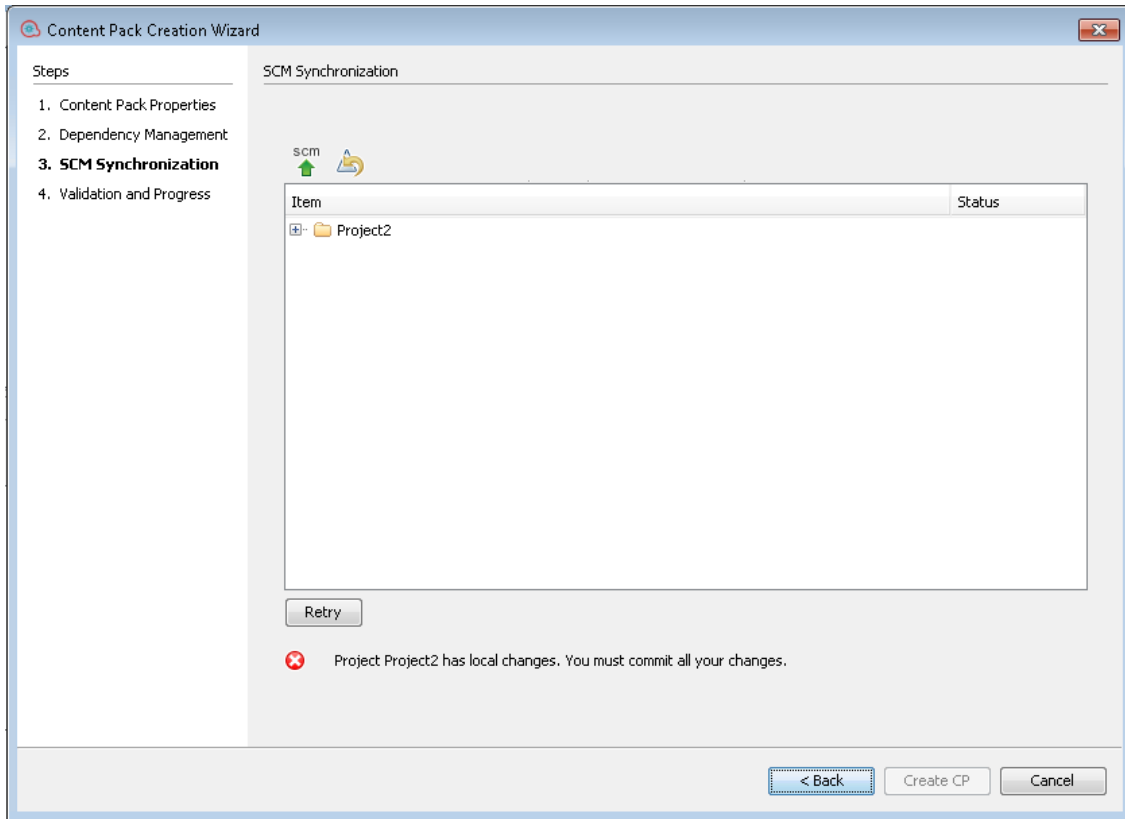
<b>Publisher</b>	Enter the publisher of the content pack. This information will appear in the content pack's <b>Properties</b> page.
<b>Description</b>	To edit the <b>Description</b> field, right-click on the project in the Projects tree and select <b>Properties</b> . Then, type a description in Description area.
<b>Release</b>	Select this check box if you want to increment the revision number when you commit the changes to SVN/Git.  This also creates a tag with the current state of the project (SVN) or repository (GIT).  If you select this check box, Studio adds the SCM Synchronization step to the Create Content Pack wizard.
<b>Version</b>	Enter the version of the content pack. This information will appear in the content pack's <b>Properties</b> page.
<b>Advanced</b>	Select the arrows to show the SCM fields.
<b>Development version</b>	The next version with the extension -SNAPSHOT automatically appears in this field. You can change it as required.
<b>SCM Tag</b>	Type the text you want to use as the SCM tag name. You can use this field to identify the version in the source control.
<b>Include the system accounts passwords?</b>	Select the check box if you want the content pack to include passwords for the system accounts. These passwords will be obfuscated. If this check box is not selected, the passwords will not be included in the content pack.  <b>Note:</b> For security reasons, it is recommended to configure the system account passwords in Central.



## Content Pack Creation Wizard - Step 2



GUI item	Description
	Add a new dependency.
	Delete a dependency from the list.
	Edit a dependency's properties.
	Update dependencies.

## Content Pack Creation Wizard - Step 3



GUI item	Description
	<b>Commit all changes:</b> Commits all the changes in the <b>SCM Changes</b> pane. Only available when there are changes.
	<b>Revert all changes:</b> Reverts all changes that show in the <b>SCM Changes</b> pane.

## 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

There are three ways to create operations in Studio:

- Copying and modifying existing operations.
- Importing an operation from an already existing plugin.
- Creating an action plugin in Java and importing that action plugin to Studio.

### Creating Operations from Action plugins

An action plugin is a jar file containing IActions or @Actions. You can import an action plugin to Studio, in order to create an operation from one of the actions in it.

An action plugin may include multiple actions, and you can create an operation from each one of these actions.

For information about developing action plugins, see the *Action Developers Guide*.

### *Copying Operations that were Created from Action plugins*

When you copy an operation that was created by importing an action plugin, the copied operation continues to reference the original operation. If the action plugin 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 plugin 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

**Important!** If you need to delete, create, or rename an operation inside your project, make sure to do this from within Studio, and not by deleting, creating, or renaming the item in the file system.

1. In the **Dependencies** 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 211](#).
- For information on defining an input data source, see ["Specifying the Input Source " on page 222](#).

7. Add and define any output data.

For information on adding and working with output data, see ["Setting Operation Outputs" on page 252](#).

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 242](#).

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**

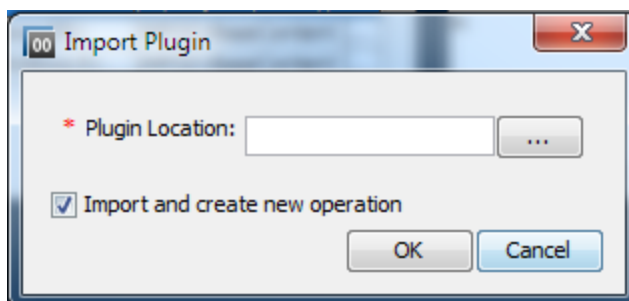
If you created an operation by copying an existing operation, you can change the source operation. Your copied operation will become a copy of the operation that you select as the source.

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 plugin**

The most direct way to create an operation from an action plugin is to import it and create the operation at the same time.

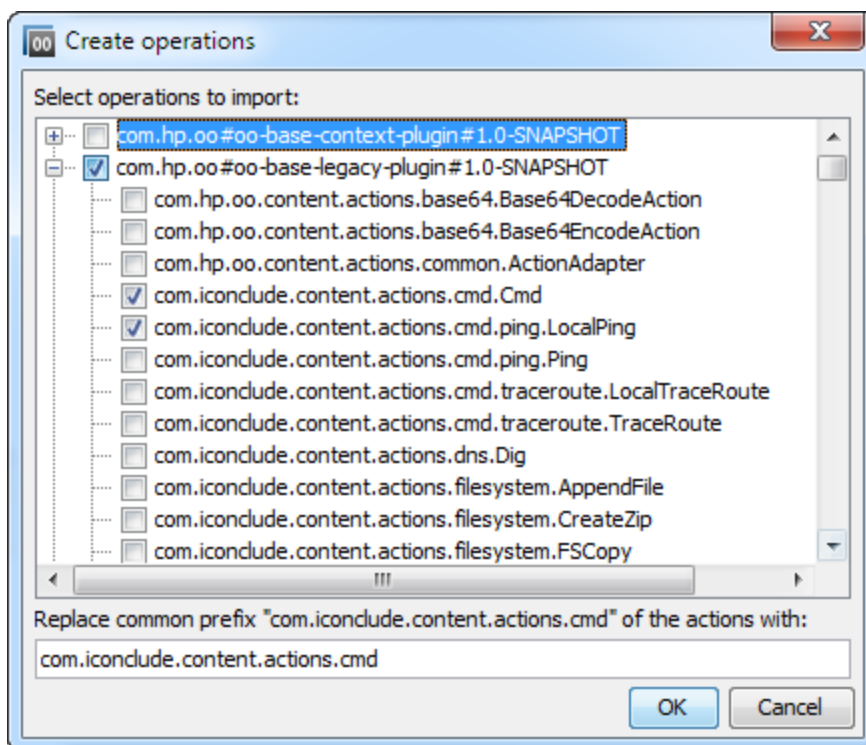
1. Create and pack an action plugin, so that it includes the actions that were developed. For information about how to develop an action plugin, see the *Action Developers Guide*.
2. In Studio, right-click the folder where you want to create the new operation, and select **Import plugin**.
3. In the Import Plugin dialog box, click the browse button to browse to and select the HP OO plugin that you want to import.



**Note:** It is possible to import a single plugin (maven artifact), either by the JAR file or by the POM file. The plugin must have both its JAR and POM file at the same location.

If you import a plugin that was already deployed, the new plugin replaces the existing one.

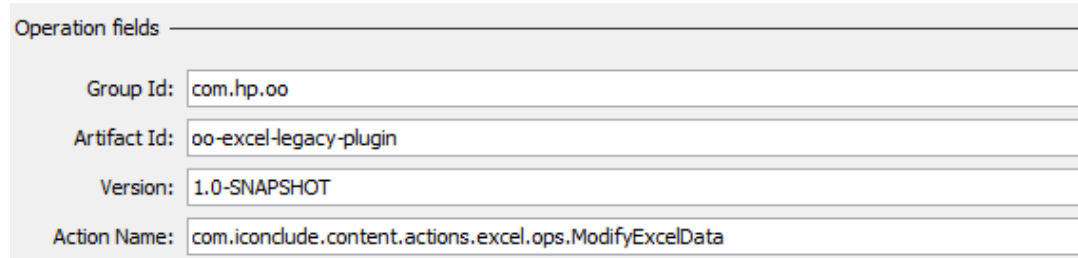
4. Select the **Import and create new operation** check box and click **OK**.
5. In the Create Operations dialog box, expand the plugin containing the actions that you need, and select the actions that you want use to create the operation from.



**Note:** If a plugin 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 plugin is displayed in the **Operation Fields** section at the top of the **Inputs** tab, in the operation's Properties window.



Operation fields

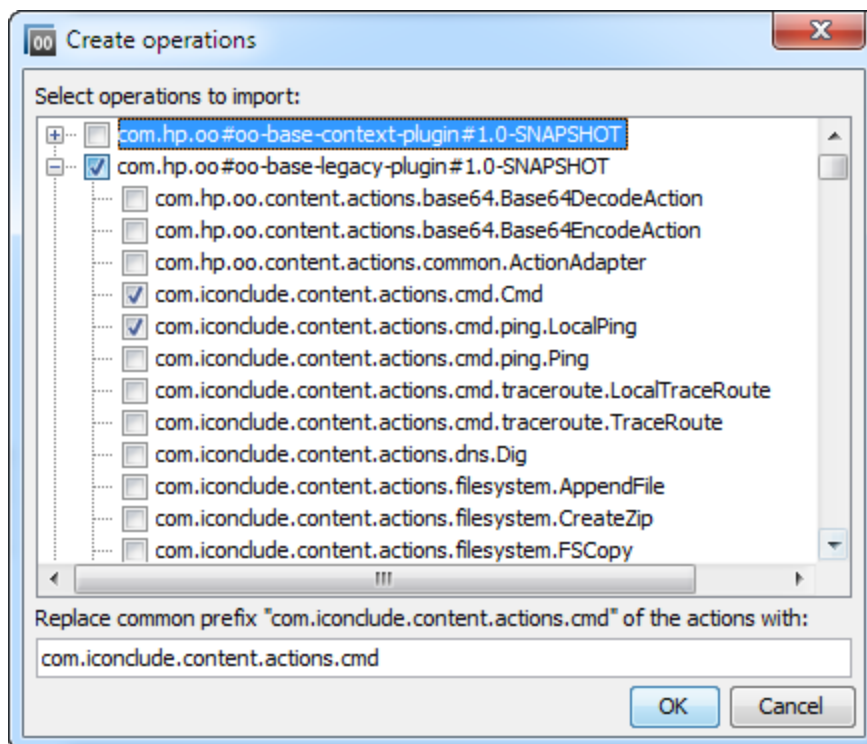
Group Id:	com.hp.oo
Artifact Id:	oo-excel-legacy-plugin
Version:	1.0-SNAPSHOT
Action Name:	com.iconclude.content.actions.excel.ops.ModifyExcelData

6. Save the operation.

### Create an operation from an imported action plugin

After action plugins 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 plugin, 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 plugin 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 plugin is displayed at the top of the **Inputs** tab, in the operation's Properties window.

## Import action plugins

It is possible to simply import action plugins 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 plugin**.
2. In the Import plugin dialog box, browse to and select the HP OO plugin that you want to import to Studio's local Maven repository.
3. Click **OK**. The plugin is available for authors to create operations from. For more information, see *Create an operation from an imported action plugin*.

## 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 **Dependencies** 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.

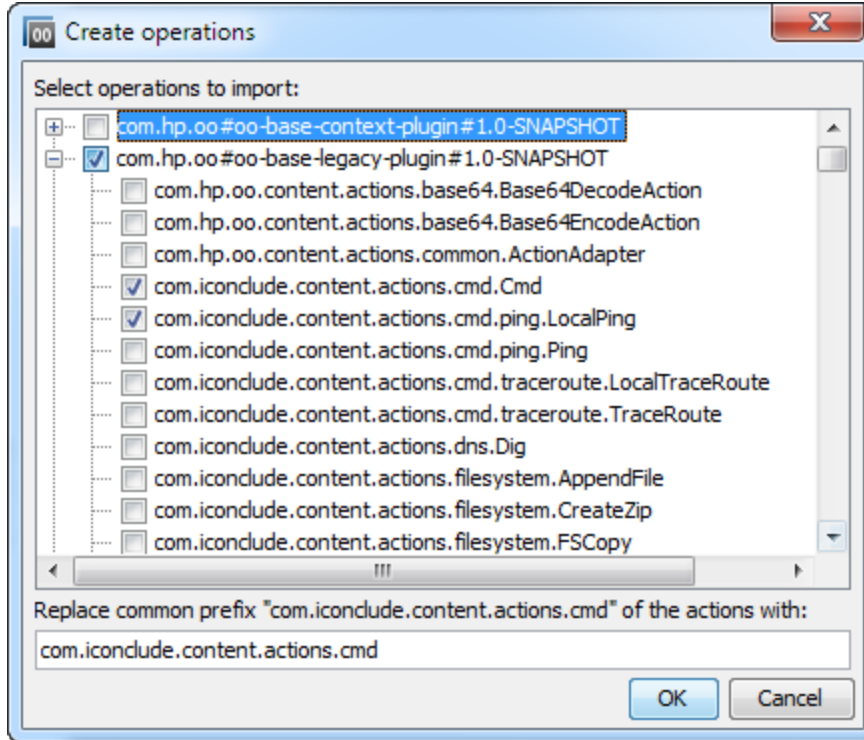
To create a display operation, you copy the display operation template from the base content in `Library/Utility Operations/Display Message` and define the information that will be displayed to the user.

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 **Dependencies** 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 (up to 128 characters).
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.

## Reference Material

### Create Operations dialog box



GUI item	Description
<b>Select operations to import</b>	Expand the plugin containing the actions that you need, and select the actions that you want use to create the operation from.

### 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 (up to 128 characters).
<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 Plugin

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



<b>Plugin location</b>	Browse to and select the HP OO plugin that you want to import.
<b>Import and create new operation</b>	Select this check box to create a new operation from the imported plugin.

### Operation Properties: Advanced tab

GUI item	Description
<b>Link To</b>	Displays the source operation, from which the selected operation was copied.
<b>Location</b>	Displays the location of the source operation.
<b>Select</b>	Lets you select a different source operation
<b>Open</b>	Opens the Properties sheet of the source operation.
<b>Detach</b>	Detaches the operation from the parent plugin.

## 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 **Dependencies** pane
- View the descriptions of the folders, flows, and operations
- Run a search using the **Search** tab
- Run a search using the **Navigate to flow...** option

## 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 **Dependencies** 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 380.

## Search for an operation or configuration item using the Search tab

Using the **Search** pane, you can perform a full-text search throughout the Library. You can search for flows, operations, or configuration items 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. The search criteria includes: by name, description, Lucene query and all fields

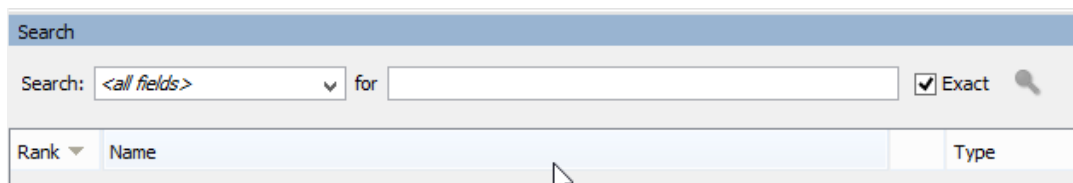
**Note:** To include all the fields in your search, leave the **Search** list set to **<all fields>**.

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** check box.
  - To specify a search that treats spaces as separators for alternate search strings, clear the selection from the **Exact** check box.
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:

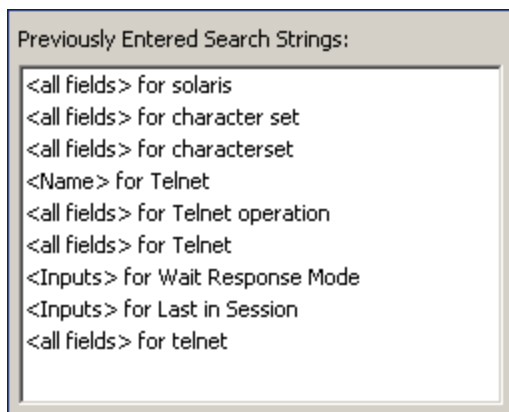
- The kind of information the operation's inputs need.
- The information that the results include.
- The operation's requirements and assumptions.

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.

### Search for a flow using the Navigate to flow... option

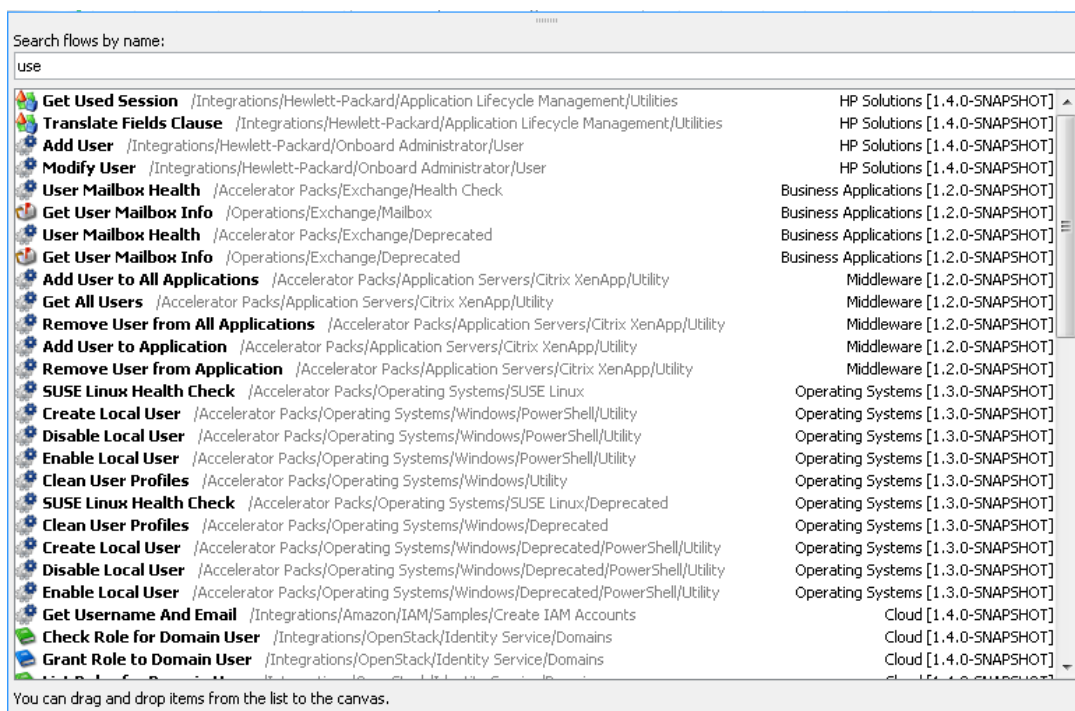
You can also use the **Navigate to flow..** option to easily find and open a flow in your workspace.

1. Select **Tools > Navigate to flow...** .

The **Search flows by name:** window opens.

2. Type in part of the flow name.

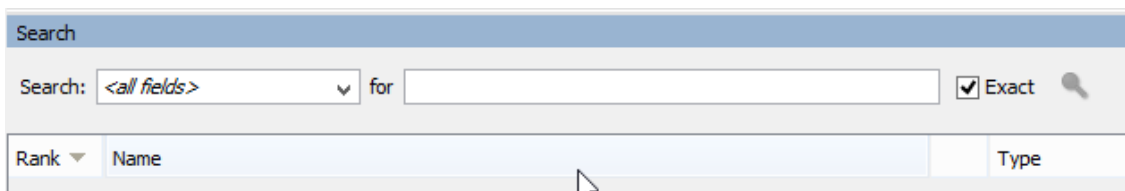
The window is immediately populated with search results. For example, if you search for a flow whose name contains the string "use", you may see the following results:



- Use the Up and Down arrow keys to move in the list, then press Enter to open the selected item. Alternatively, you can open an item with a double-click.
- You can create steps in the current Flow Editor by dragging and dropping flows from the list.

## Reference Material

### Search pane



GUI item	Description
Search	<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 content 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 plugin. 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 parentless and loses its link to the plugin. In this case, a new parent will have to be selected manually.

**Note:** In order to fix an orphan soft copy operation that points to a nonexistent parent, use the **Select** button from the **Advanced** tab of the operation to attach a parent operation.

- 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 360.

### **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.x, 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.

**Note:** The **Select** button in the **Advanced** tab in the operation editor is not available for hard copies.

## Changing From a Soft Copy to a Hard 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.

## 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 **Dependencies** 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 > Copy**.

### 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 **Dependencies** 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.

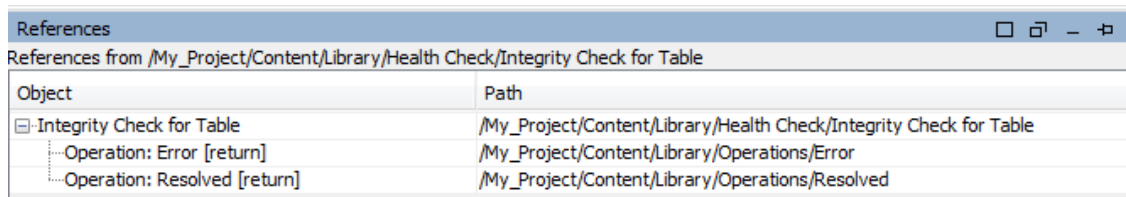
1. In the **Dependencies** pane or **Projects** pane, right-click the flow that you want to copy.
2. Select **Edit > Copy Deep**.
3. Navigate to the folder in which you want to place the flow and its operations, right-click and select **Edit > Paste**.

## Cut a flow or operation

1. In the **Dependencies** 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 > Cut**.

# 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.



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.

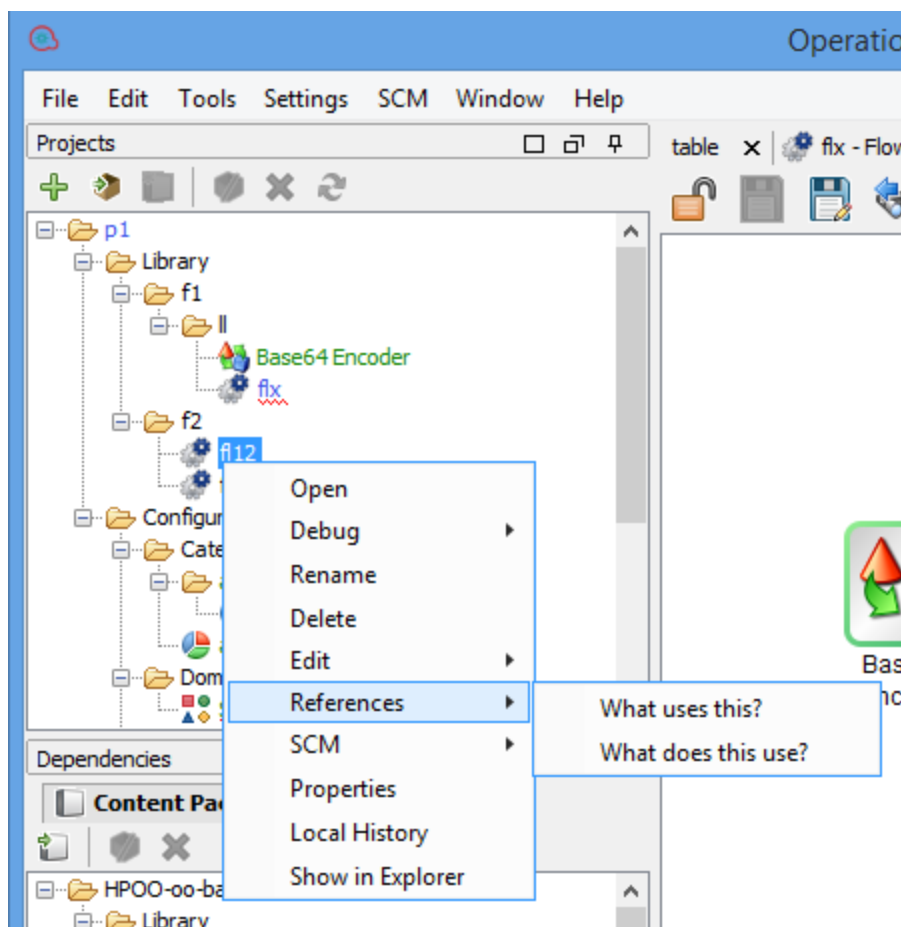
## Best Practices

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, and select **References > What uses this?**.



The **References** pane opens, displaying the references to the operation or flow.

2. Double-click a row in the **References** pane to open the item for editing.

**Note:** To open multiple items, select them using the SHIFT or CONTROL keys, right-click, and select **Open**.

### 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?**.

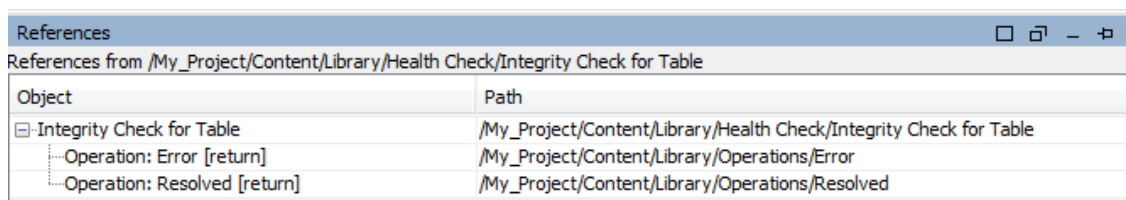
The **References** pane opens, displaying the references from the operation or flow.

**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.



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.

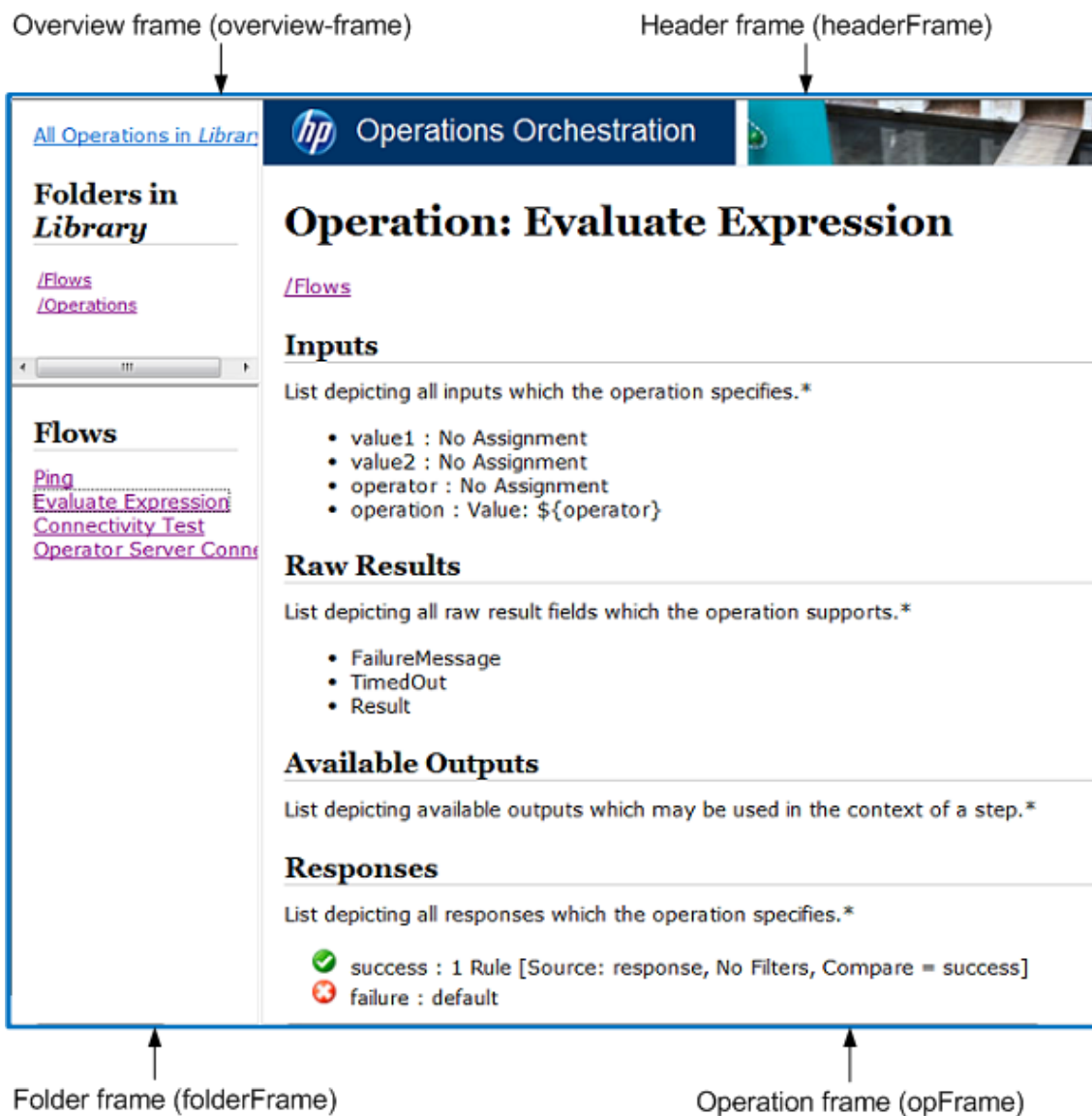
You can document flows and operations, to provide more information about them for other users:

- 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.

**Note:** You can only generate documentation in the **Content Packs** tab. This option is not available in the **Library** view.

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.

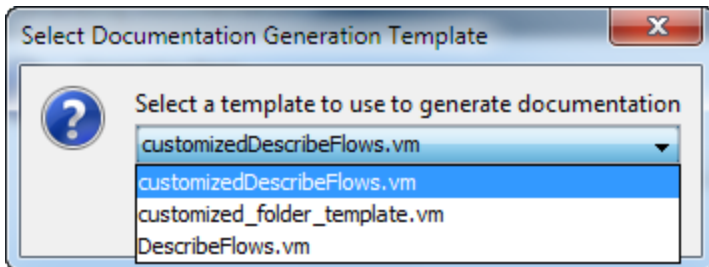
### Generate documentation in a custom format

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**

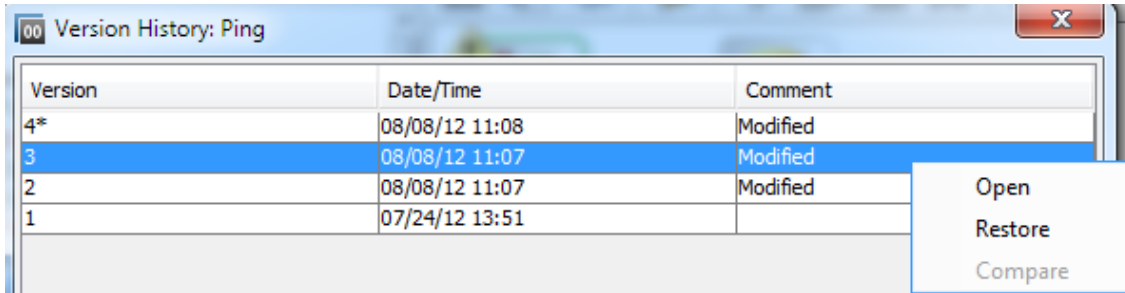
Each time that a configuration item, flow, or operation is saved, a new version of the item is created. The Version History dialog box lists these versions, and lets you:

- 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.


For full details of how to handle version history with a source control repository, see "[View a history of Git operations in the repository](#)" on page 110.



## 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.

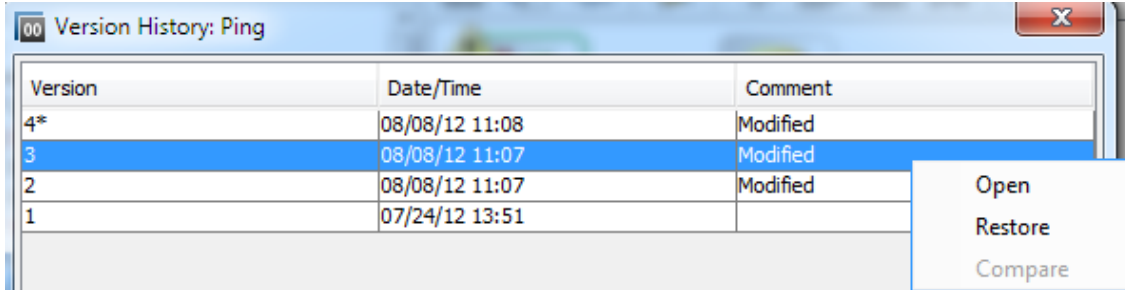
### Compare versions

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



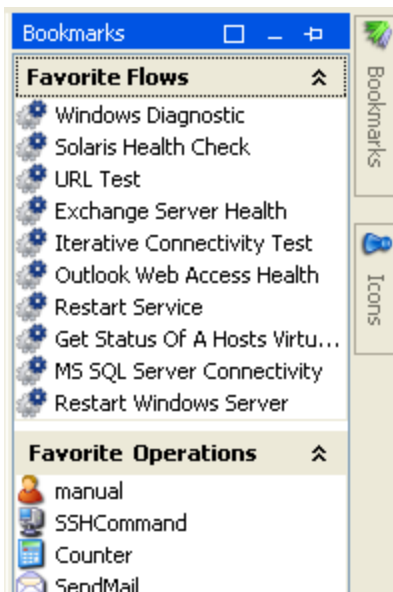
Menu item	Description
<b>Version</b>	The number of the project version, generated automatically.
<b>Date/Time</b>	The date and time that the version was changed.
<b>Comment</b>	The type of change that was made.

## 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**.

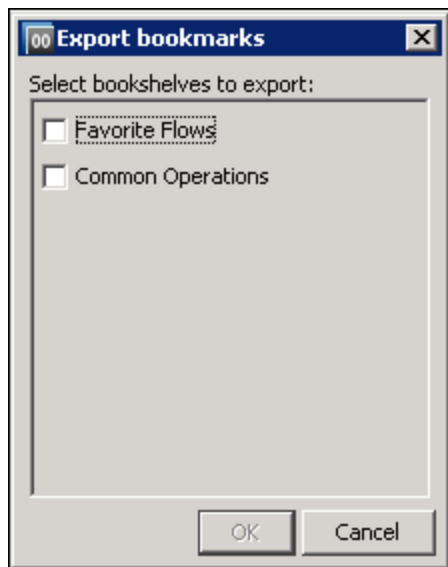
### Collapse/expand a shelf

1. To collapse a shelf, click the double chevrons  $\blacktriangleleft\blacktriangleright$  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.

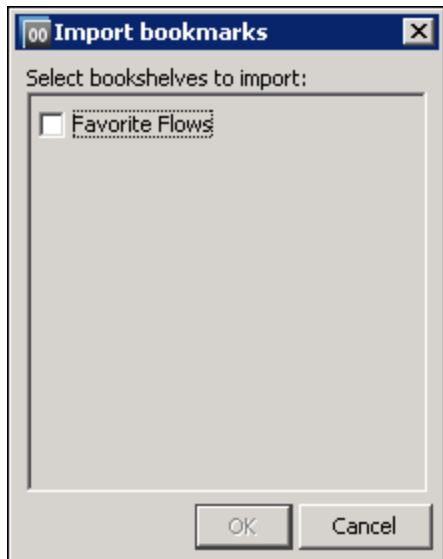
1. Right-click anywhere inside the **Bookmarks** pane, and then select **Export**.
2. In the Export bookmarks dialog box, select the bookshelves that you want to export.



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.

## Copy a flow or operation from the Bookmarks pane to a project


You can also drag flows and operations from the Bookmarks pane to the **Projects** pane, to copy them to a project.

1. Select a flow or operation in the **Bookmarks** pane.
2. Drag the item to a project in the **Projects** pane.

## 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>	



## Configuring Studio Properties

You can configure some of the properties in Studio by changing the default settings in the **Options** dialog. In addition, you can configure all the different properties in Studio by changing the default settings manually in the **Studio.properties** file. For details of all the settings, see "[Studio Properties List](#)" on page 400.

In the following sections, you can see descriptions of the more frequently used parameters.

### What do you want to do?

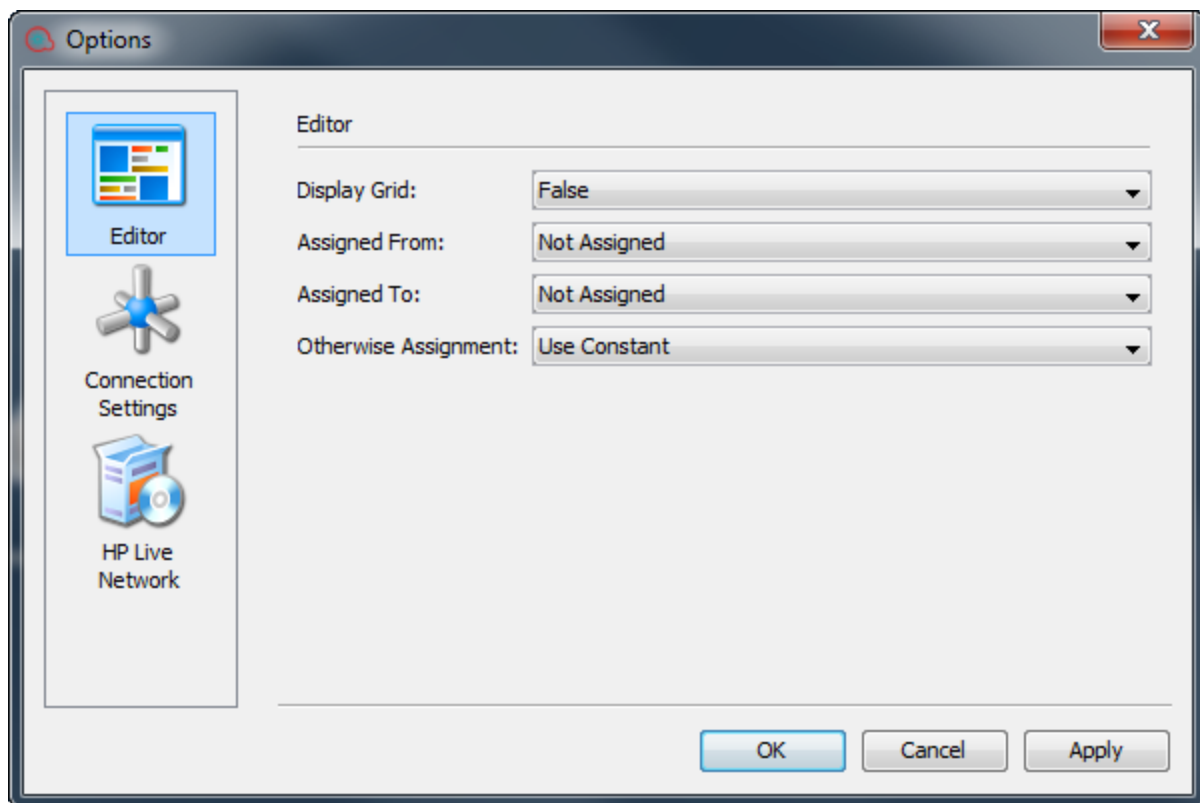
#### Edit the studio properties

1. In Windows Explorer, go to the <HP OO home> > **Studio** > **conf** directory.
2. Open the file **studio.properties** file in a text editor.
3. If you know the name of the parameter you want to change, search for it in the text editor.
4. If you do not know the name of the parameter, review the table in "[Configuring Studio Properties](#)" above to find it.

#### Specify the default values for Assign from, Assign to and Otherwise using the Options dialog

1. In Studio, select **Configuration** > **Options**.

The Options dialog opens:



2. For **Assign From** and **Assign To**, select one of the following:

<b>Not Assigned</b>	No value is assigned to or from the input.
<b>Assigned</b>	The value of a flow variable with the same name as the input is assigned to or from the input.

3. For **Otherwise**, select one of the following:

<b>Use Constant</b>	The <b>Otherwise</b> field is set to <b>Use Constant</b> with an empty value.
<b>Prompt User</b>	The <b>Otherwise</b> field is set to <b>Prompt User</b> .

### Define the default value of Assign from and Assign to manually

1. In Windows Explorer, go to the <HP OO home> > **Studio** > **conf** directory.
2. Open the file **studio.properties** for editing.
3. Find the following lines:

**dharmastudio.ui.inputinspector.assignfrom.selected=not-assigned**

### **dharma.studio.ui.inputinspector.assignto.selected=not-assigned**

For each parameter, set one of the following values:

<b>not-assigned</b>	No value is assigned to or from the input.
<b>assigned</b>	The value of a flow variable with the same name as the input is assigned to or from the input.

**Note:** This behavior is true in the case of a clean installation of Studio 10.50. If you upgrade to version 10.50, the default values of **Assigned from Variable** and **Assigned to Variable** are automatically set to the values assigned in the previous version.

### **Change the default value of Otherwise to Use Constant**

1. In Windows Explorer, go to the <HP OO home> > **Studio** > **conf** directory.
2. Open the file **studio.properties** for editing.
3. Find the following line:


**default.behavior.define.input=use\_constant**

4. For each parameter, set one of the following values:

<b>use_constant</b>	The <b>Otherwise</b> field is set to <b>Use Constant</b> with an empty value.
<b>prompt_user</b>	The <b>Otherwise</b> field is set to <b>Prompt User</b> .

**Note:** This behavior is true in the case of a clean installation of Studio 10.50. If you upgrade to version 10.50, the default value of is **default.behavior.define.input** is automatically set to the value assigned in the previous version.

### **Make the grid appear by default when opening the flow editor**

This is a global setting to control the activation of the grid for all flow editors, it makes the flow editor start with the grid activated each time. You can still change the grid using the Show/Hide Grid button  for an individual flow editor instance.

By default, the grid is not shown when opening the flow editor.

You can make the grid appear from the Options dialog or by editing the **studio.properties** file.

From the **Options** dialog:

1. In Studio, select **Configuration > Options**.
2. In the Display Grid field, select **True** to show the grid or **False** to hide the grid.

From the **studio.properties** file:

1. In Windows Explorer, go to the **<HP OO home> > Studio > conf** directory.
2. Open the file **studio.properties** for editing.
3. Find the following line:

```
dharma.studio.ui.activegrid=false
```

4. Change the value of the setting to true:

```
dharma.studio.ui.activegrid=true
```

### **Specify the number of characters for each tooltip**

1. In Windows Explorer, go to the **<HP OO home> > Studio > conf** directory.
2. Open the file **studio.properties** for editing.
3. Find the following line:

```
dharma.studio.ui.tooltip.showwhitespacecharacters.max.chars.per.line=5000
```

4. Change the value as required.

### **Show special characters in the context inspector**

1. In Windows Explorer, go to the **<HP OO home> > Studio > conf** directory.
2. Open the file **studio.properties** for editing.
3. Find the following lines:

```
dharma.studio.ui.tooltip.showwhitespacecharacters.max.chars.per.line=5000
```

```
dharma.studio.ui.tooltip.showwhitespacecharacters.space=&#8201;
```

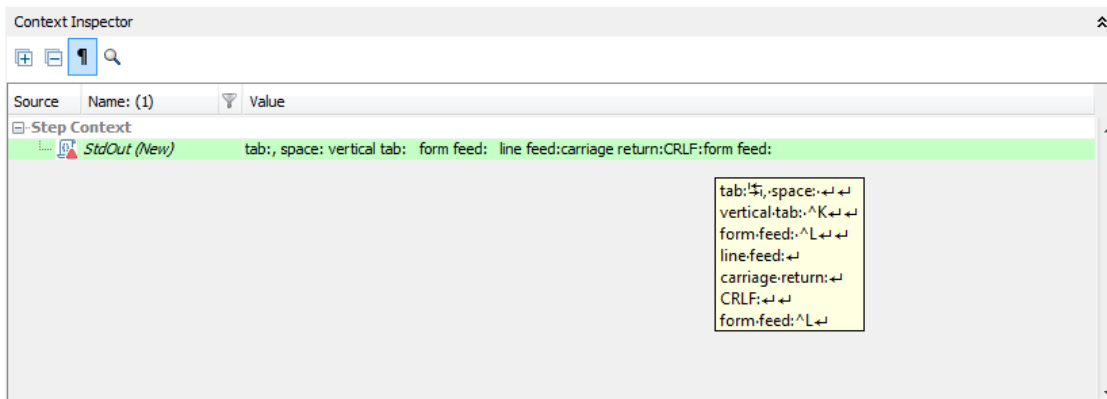
```
dharma.studio.ui.tooltip.showwhitespacecharacters.tab=&#8203;
```

```
dharma.studio.ui.tooltip.showwhitespacecharacters.enter=&#8203;
```

```
dharma.studio.ui.tooltip.showwhitespacecharacters.otherwhitespace=&#8226;
```

4. Change the values as required.

For example, if you define character overrides, you can see them in the Context Inspector as follows:



## Disable HPLN search

By default, the HP Live Network panel appears in the lower area of Studio. You can disable this panel by changing the value of the **hpln.enabled** parameter.

1. In Windows Explorer, go to the <HP OO home> > **Studio** > **conf** directory.
2. Open the file **studio.properties** for editing.
3. Find the following line:

```
hpln.enabled=true
```

4. To disable the search, change the value to false:

```
hpln.enabled=false
```

## Specify the HPLN connection/read timeout

1. In Windows Explorer, go to the <HP OO home> > **Studio** > **conf** directory.
2. Open the file **studio.properties** for editing.
3. Find the following lines:

```
hpln.connection.timeout=180
```

```
hpln.read.timeout=300
```

4. Set the timeout value as required. For no timeout, set these parameters to 0.

## Specify the maximum number of HPLN search results

This configuration setting allows you to specify the maximum number of search results to be displayed inside the HP Live Network search results panel.

1. In Windows Explorer, go to the <HP OO home> > **Studio** > **conf** directory.
2. Open the file **studio.properties** for editing.
3. Find the following lines:

```
hpln.search.results.maximum.count=100
```

4. Set the maximum number of search results as required.

## Defining the default value of Assign from and Assign to

By default, when you create a new input in an operation or a flow step, no value is assigned to the input source. Your flow may require that the input value to be assigned from a flow variable.

Similarly, by default, Studio does not assign the input value to a flow variable. Again, your flow may require that the input's value be assigned to a flow variable.

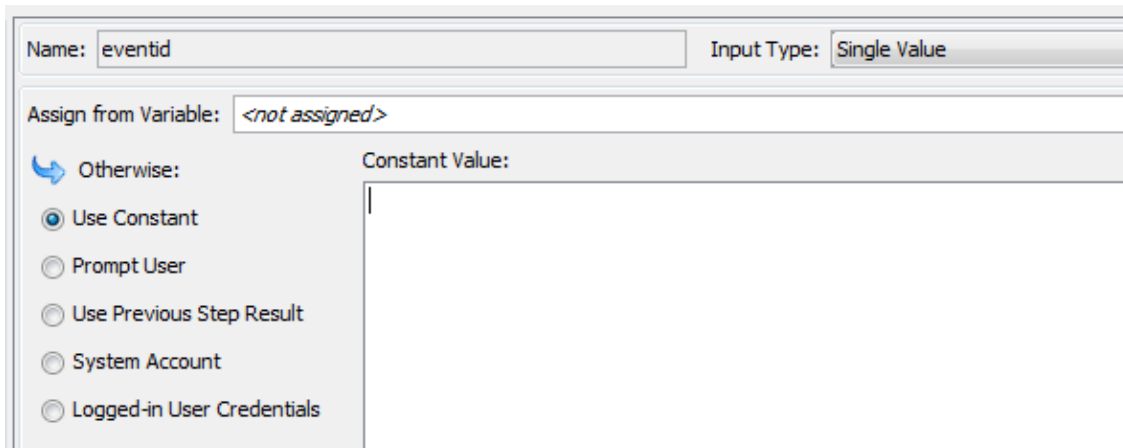
You can configure Studio so that its default behavior is to assign a flow variable of the same name as the input either or both of the source of the input's value. You do so in the **Studio.properties** file.

Changing the setting's value from **not-assigned** to **assigned** causes the **Assign from Variable** and/or **Assign to Variable** to be assigned with the flow variable of the input with the same name, when you:

- Add a new input to an operation or flow step
- Create a new step in a flow
- Create a new operation

## Changing the default value of Otherwise to Use Constant

In Studio10.20, when you create a new input in an operation or a flow step, by default the **Otherwise** parameter is set to **Use Constant** with an empty value for the **Constant** value. With this assignment, some inputs may not pass validation.



Similarly, the same rule applies when you create a new step in a flow using drag and drop from the Project or Dependencies pane. For the step inputs that have the **Otherwise** parameter set to **Prompt User** using text before a drag and drop operation, after the drag and drop, these inputs are changed to **Use Constant** with an empty value. The step inputs that were set to **Prompt User**, but are marked as **Required** or are part of a list (selection list, domain term, flow variable) remain unchanged.

You can configure the default behavior in Studio to use **Prompt User** when a new input is created in an operation or a flow step. In this case, when a new step is created in a flow using a drag and drop operation, the **Prompt User** setting will remain unchanged.

Changing the default behavior from **Use Constant** to **Prompt User** causes the selected **Otherwise** option to be **Prompt User - Prompt for Text** when performing one of the following actions:

- Add a new input to an operation or flow step
- Create a new step in a flow
- Create a new operation

## Showing special characters in the Context inspector

The ¶ button enables you to toggle on and off hidden formatting marks on variables. For example, white space, carriage returns, special characters, ^Ms, and more. By default, this is disabled.

When you enable this switch, the value elements inside the categories, Step Context, Global Context, Select Lists and System Properties, show these special characters and white space inside the tool tips associated with the value.

- If invalid or empty parameters are used in the studio.properties file, the default parameters are used.

- If the preferred value for a property is not properly displayed inside the tool tip, then you can switch to a different value, or leave that property empty.
- New line characters, <CR><LF>, <CR>, <LF> are followed by a new line in the tool tip along with a visual representation.
  - The vertical tab character \u000B is displayed as ^K.
  - The form feed character \u000C is displayed as ^L.

## Studio Properties List

### Studio Properties

Parameter	Default Value	Description
dharmadefault.repository	location=default/repo	The default Studio local repository.
user.language	en	The locale of Studio. These value do not affect the SCM repository. If they are not set or are invalid, Studio uses the system locales.
user.region	US	
dharmarepo.max_history_length	50	The number of item to be saved in the history for the offline repository.
dharmastudio.ui.lookandfeel	windows	The look and feel of the Studio application.
dharmastudio.ui.input.constant.max.chars	8192	A positive integer representing the maximum number of characters of the constant value. Must be less than 8193.
dharmastudio.ui.tooltip.showwhitespacecharacters.max.chars.per.line	5000	Showing special characters in Context Inspector How many characters per tooltip line
dharmastudio.ui.tooltip.showwhitespacecharacters.space	&#8226;	One of the following values: &#8226;&#8226;
dharmastudio.ui.tooltip.showwhitespacecharacters.tab	&#8633;	One of the following values: &#8633;&#8677;&#8594;&#172;
dharmastudio.ui.tooltip.showwhitespacecharacters.enter	&#8629;	One of the following values: &#8629;&#8626;&#8626;



Parameter	Default Value	Description
dharma.studio.ui.tooltip. showwhitespacecharacters. otherwhitespace	" ";	One of the following values: " ";, " ", "775";

Parameter	Default Value	Description
<p>dharma.scripting.template.Rhino                      dharma.scripting.template.Nashorn</p>		<p>This template shows how to access operation return data, inputs and context data.</p> <p>Examples:</p> <p>To access an input, simply refer to the input name from the Inputs pane. For example: <b>myData = inputName;</b></p> <p>To access the context instance of the current branch:  <b>myBranchData = scriptletBranchContext.get("myBranchVarName");</b></p> <p>To get value associated with the context key myContextKey:  <b>myContextData = scriptletContext.get("myContextKey");</b></p> <p>To get a value associated with the context key myContextKey from the local context:  <b>myContextData = scriptletContext.getLocal("myContextKey");</b></p> <p>To access the return code from a command line or script (note that this is a string and must be converted into an integer for numeric processing):  <b>code = parseInt(scriptletRawResult["Code"]);</b></p> <p>To access the output from the operation (for example, stdout):  <b>data = scriptletRawResult['Output String'];</b></p> <p>To access the error string from the operation (for example, stderr):  <b>error = scriptletRawResult['Error String'];</b></p> <p>Other operations may have different result variables available. To see the list for a particular operation open the <b>Output Field</b> dropdown on the Outputs tab.</p> <p>To set the response of the operation (must match one of the responses from the Responses tab):  <b>scriptletResponse = "success";</b></p> <p>To set the result of the operation:  <b>scriptletResult = "Your Result Here";</b></p> <p>Use the context to store data used in other steps. It is also helpful to place information into the context to examine variables when developing and debugging scriptlets. For example to place the OutputString information into a context key called 'Output':  <b>scriptletContext.putGlobal("Output", data);</b></p> <p>Note that the key and value are both strings. For example, to place 'code' into the context:  <b>scriptletContext.putGlobal("Code", code.toString());</b></p> <p>You can also place data into the local context, which means it will only be available to the current flow, but not to its parent flow or other subflows. To do this, use:  <b>scriptletContext.put("LocalVariable", "LocalValue");</b></p>

Parameter	Default Value	Description
	<p><b>Note:</b> If a flow is larger than 64K, when you try to save the flow, you will see an exception. Also, the exception will occur every time you try to access that flow, even if you restart Studio.</p> <p>You should avoid making scriptlets too long. When the scriptlets are compiled, if they are larger than 64K, you get an exception when trying to save the flow. The exception will occur anytime you try to access that flow, even if you restart Studio.</p>	
dharma.studio.ui.inputinspector.assignfrom.selected	see description	<p>This property allows setting the initial option in the "Assign from Variable" combobox, in the Input Inspector.</p> <p>Possible values:</p> <p>assigned - default when upgrading from an earlier Studio version not-assigned - default when installing Studio 10.20</p>
dharma.studio.ui.inputinspector.assignto.selected	see description	<p>This property allows you to set the initial option to be shown in the "Assign to Variable" combobox, in the Input Inspector.</p> <p>Possible values:</p> <p>assigned - default when upgrading from an earlier Studio version not-assigned - default when installing Studio 10.20</p>
default.behavior.define.input	see description	<p>This property allows you to set the default behavior that will be used when defining inputs in a flow.</p> <p>Possible values:</p> <p>prompt_user- default when upgrading from an earlier Studio version use_constant - default when installing Studio 10.20</p>
dharma.studio.ui.activegrid	false	When set to true, the grid is active by default in the Flow Editor.

Parameter	Default Value	Description
engine.assigner.trigger.repeatInterval	10	
community.home.page.link	<a href="https://hpln.hp.com/group/operations-orchestration">https://hpln.hp.com/group/operations-orchestration</a>	Community Home Page
git.history.page.size	100	Specifies the number of items to be shown in a single Git history page
git.history.changes.length	500	Specifies the default number of changes to show in the <b>Changes under the selected item</b> and <b>Other changes</b> sections in the Git history.
hpln.enabled	true	Specifies whether or not HPLN search is enabled
hpln.connection.timeout	180	Specifies the HPLN connection timeout in seconds. Use 0 for infinite timeout.
hpln.read.timeout	300	Specifies the HPLN read timeout in seconds. Use 0 for infinite timeout.
hpln.connection.url	<a href="https://api.hpln.hp.com/hpln">https://api.hpln.hp.com/hpln</a>	Specifies the HPLN connection URL
hpln.authentication.service.uri	rest/authenticate	Specifies the HPLN authentication service uri
hpln.content.packages.service.uri	rest/contentpackages	Specifies the HPLN content packages service uri
hpln.content.offerings.service.uri	rest/contentofferings	Specifies the HPLN content offerings service uri
hpln.search.service.uri	rest/contentpackages/search	Specifies the HPLN search service uri
hpln.search.service.search.param	search	Specifies the HPLN search service search parameter
hpln.search.service.size.param	pageSize	Specifies the HPLN search service size parameter
hpln.search.service.provider.param	provider	Specifies the HPLN search service provider parameter
hpln.search.service.product.name.param	primaryproduct	Specifies the HPLN search service product name parameter

Parameter	Default Value	Description
hpln.search.service.product.version.value	10.20	Specifies the HPLN search service product version value. This version number is identical to the Studio version. For example, if the Studio version is 10.20, hpln.search.service.product.version.param is 10.20.
hpln.search.results.maximum.count	100	Specifies the maximum number of HPLN search results. This value must not be greater than 100.
local.debugger.prompt.for.run.user	true	When set to true, the author is prompted for a run user on the local debugger
online.content.packs.link	<a href="https://hpln.hp.com/group/operations-orchestration-content-packs">https://hpln.hp.com/group/operations-orchestration-content-packs</a>	Online Content

**Note:** If a property is missing from the **studio.properties** file, the default value is to prompt the user to enter a value.

# Troubleshooting

## 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 check out from the changes pane if you are connected to source control. See [Working with Source Control](#).

##### Why do the Projects and Dependencies panes seem to contain the same items?

The **Projects** pane and the **Dependencies** pane are different:

- The **Projects** pane contains the *editable* flows, operations, and other HP OO objects that you can use in the project.
- The **Dependencies** 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 built-in 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 360.

##### Why can't I create sleep scriptlets?

Sleep scriptlets have been deprecated. In HP OO 10.x, scriptlets must be written in Rhino or Nashorn.

## 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.x

Task	How it was done in HP OO 9.x	How it is done in HP OO 10.x
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 360</a> .
Checking flows into a shared repository	Use the <b>Check In</b> button within Studio.	Save projects locally and commit them to a shared repository, using a source control management tool.  See <a href="#">"Working with Source Control in HP OO Studio" on page 60</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 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 299</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 360</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.

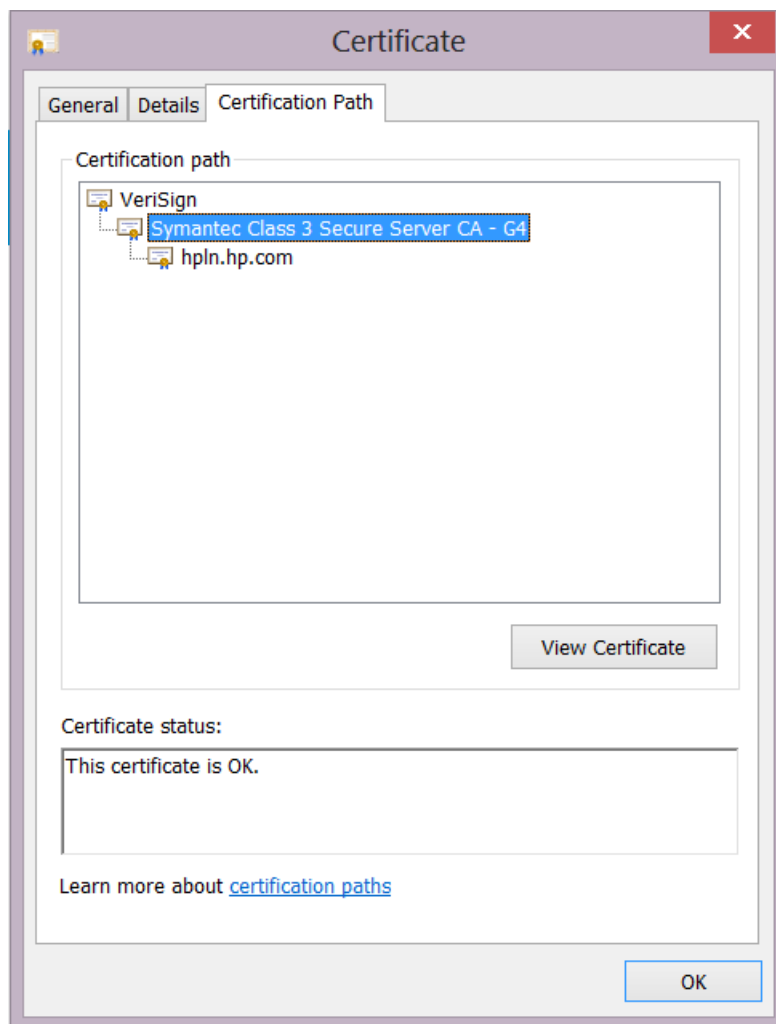
## HPLN Troubleshooting

You may receive the following error when performing a HP Live Network search:

```
sun.security.validator.ValidatorException: PKIX path building failed:  
sun.security.provider.certpath.SunCertPathBuilderException: unable to  
find valid certification path to requested target
```

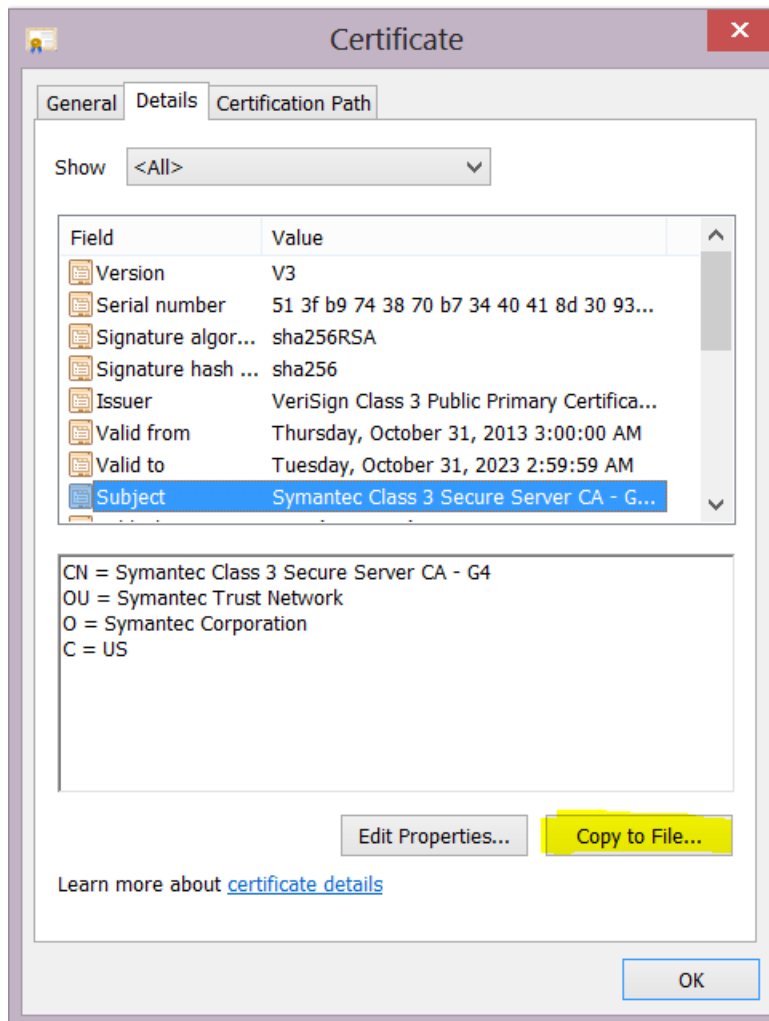
To solve this issue:

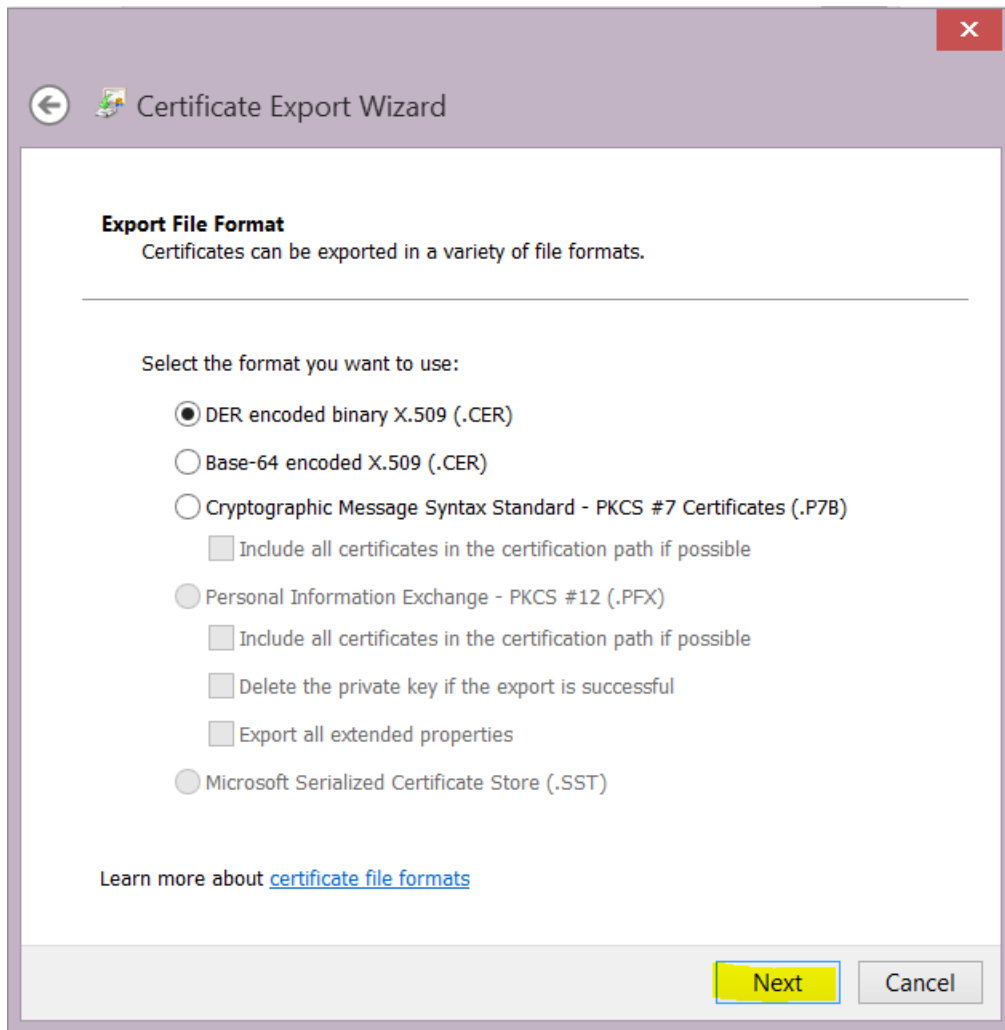
1. Close Studio.
2. Export the HP Live Network root certificate, (the certificate with the name **Symantec Class 3 Secure Server CA - G4**—the second one in the chain) into a file named **hpln-ca.cer** in DER encoded binary X.509 format.

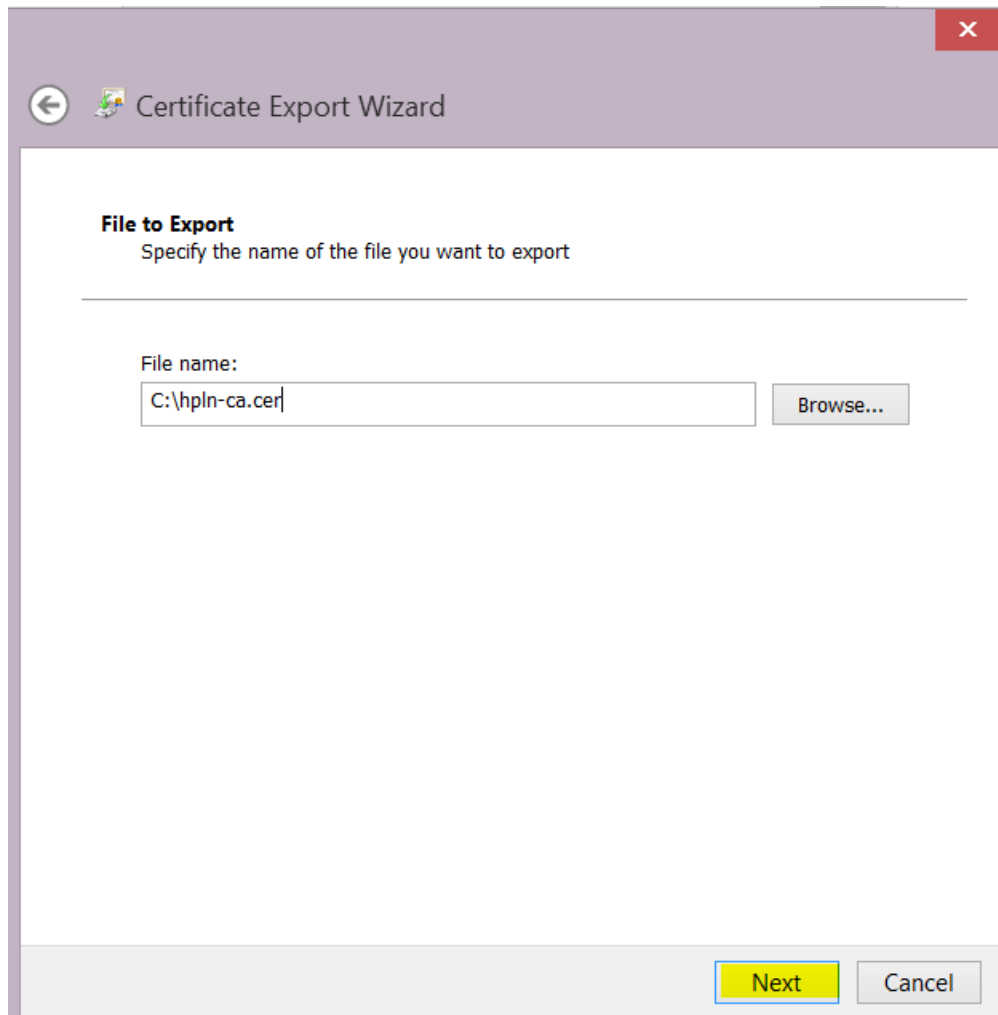


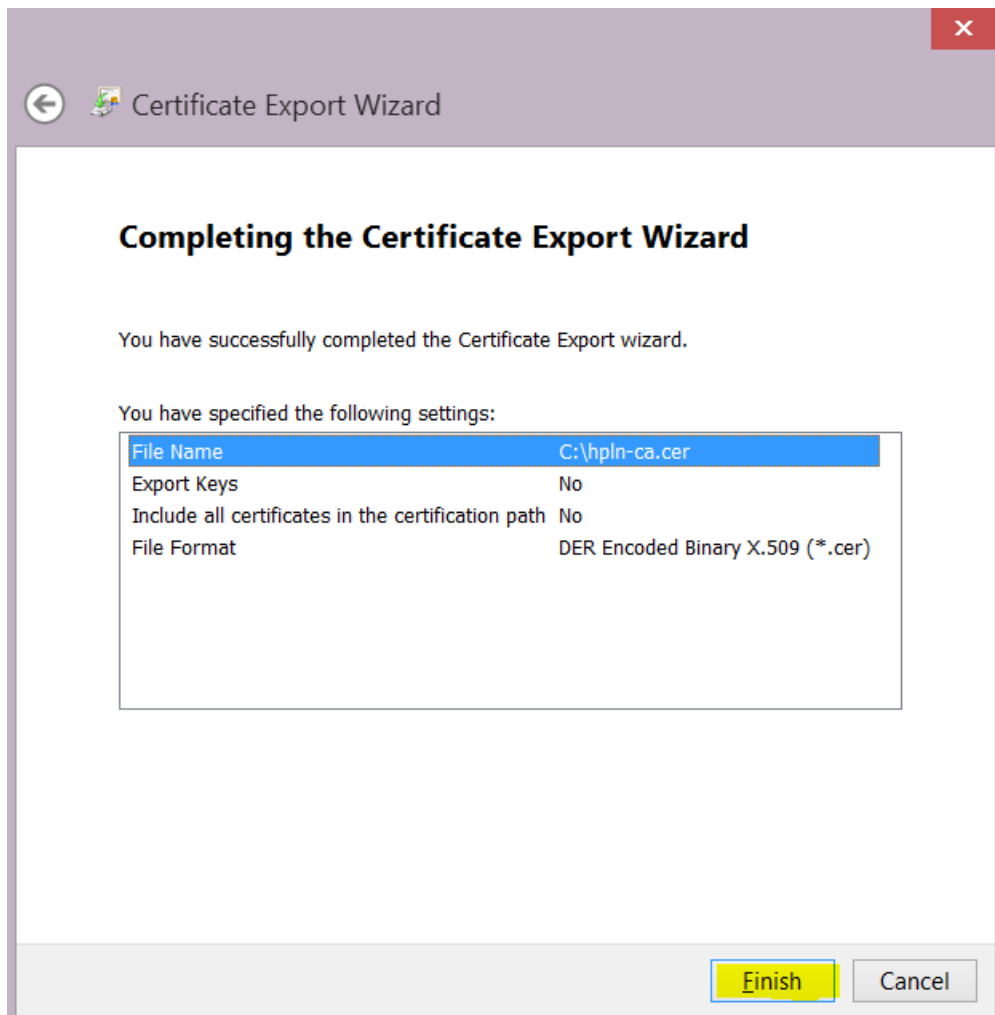


You can use a browser to navigate to <https://api.hpln.hp.com/hpln> and then export the root certificate as shown below:









3. Copy the certificate file **hpln-ca.cer** to the `<oo_install_folder>\studio\var\security` folder.
4. Import the certificate into Studio's TrustStore using the following command:

```
<oo_install_folder>\java\bin\keytool -import -alias hplnroot -keystore  
client.truststore -file hpln-ca.cer
```

**Note:**

- The default password for the keystore is **changeit**. If you changed the default password, when prompted, use the value of the parameter **Djavax.net.ssl.trustStorePassword** from the `<oo_install_folder>\studio\Studio.l4j.ini` file.
- To change the Studio Truststore password, add the property **client.truststore.password** with value the password in obfuscated format to the

Studio.properties file from the “.oo” folder.

```
client.truststore.password=={OBFUSCATED}6L9+NqBjKYp5heuvMEzg0g==
```

If this property is not defined, Studio will fall back to the system property **javax.net.ssl.trustStorePassword** for the truststore password.

5. Restart Studio.

## GIT Troubleshooting

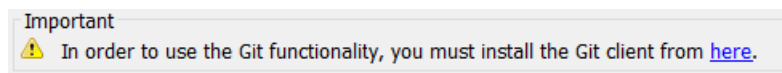
### System cannot find specified file

There may be cases in which you may see the following error when performing Git SCM operations:

```
“Cannot run program "C:\<oo_installation_folder>\studio\Git\bin\git" (in directory "C:\<user_home_folder> \.oo\Workspace"): CreateProcess error=2, The system cannot find the file specified “
```

To fix this:

1. Download the Git Client directly from the link at the bottom of the SCM Connection window:



or:

1. Download the Git client from the following URL:  
<https://github.com/msysgit/msysgit/releases/download/Git-1.9.5-preview20150319/Git-1.9.5-preview20150319.exe>.
2. Save the Git client to **<oo\_installation\_folder>/studio/Git**, so that the **bin** folder is directly under **<oo\_installation\_folder>/studio/Git**. In the Git installation wizard, use the default options.

Alternatively, if you already have a Git client installation with version **git-1.9.5-preview20150319** on your local disk, point Studio to use that Git installation by performing the following steps:

1. Close Studio.
2. Go to the user home folder **C:\Users\<user>\.oo** (the Studio workspace location) and locate the **Studio.properties** file.
3. Modify the **Studio.properties** file by adding the following property at the end of the file:

```
studio.git.installation.location=<git-1.9.5-preview20150319_installation_folder>
```

For example:

```
studio.git.installation.location=C:/Program Files (x86)/Git
```

The **bin** folder should be directly under **C:/Program Files (x86)/Git**. Note that / should be used as a path separator.

4. Save the **Studio.properties** file and start Studio.

If you have another version of the Git client installed, note that you must use the **git-1.9.5-preview20150319** version with Studio.

## Viewing Studio Errors in the Log Viewer

The **Log Viewer** is a useful tool that allows you to see all the errors that have occurred in the current user session in one central location within Studio. The errors are presented in the same way as they are displayed in the **Studio.log** file.

The Log Viewer shows two types of errors:

- Fatal errors (labeled FATAL in Studio.log)
- Errors (labeled ERROR in Studio.log)

**Note:** Warnings (WARN) and Information (INFO) messages are not shown in the Log Viewer but can be seen in the **Studio.log** file.

If the **Studio Log Viewer** option in the Window menu is selected, the Log Viewer tab appears as one of the tabs at the bottom of the Studio window when Studio opens.

## What do you want to do?

### Enable the Log Viewer tab

1. From the Windows menu, select **Studio Log Viewer**.

